SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics Cognition Intelligence

Optimizing a minimal language using Pre-trained Language Models

Ajay Narayanan

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics Cognition Intelligence

Optimizing a minimal language using Pre-trained Language Models

Optimierung einer Minimalsprache mithilfe vorab trainierter Sprachmodelle

Author:	Ajay Narayanan
Examiner:	Vincent Fortuin
Supervisor:	Vincent Fortuin
Submission Date:	15-05-2025

Acknowledgments

I would like to thank Vincent Fortuin, my supervisor for his guidance and support throughout this thesis. I would also like to thank Sara Visconti, who worked with me to develop the language pipeline. I would also like to thank my family and friends for their continued support and encouragement during my work.

I would also like to thank OpenAI for the grant that made this work possible.

Abstract

Human languages evolve as a trade-off between expressiveness and efficiency, constrained by the cognitive and social demands of communication. Constructed languages (ConLangs), in contrast, allow for deliberate design choices that can potentially optimize these trade-offs. This thesis investigates whether Large Language Models (LLMs), trained on vast corpora of natural language, can aid in the systematic construction of minimal languages that retain communicative efficacy. To explore this, we design a modular language generation and evaluation pipeline, encompassing phonology, phonotactics, grammar, and vocabulary. The generated languages are assessed using both traditional NLP evaluation metrics (BLEU, ROUGE, METEOR), as well as language model perplexity and reading comprehension scores on the RACE-C dataset. We also analyze how well these languages conform to linguistic properties such as Zipf's Law. The results suggest that it is possible to simplify several linguistic subsystems, such as reducing phoneme count or grammatical complexity without significantly degrading performance on comprehension or translation tasks. This indicates that LLMs can indeed provide valuable insights into the design of compact, easy-to-learn, and efficient constructed languages.

Contents

1. Introduction

This opening chapter will provide an overview of the research topic, reviewing the study's motivation, the research questions, and the methodology. The chapter will also provide a brief outline of the structure of the thesis.

1.1. Motivation

The goal of human language, much like any other human activity, is to aid our survival and propagation. It helps us to collaborate, compete and influence others [Lev22]. Efficiency is another hallmark of all living beings, as a product of biological evolution [Ha10]. Thus, languages form from a trade-off between expressiveness and efficiency. They tend to follow rules like Zipf's Law of Abbreviations [Pia14], which states that words that are more frequently used tend to be shorter than sparsely used words.

This is, of course, an idealization. In the real world, speakers must talk in noisy environments, which means that languages often have redundancy built in. This redundancy helps in error correction and understanding in such noisy environments. In addition, natural languages are not an optimal code in terms of information theory. This is due to the fact that human languages are constrained by the limitations of incremental language processing, which enforces constraints which force systematicity, compositionality and concatenation as means of combination [FH22].

Constructed Languages (ConLangs) are languages constructed by an individual or group of individuals rather than having evolved naturally [Sch21]. ConLangs are used for various purposes, such as to serve as an international auxiliary language (e.g. Esperanto), to create a fictional world (Dothraki, Klingon, or Quenya), to be logically rigorous, (e.g. Lojban), or simple and easy to learn (e.g., Toki Pona). Some, like Ithkuil [24] are designed to express more profound thoughts briefly but clearly.

Naturally, the next question that arises is whether we could design a language that is more optimal or efficient than natural languages. This question itself is more challenging than it seems. What does it mean for a language to be optimal? Is it efficient in terms of information theory? Is it ease of learning? Is it expressiveness? Is it the ability to convey complex thoughts? Is it the ability to be understood in even the most noisy environments? In addition, how do we measure these properties? Therefore, the first motivation of this thesis is to explore what it means for a language to be optimal and whether we can design a language that is more optimal than natural languages. To develop this language, one must define its phonology, orthography, morphology, syntax, and vocabulary.

The second motivation of this thesis is to explore whether Large Language Models (LLMs) can provide insights into this problem. In the process of learning, Large Language Models and Machine Learning models in general, learn to encode various features. These features potentially encode information about the structure of language itself, and facts about our world. Trained on a large and varied corpus, they could encode information about nearly everything humans communicate about. Thus, it is possible that LLMs could provide insights that could help us design a more optimal language.

1.2. Research Questions

The primary research questions of this thesis are:

1. What are the metrics we can use to evaluate natural languages?
2. Given these metrics, can we design an optimal constructed language?
3. What factors affect these metrics?
4. Can we use LLMs to generate a language that is more optimal than natural languages?

To test these hypotheses, we setup a modular pipeline for generating and evaluating constructed languages. The code for this pipeline is available at `https://github.com/fortuinlab/conlang`.

1.3. Structure of the Thesis

This thesis is divided into 6 chapters. Chapter 2 introduces the necessary background information prerequisite for the main topic. It also details the various literature surveyed as part of the research. Chapter 3 introduces the research methodology and the primary research questions. Chapter 4 describes the implementation of the pipeline, and the various modules used in the pipeline. Chapter 5 details the results of the experiments conducted using the pipeline, which are then discussed in Chapter 6, along with the conclusion and areas for future research. Chapter 7 details the individual contributions of the author to the research, and the codebase.

2. Background

In this chapter, we will provide an overview of the key concepts and theories that are relevant to our work. We will also mention key background literature related to this work.

2.1. Linguistics

Linguistics, the scientific study of languages is a broad and complex field encompassing various subfields. Although a comprehensive summary of Linguistics is beyond the scope of this thesis, we will briefly discuss some of the subfields and key concepts that are relevant to our work. [TS07] provides a glossary of linguistic terms that can be useful for readers interested in a more detailed overview of the field.

2.1.1. Phonetics and Phonology

Phonetics is the study of the physical sounds of human speech, their production, transmission and reception. [TS07]. The International Phonetic Alphabet (IPA) is a standardized system of phonetic notation that represents the sounds of spoken language. The system is based on the assumption that speech can be represented partly as a sequence of discrete sounds or *segments* [99]. In addition, the IPA also includes symbols for suprasegmental features such as stress and intonation. The full IPA Chart (reproduced here in A.1) shows all the symbols and diacritics used to represent sounds in the IPA. Sounds and words can be transcribed in IPA using [ ] brackets. For example, the sounds for the word *this* can be transcribed as [ðɪs]. The IPA helps linguistics transcribe sounds in a language-agnostic way, allowing them to compare sounds across languages. The IPA Handbook [99] provides a comprehensive guide to the use of the IPA.

Phonology is the study of the sound systems of languages, including the patterns of sounds and the rules that govern their distribution. [TS07]. The key difference in the disciplines is driven by the concept of a *phoneme*. A phoneme is an abstract unit of sound that can distinguished by a native speaker of a language. Phonemes and Phonemic transcriptions are represented using slashes / /. The key points about phonemes are:

1. Letters do not necessarily correspond to phonemes. For example, the English word *this* has four letters but 3 phonemes (/ðɪs/).
2. Phonemes can be realized as different sounds in different contexts. For example, the English phoneme /p/ can be realized as [p^h] in the word *pin*([p^hɪn]) and [p] in the word *spin*([spɪn]). i.e. in English, the sounds [p] and [p^h] are *allophones* of the phoneme /p/.
3. Two sounds are considered different phonemes if changing them can change the meaning of a word. e.g. [dɛn] *den* and [ðɛn] *then* are distinct words in English.

Phonotactics defines the rules that govern the permissible sound sequences in a language [TS07]. For example, in English, the sequence /bl/ is permissible at the beginning of a word (e.g. *bled*) but not the sequence /bn/. Languages usually modify loanwords to fit their own phonotactic constraints. For example, the English word *beer* is borrowed into Japanese as *biru*.

2.1.2. Morphology

Morphology is the study of the structure of words and the rules that govern the formation of words in a language [TS07]. Most studies of morphology focus on the concept of a *morpheme*, the smallest unit of meaning in a language. For example, the word *unhappiness* can be broken down into three morphemes: *un-*, *happy* and *-ness*. Morphemes can be free or bound. Free morphemes can stand alone as words (e.g. *happy*) while bound morphemes must be attached to other morphemes (e.g. *-ness*).

Morphology can be further divided into **inflectional** and **derivational** morphology. **Inflectional morphology** involves the modification of a word for grammatical purposes such as tense, aspect, mood, number, e.g. the English verb *walk* can be inflected to *walked*, *walks*, *walking*, etc. **Derivational morphology** involves the creation of new words from existing words. For example, the English noun *happiness* can be derived from the adjective *happiness* by adding the suffix *-ness*.

Lexicon

The **lexicon** of a language is the vocabulary of a language, i.e. the total set of words available for a speaker. It is better to consider the lexicon, not as a list of word, but a set of lexical resources including morphemes, and processes to construct words from these resources [TS07].

2.1.3. Grammar

Grammar is the set of rules that govern the structure of sentences in a language. Traditional Grammar describes certain terms for basic grammatical components such as *article*, *adjective*, *noun*, etc known as *parts of speech* [Yul20].

1. **Nouns** are words that refer to people, places, things, or abstract ideas, as if they were objects. For example, *cat*, *house*, and *happiness* are all nouns.
2. **Verbs** are words that express the actions or states of nouns. For example, *run*, *is*, and *happen* are all verbs.
3. **Adjectives** are words that describe or modify nouns. For example, *happy*, *red*, and *tall* are all adjectives.
4. **Adverbs** are words that describe or modify verbs, adjectives, or other adverbs. For example, *really*, *very*, and *well* are all adverbs.
5. **Articles** are words that define a noun as specific or unspecific. For example, *the* is a definite article, while *a* and *an* are indefinite articles.
6. **Pronouns** are words that take the place of noun phrases, typically when they are already known. For example, *he*, *she*, and *they* are all pronouns.
7. **Prepositions** are words that show the relationship between a noun or pronoun and other words in a sentence. For example, *in*, *on*, and *at* are all prepositions.
8. **Conjunctions** are words that connect words, phrases, or clauses, and indicate the relationship between them. For example, *and*, *but*, and *or* are all conjunctions.

Sometimes, parts of speech exhibit multiple forms, used in different grammatical circumstances. Each of these forms indicate a certain *grammatical category* or *feature*. For example, in English, verbs can be inflected for tense, aspect, mood, person, number, etc. This is known as *agreement*.

A language may choose to explicitly mark these features, i.e. *grammaticalize* them [Ros10]. All features can be expressed in any language (perhaps by adding explicit information), but every language chooses to express only a subset of these features grammatically.

2.1.4. Grammatical Features

Grammatical features or categories provide some extra information about the sentence, and different parts of speech may exhibit different forms to indicate these features. we will briefly discuss some of the most common grammatical features.

Grammatical Gender

In many languages, nouns are often classified into different classes, and different parts of speech will often agree with the specific class. These classes are called Grammatical Gender, and it often need not have anything to do with sex or gender. Languages with gender may only have 2 gender classes, but can also have many more. The gender assignment of many objects are often arbitrary.

Grammatical Number

Grammatical Number is a grammatical category that expresses count distinctions. The most common distinction is between singular (one) and plural (many), but some languages also have dual (two), trial (three), paucal (few), and other forms. For example, in English, the noun *cat* is singular, while *cats* is plural.

Grammatical Case

The **Grammatical Case** indicates one or more functions of a noun or noun phrase in a sentence. Many different cases have been identified in the worlds languages, such as

1. **Nominative** case, which indicates the subject of a verb.
2. **Accusative** case, which indicates the direct object of a verb.
3. **Dative** case, which indicates the indirect object of a verb.
4. **Genitive** case, which indicates possession.
5. **Locative** case, which indicates location.
6. **Instrumental** case, which indicates the means by which an action is performed.

These descriptions are not exact, and precise distinctions can heavily depend on the specific language.

Tense, Aspect and Mood

Tense, aspect and modality all provide some kind of information that is temporal in nature, or tell us about the status of the action or verb. They are often grouped together as **TAM** (Tense, Aspect, Modality).

Tense is a grammatical category that indicates the time at which an action takes place. The most common tenses are past, present, and future. For example, in English, the verb *walk* can be inflected to *walked* (past), *walks* (present), and *will walk* (future).

Aspect is a grammatical category that indicates the temporal structure of the action or event described by a verb. It indicates for example, whether the action is bounded, and unitary, or continuous or habitual. For example, in English, the sentences *She danced* and *She was dancing* have different aspects. They are both in the past tense, but the first sentence indicates a *perfective*, or completed aspect, while the second sentence indicates an *continuous*, or *progressive* aspect.

Modality is a grammatical category that indicates the speaker's attitude towards the action or event described by a verb. Modern linguists usually associate it with the expression of obligation, permission, prohibition, necessity, possibility and ability [TS07]. In English, modality is primarily expressed using Auxiliary verbs, such as *can*, *may*, *must*, etc. For example, the sentence *He can dance* indicates ability, while the sentence *He must dance* indicates obligation.

Grammatical Person

Grammatical Person is a grammatical category that indicates the different relationships between the speaker, the listener, and others in the discourse. Languages typically indicate this relationship using pronouns. The most common distinctions are between first person (the speaker), second person (the listener), and third person (others). For example, in English, the pronouns *I* (first person), *you* (second person), and *he/she/they/it* (third person) indicate the grammatical person.

Some languages also have a distinction in *clusivity* for the first person plural pronoun. The inclusive form includes the listener, while the exclusive form does not. For example in Malayalam, the pronoun *nammal* includes the listener, while the pronoun *njangal* does not.

2.1.5. Syntax

Syntax is the study of the structure of sentences and the rules that govern the formation of sentences in a language [TS07]. The goal of Syntactic Analysis is to have a finite set of rules that could be used to generate potentially infinite sentences. This set of rules is known as a **Generative Grammar** [Yul20]. We move from the concepts of Nouns to Noun Phrases, and Verbs to Verb Phrases, and so on.

A Noun Phrase is a phrase that is interchangeable with a noun. Take for example the sentence:

__________ bought a new car.

The cloze in the sentence could be filled with a noun, like "John", but also by say , "The young man".

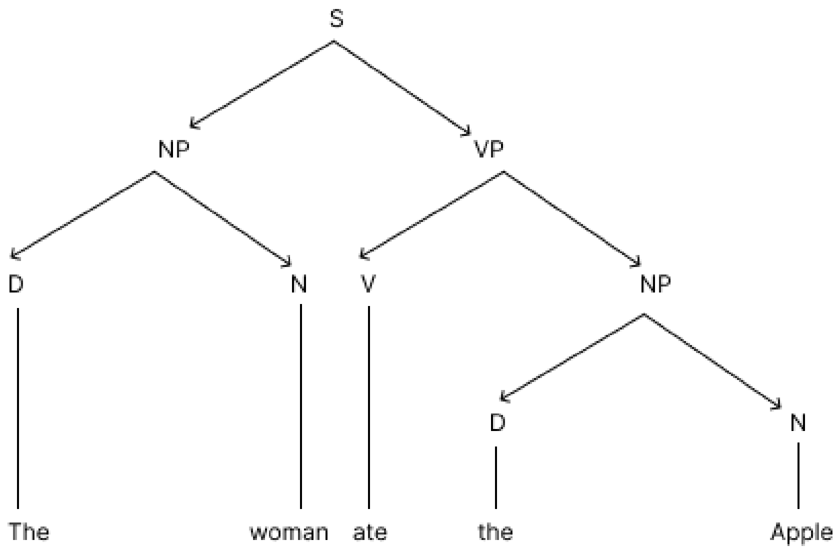


Figure 2.1.: Parse Tree for the sentence *The woman ate the Apple*

With these definitions in mind, we could define a sentence as a Noun Phrase followed by a Verb Phrase. We can also have production rules for Noun Phrases, Verb Phrases, and so on. For example, a Noun Phrase could be defined as a determiner followed by an adjective followed by a noun. With these rules, known as **Phrase Structure Rules**, we can generate a tree structure for a sentence, known as a **Parse Tree** [JM25].

2.2. Constructed Languages

Constructed Languages are languages that have not naturally evolved, but were artificially constructed. Some conlangs are created for fictional word-building, like *Quenya* and *Sindarin* from J.R.R. Tolkien's Middle-Earth, or *Dothraki* and *High Valyrian* from George R.R. Martin's A Song of Ice and Fire. Others are created for international communication, like *Esperanto* and *Volapük*. Some conlangs are created to be more logical or efficient, like *Lojban* and *Ithkuil* , or easy to learn, like *Toki Pona*.

Although people have been likely creating languages since time immemorial, *Lingua Ignota*, created by the nun Hildegard von Bingen in 12 CE [Sch21]. Language Creation became popularized in the mainstream by J.R.R. Tolkien, who created several languages for his works, and his technique has influenced many conlangers since [Pet15]. In the early 90s, the internet allowed for the formation of a community of conlangers, who share their work and techniques online.

2.2.1. The Process of Language Creation

There has been a strong online community of language creators, called conlangers since the 1990s [Pet15]. The community has created a wealth of resources for language creation. In particular, [Pet15] and [Ros10] provide a comprehensive overview of the process of language creation. The process starts with the creation of a phonetic inventory, and phonotactics, which gives the language a distinct character. The next steps involve, word building, grammar, semantics and pragmatics [Ros10].

2.3. Evaluation

2.3.1. Evaluation of Machine Translations

Evaluation of machine translations is a complex task, and is essential for assessing the accuracy and fluency of the translations. Human evaluations are often expensive and time-consuming [Pap+02], and are not always feasible for large datasets. As a result, many researchers have developed automatic evaluation metrics to assess the quality of machine translations. Classic methods like BLEU [Pap+02] measures the similarity between the machine translation and a reference translation by comparing n-grams. ROUGE [Lin04] is another popular metric that measures recall as opposed to precision. It is often used for evaluating the quality of summaries, but can also be used for machine translation evaluation. METEOR [BL05] improves upon such methods by for example, considering synonyms and stemming.

We use these machine translation evaluation metrics by comparing the detranslated text with the original text. Although the actual values for these metrics would be dependent on the model and its parameters, it would still be useful to compare the performance between different generated conlangs.

BLEU: Bilingual Evaluation Understudy

BLEU (Bilingual Evaluation Understudy) [Pap+02] is one of the most widely used automatic metrics for evaluating machine translations. It measures the similarity between a machine translations and reference translations by analyzing their *n-gram* overlap. The BLEU score is given by:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2.1}$$

where p_n is the geometric average of the precision of n-grams, w_n are weights assigned to different n-grams, and BP (the brevity penalty) penalizes short translations to prevent artificially high scores. BLEU focuses primarily on precision but does not consider recall or semantic meaning.

ROUGE: Recall-Oriented Understudy for Gisting Evaluation

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [Lin04] is a set of metrics primarily used for text summarization but also applicable to machine translation. Unlike BLEU, which focuses on precision, ROUGE focuses on recall, i.e. how much of the reference translation appears in the generated text. The most common variant, ROUGE-N, computes the recall of n-gram matches:

$$\text{ROUGE-N} = \frac{\sum_{s\in\text{ref}} \sum_{gram_n \in s} \text{count}_{match}(gram_n)}{\sum_{s\in\text{ref}} \sum_{gram_n \in s} \text{count}(gram_n)} \tag{2.2}$$

Another variant, ROUGE-L, measures the longest common subsequence (LCS) between the reference and candidate translations, capturing fluency better. ROUGE is especially useful for evaluating translations with different word orders but similar meanings.

METEOR: Metric for Evaluation of Translation with Explicit ORdering

METEOR (Metric for Evaluation of Translation with Explicit ORdering) addresses some of BLEU's limitations by incorporating recall, stemming, synonym matching, and word order penalties. METEOR aligns words between candidate and reference translations using exact matches, stemmed matches, and synonym matches. The metric is computed as:

$$\text{METEOR} = F_{mean} \cdot (1 - Penalty) \tag{2.3}$$

where F_{mean} is the harmonic mean of precision and recall, and Penalty reduces the score for word order mismatches. METEOR can correlate better with human judgments than BLEU due to its ability to consider semantic variations.

2.3.2. Information Theoretic Evaluation of Languages

Natural language, being a means of communication consists primarily of information transmission [DB20]. There have been several attempts to quantify the information content of a language using information theory. Shannon [Sha51] introduced a new method of estimating the entropy and in turn the redundancy of a language. Entropy in a sense measures the average amount of information for each letter in a text. Natural language however, is not a minimal length source code. It operates under the constraints of incremental language processing, which enforces them to be systematic, compositional and use concatenation as means of combination [FH22]. Redundancy often serves another important purpose, which is to help in error correction and understanding in noisy environments [Gib+19].

2.3.3. Reading Comprehension Evaluation of Language Models

One of the primary goals of Natural Language Processing is teaching computers to read and understand the meaning of text. Reading comprehension thus presents itself as a suitable benchmark to evaluate a model's language understanding ability [Zen+20]. The goal of a reading comprehension task is to require the model to read one or more text passages and then answers questions about them.

MRC tasks are generally classified into four categories.

1. **Cloze Style** tasks require the model to fill in the blanks in a passage.
2. **Multiple-Choice** tasks require the model to select the correct answer from a set of options.
3. **Span Prediction** tasks require the model to extract a span of text from the passage that answers the question.
4. **Freeform** tasks require the model to generate a free-form answer to the question.

In our work, we decided to use Multiple-Choice tasks as a proxy for evaluating the constructed language. By evaluating the dataset on a model, first with the original passage and then with the detranslated passage, we get a proxy measure for the information loss in the translation process. Although some of this loss may be due to the model's inability to understand the constructed language, we can still use this for comparative analysis of different constructed languages.

RACE-C Dataset

The RACE-C Dataset [LLY19] is a large-scale reading comprehension dataset based on college english exams from China. It improves upon the previous RACE dataset [Lai+17] by providing a more challenging set of questions and answers. It consists of 4275 passages and 14122 questions. Figure 2.2 shows a sample article and question from the dataset.

People have been painting pictures for at least 30,000 years. The earliest pictures were painted by people who hunted animals. They used to paint pictures of the animals they wanted to catch and kill. Pictures of this kind have been found on the walls of caves in France and Spain. No one knows why they were painted there. Perhaps the painters thought that their pictures would help them to catch these animals. Or perhaps human beings have always wanted to tell stories in pictures.

. . .

These days, we can write down a story, or record information, without using pictures. But we still need pictures of all kinds: drawing, photographs, signs and diagrams. We find them everywhere: in books and newspapers, in the street, and on the walls of the places where we live and work. Pictures help us to understand and remember things more easily, and they can make a story much more interesting.

Question: Pictures of animals were painted on the walls of caves in France and Spain because ________________.

Options:

A. the hunters wanted to see the pictures

B. the painters were animal lovers

C. the painters wanted to show imagination

D. the pictures were thought to be helpful

Correct Answer: D

Figure 2.2.: Sample Article and Comprehension Question from RACE-C

2.3.4. Zipf's Law

In 1935, George Zipf made an observation which he claimed to be a universal property of human languages: The more frequent a word, the shorter it tends to be [Kan+17]. This is known as **Zipf's Law of Abbreviation**, or simply **Zipf's Law**.

In mathematical terms,

$$f(r) \propto \frac{1}{r^{\alpha}} \tag{2.4}$$

Where $f(r)$ is the frequency of the r^{th} most frequent word, and α is a constant known as *Zipf's Exponent*. For natural languages, the value of α is noted to be around 1.

2.4. Language Models and Embeddings

2.4.1. Representation Learning

The performance of NLP tasks, or any machine learning tasks in general is dependent on the choice of data representation [BCV14]. Classical methods often involve painstakingly hand-crafting features for each task, which is often not feasible for large datasets. **Representation learning** presents a way to automatically learn representations from the data, which can then be used for various downstream tasks.

Coming specifically to NLP, Words that occur in similar contexts tend to have similar meanings [Har54]. This is known as the **Distributional Hypothesis**. This hypothesis is the basis for **Vector Semantics**, the idea of representing words as a point in a high-dimensional space, where the distance between points represents the semantic similarity between words [JM25]. These vectors are called **embeddings**. This idea has been extended to subword units [Wu+16], sentences, and even larger pieces of text.

Embeddings can be broadly classified into two categories: **static** and **contextual** embeddings. Static embeddings, such as Word2Vec [Mik+13] or GloVe [PSM14] are fixed representations for each word, regardless of the context in which they appear. For example, the word *bank* would have the same representation in both the sentences *I went to the bank to deposit money* and *The river bank was flooded*. Contextual embedding methods like ELMo [Pet+18], BERT [Dev+19], and GPT-3 [Bro+20], on the other hand, create dynamic representations that depend on the context in which the word appears.

2.4.2. Large Language Models

In NLP, Language Modeling aims to model the generative likelihood of word sequences to to predict the probabilities of the next or missing word in a sequence [Zha+25]. Traditionally, language models were based on the markov assumption, predicting the next word in a sequence based on the previous *n* words, and they were called *n*-gram models [JM25]. Neural Language Models, on the other hand, use neural networks to learn the probability distribution of the next word in a sequence, and are trained on large corpora of text. Initially they were based on recurrent neural networks such as LSTMs [HS97], but have since evolved to use transformer architectures [Vas+23].

The transformer architecture is based on the self-attention mechanism, which allows the model to weigh the importance of different tokens in a sequence when making predictions. This allows the model to capture long-range dependencies and relationships between tokens, which is crucial for understanding natural language. These models are often called Pre-trained Language Models (PLMs) because they are initially trained on large corpora of text, called *pre-training*, and then *fine-tuned* on specific tasks, such as machine translation or reading comprehension. Researchers have noted that increasing the size of these models and the amount of training data leads to better performance on a wide range of NLP tasks, leading to the rise of what are called **Large Language Models** (LLMs) [Bro+20].

In our work, we integrate multiple LLMs into our workflow.

GPT-4o

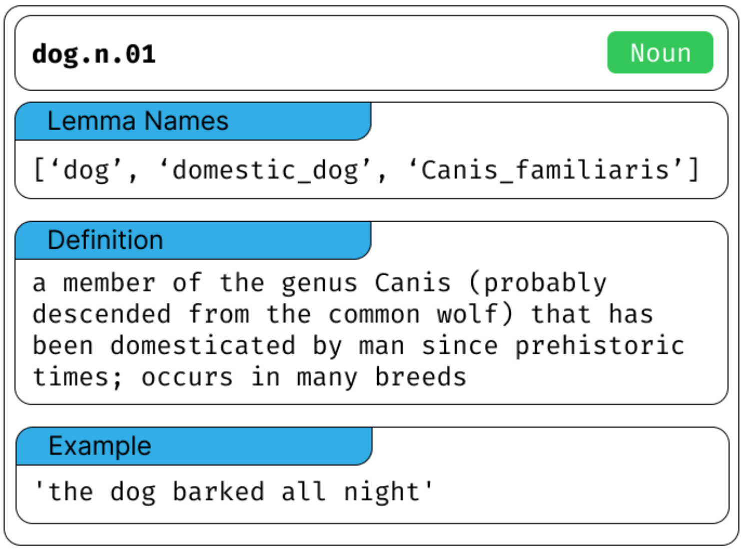
gpt-4o is a decoder based LLM from the GPT family of models [Bro+20]. Although the exact architecture and training data are not publicly available, details about the previous versions of the GPT model have been published. The GPT models are based on the transformer architecture, and are a decoder-only model. They are trained using a causal language modeling objective, which means that they predict the next word in a sequence given the previous words.

DeepSeek-r1

The DeepSeek family of models [Dee+25] are a set of reasoning models whose performance has been improved via reinforecement learning. These models also include smaller models which are distilled from larger models.

BERT

BERT (Bidirectional Encoder Representations from Transformers) [Dev+19] is an encoder based LLM from the BERT family of models. BERT is based on the transformer architecture, and is a bidirectional model. It is trained using a masked language modeling objective, which means that it predicts the missing words in a sequence given the surrounding words. BERT has been shown to achieve state-of-the-art performance on a



dog.n.01 Noun

Lemma Names

```
[‘dog’, ‘domestic_dog’, ‘Canis_familiaris’]
```

Definition

```
a member of the genus Canis (probably
descended from the common wolf) that has
been domesticated by man since prehistoric
times; occurs in many breeds
```

Example

```
'the dog barked all night'
```

Figure 2.3.: WordNet synset for the word *dog.n.01*.

wide range of NLP tasks, including reading comprehension, named entity recognition, and sentiment analysis [RKR20].

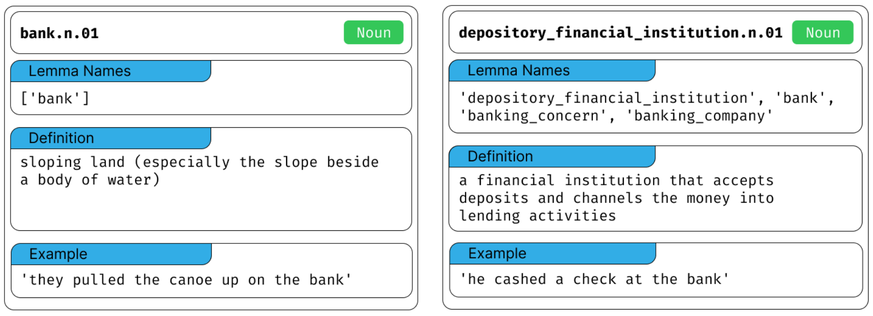
2.4.3. Clustering Algorithms

Clustering can be defined as the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups [Jai10]. There have been several clustering algorithms proposed in the literature, including K-Means [Llo82], DBSCAN [Est+96], and HDBSCAN [CMS13].

Clustering algorithms can be broadly classified into two categories: **partitional** and **hierarchical** clustering [Jai10]. hierarchical clustering algorithms recursively find nested clusters, in either a bottoms-up or top-down fashion. Partitional clustering algorithms, on the other hand, find all clusters in a single pass and do not create a hierarchy.

2.5. WordNet

In order to facilitate our pipeline with proper embeddings, We use WordNet [Mil94], a large lexical database of English. WordNet groups words into sets of synonyms called *synsets*, and provides short definitions and usage examples. It also records the various semantic relations between these synonym sets.


bank.n.01 Noun

Lemma Names
['bank']

Definition
sloping land (especially the slope beside a body of water)

Example
'they pulled the canoe up on the bank'

depository_financial_institution.n.01 Noun

Lemma Names
'depository_financial_institution', 'bank', 'banking_concern', 'banking_company'

Definition
a financial institution that accepts deposits and channels the money into lending activities

Example
'he cashed a check at the bank'

Figure 2.4.: Comparing two senses for the word *bank*.

WordNet does not just interlink word forms, but specific senses of each word, which provides a more fine-grained representation of meaning.

2.6. Exquisite Corpus

The Exquisite Corpus [25] is project to create a good varied multilingual corpus data, which is used for other tools such as the *wordfreq* python package. The data comes from multiple sources including Wikipedia, Subtitles, News, Books, websites, etc.

3. Methodology

3.1. Methodology

This chapter outlines the methodological approach we used to construct and evaluate constructed languages. The methodology follows the step by step approach from existing literature on language construction [Pet15; Ros10]. The language description contains the phonemic inventory, phonotactic rules, grammar and vocabulary of the new language. We also generate translation of an existing text, which can then be used for evaluation.

3.1.1. Research Design

We adopt a modular, computational experimental design. The overall aim is to investigate whether LLMs can be systematically guided to generate conlangs that exhibit properties found in natural human languages. To this end, a pipeline-based framework was developed, consisting of discrete modules for phonology, morphology, syntax, and vocabulary generation, followed by a suite of evaluation metrics. This modular design allows for targeted ablative analysis of each linguistic subsystem. This would also allow further extensions to the pipeline in the future.

3.1.2. Modular Language Generation Pipeline

The constructed language is generated through a modular pipeline, where each module can modify and append to the previous module's output. The design purposefully avoids requiring a strict ordering of modules, and each module can specify what it requires from the previous modules. Each module represents an independent variable in the generation process, which are manipulated to compare and contrast the generated languages, and answer the following research questions:

1. How does the number of phonemes in the phonemic inventory affect the generated language?
2. How do the phonotactic rules affect the generated language?
3. How do different grammatical structures affect the generated language?

4. How does different vocabulary generation methods affect the generated language?

3.1.3. Evaluation Methodology

Once the linguistic modules generate a language, we can run the evaluator modules to benchmark the results. Each evaluator module can run its evaluations and the results can be stored along with the language in the results. In particular, we run the following evaluations:

Information Loss Metrics

The Information loss metrics evaluate how well the generated language can retain information from the original text. The different benchmarks we use for this metric are:

1. **Machine Translation Scores**: We translate a text from English to the generated language and back to English, and evaluate the translation score using BLEU [Pap+02], ROUGE [Lin04] and METEOR [BL05]. The translation is done using a pre-trained LLM, and the scores are calculated using the generated text.
2. **Reading Comprehension Scores**: We evaluate the Reading Comprehension score of the generated text using a pre-trained LLM. The LLM is given a passage in both English and the generated language and a set of questions, and it is evaluated on how well it can answer the questions. The score is calculated as the percentage of questions answered correctly.

Simplicity Metrics

The simplicity metrics evaluate how simple the generated language is.

1. **BERT Fine-tuning**: We fine-tune a BERT like model on the generated language and evaluate the perplexity of the model.
2. **Vocabulary Size**: The size of the vocabulary.
3. **Phonemic Inventory**: The number of phonemes in the phonemic inventory.
4. **Phonotactic Complexity**: The complexity of the phonotactic rules.

Zipf's Law Metric

The Zipf's law evaluation evaluates how well the generated language follows Zipf's law. We calculate the Zipf's exponent for the generated text and compare it to the original text.

4. Implementation and Experimental Setup

In this chapter, we will describe the pipeline we have set up to generate constructed languages. The codebase is built in Python, and we used Poetry for Dependency Management. The pipeline is modular, to allow us to perform ablation studies.

4.1. Pipeline Overview

In order to generate constructed languages, we setup a modular pipeline. Figure 4.1 shows the Structure of a generation pipeline. Although we ended up having the modules in a specific order, the codebase is flexible enough that we can reorder modules, as long as the input to any module has all the features required for its execution. This is facilitated by the `LanguageDescription` class, which contains all the features of a language. This class is piped through the modules, and each module can add or modify features of the language. Each module also generates a results `dict` which is stored as a `JSON` file after the run. If the module requires a certain element of the description to be generated beforehand, it check for those features and can throw an error if they are not present.

A pipeline can be setup by subclassing the `Pipeline` class, which takes care of executing the modules and saving the results. Each module of the pipeline is a subclass of the `Module` class, which has an `execute` method that takes a `LanguageDescription` object and other optional arguments, and returns the modified `LanguageDescription` object and other optional results.

During the course of development, we found ourselves using more or less similar modules in most of our experiments. We have identified the following modules that are common to most setups:

4.1.1. Phonetics Modules

The goal of a Phonetics Module is to generate the phonemic inventory of the language. The module configures the `PhonemeDataInventory` class, which contains a list of `PhonemeData`. The phoneme segments for this class are based on PHOIBLE [MM19] segments, with a `GlyphID` corresponding to their database. For our purposes, we also

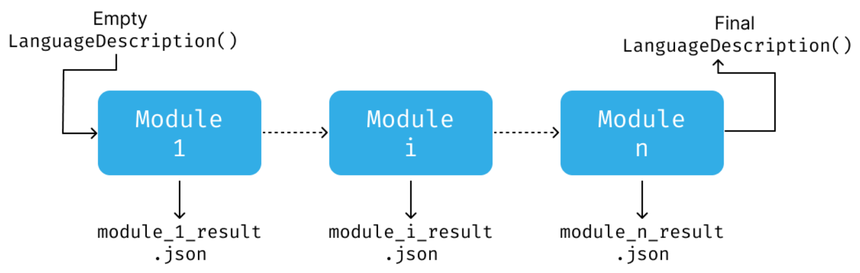


Figure 4.1.: The pipeline structure for generating constructed languages. The modules are run in order, and each module can add or modify features of the language.

implemented an `alphabet` attribute, to have a simpler representation for phonemes that are hard to read or write. This was useful for debugging and visualization purposes.

MostCommonPhonemes Module

The `MostCommonPhonemesModule` module takes the number of consonants and vowels as arguments, and generates a phonemic inventory based on the most common phonemes in the world languages, based on PHOIBLE [MM19] data.

4.1.2. Phonotactics Modules

The goal of a Phonotactics Module is to generate the phonotactic rules of the language. The module configures the `PhonotacticData` class, which specifies the rules for syllable structure and phonotactic constraints for word beginnings and endings.

BasicPhonotacticsModule

The `BasicPhonotacticsModule` module takes a Phonotactics string and generates a basic set of phonotactic rules. It supports defining the syllable structure with a string of consonants and vowels, where `C` represents a consonant and `V` represents a vowel. The module also supports defining optional consonants and vowels, by placing them in parentheses. For example, the string `(C)VC` would generate a syllable structure with an optional consonant at the beginning and a mandatory vowel and consonant at the end.

CustomPhonotacticsModule

The `CustomPhonotacticsModule` module supports everything that the `BasicPhonotacticsModule` does, but also allows for defining specific phonemes or set of phonemes that are allowed or not allowed in certain positions. For example, the string `C[e,o]` would generate a syllable structure with an consonant at the beginning and a mandatory vowel that is either `e` or `o` at the end.

4.1.3. Syllable Builder Module

The Syllable builder modules combines the phonemes generated by the Phonetics Module and the phonotactic rules generated by the Phonotactics Module to generate all the possible syllables in the language.

4.1.4. Grammar Modules

The grammar modules generate the grammatical rules of the language. This is done by configuring the `GrammaticalFeatures` class, which contains a list of features that inherit from either the `AbstractFeature` class or the `AbstractPOS` class. Any feature that inherits from either of these classes implements serialization and deserialization methods, and most importantly the `prompt_string` method, which generates the string representation of the feature.

BaselineAgglutinativeGrammarModule

This is the baseline grammar module, which describes a basic agglutinative grammar. Nouns are inflected for number, case, and gender. Verbs are inflected for tense, aspect and mood.

BaselineIsolatingGrammarModule

This grammar module describes an isolating grammar, where each feature is represented by independent particles.

4.1.5. Vocabulary Modules

The goal of the Vocabulary Module is to generate the vocabulary of the language. The module configures the `VocabDictionary` class, which holds the list of `VocabularyEntry` objects. Each `VocabularyEntry` object contains the word in the constructed language, its translation, and its definition in english. A source words list is also part of each entry, to facilitate easy search for translation purposes. The class can also hold embeddings for each word, which could be also used for downstream tasks. The vocabulary building is functionally split into two modules: Generation modules which generate the vocabulary, and Mapping modules which map the concepts with specific words. For some setups, this is combined into a single module.

FromSourceVocabularyModule

This is a baseline generation module that takes the `n` most common words in english, and adds any missing words from the source text, and adds them to the vocabulary.

ClusterTwoLevelVocabularyModule

This is a combined vocabulary module that clusters the vocabulary into `n` clusters, based on embeddings. Each cluster is then assigned a root syllable, and each word in the cluster is assigned a leaf syllable. The final word is generated by concatenating the root syllable and the leaf syllable. Agglomerative Clustering is used to cluster the vocabulary, and the number of clusters is passed as an argument to the module.

ClusterAndSimplifyVocabularyModule

This module behaves similarly to the `ClusterTwoLevelVocabularyModule`, but instead of assigning every word, each cluster is assigned a single word.

ApproximatingVocabularyModule

This module takes a vocabulary size as argument, and generates a vocabulary of that size based on the most common words in english. Other words required for the translation are then approximated to the closest word in the vocabulary, again based on the distance norm of the embeddings.

RandomMappingModule

This is a mapping module that randomly assigns a word to each concept in the vocabulary.

4.1.6. Translation Modules

Once the language description is generated, we use a translation module to translate some text corpora represented by `AbstractSourceText`. The module translates the source text paragraph by paragraph, using an LLM model.

4.2. Evaluation Modules

Figure 4.2 shows the setup for running evaluations on the generated languages. The Evaluators are a separate class that does not inherit from the abstract `Module` class. They inherit the `Evaluator` class, which creates an evaluations folder inside the results folder, to store the results of the evaluations.

4.2.1. GenerateTranslationSummary

This evaluator generates an HTML summary of the language description, and the generated translation. It also counts the vocabulary size and the number of syllables in the conlang.

4.2.2. DetranslationEvaluator

This evaluator takes the generated translation and translates it back to english, using the same LLM model. It then compares the original text with the translated text, and generates translation metrics like BLEU, METEOR, and ROUGE. It also generates an HTML summary of the detranslation.

4.2.3. CompressionEvaluator

This evaluator compresses the generated translation and source text using a compression algorithm, and generates the compression ratio for both.

4.2.4. RaceCEvaluator

This evaluator evaluates the performance on the Race-C Dataset, using the original text and the detranslated text. It generates the accuracy scores for the evaluation.

4.2.5. BertEvaluator

This evaluator fine-tunes a BERT model and records the initial and final perplexity, along with the eval loss and loss reduction rate.

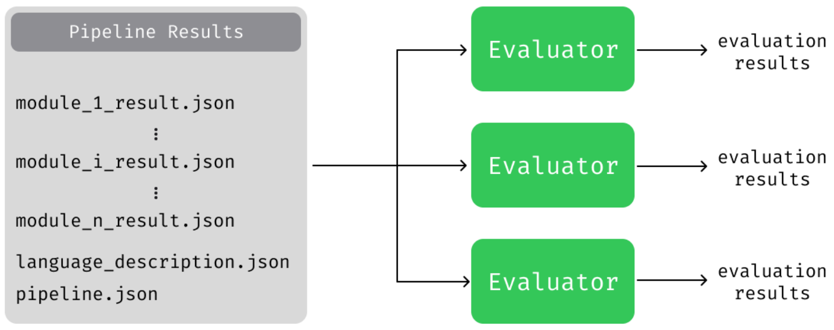


Figure 4.2.: The setup for running evaluations. Each evaluator stores the results in a folder inside the evaluations folder.

4.2.6. ZipfsLawEvaluator

This evaluator generates a Zipf's Law plot for the generated translation and the source text, and records the Zipf's Exponent for both.

4.3. Pipeline Implementations

4.3.1. Baseline Pipeline

The baseline pipeline is the most straightforward pipeline, which we use to compare with different setups. Figure 4.3 shows the structure of the baseline pipeline. We used the most common number of phonemes in the world languages, as well as the most common phonotactic rules. It uses a simple Agglutinative grammar, and ensures all words from the source text is included in the vocabulary.

4.3.2. Approximation Pipeline

The approximation pipeline uses an approximation vocabulary module, which create a fixed vocabulary based on the most common words in english, and then approximates the rest of the words to the closest word in the vocabulary. Everything else is the same as the baseline pipeline.

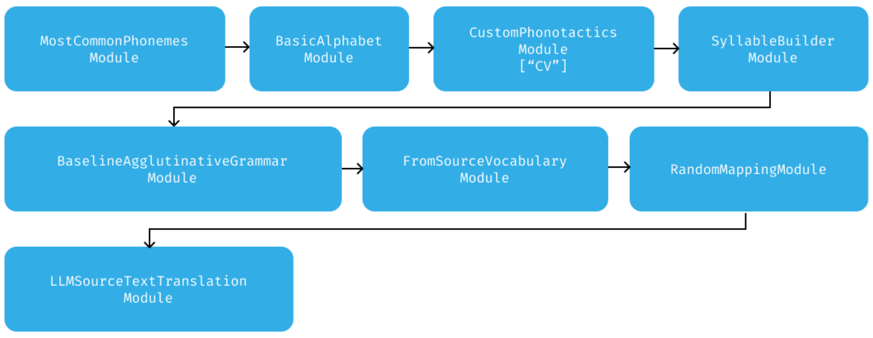

MostCommonPhonemes Module
BasicAlphabet Module
CustomPhonotactics Module ["CV"]
SyllableBuilder Module
BaselineAgglutinativeGrammar Module
FromSourceVocabulary Module
RandomMappingModule
LLMSourceTextTranslation Module

Figure 4.3.: The baseline pipeline for generating constructed languages.

4.3.3. Two Level Pipeline

The two level pipeline uses a two level vocabulary module, which clusters the vocabulary into `n` clusters, based on embeddings. Each word is bisyllabic, and the first syllable is the root syllable based on the cluster, and the second syllable is the leaf syllable.

4.3.4. Cluster and Simplify Pipeline

The cluster and simplify pipeline uses the cluster and simplify vocabulary module, which clusters the vocabulary into `n` clusters, based on embeddings. Each cluster is assigned a single word, and the rest of the words are approximated to the closest word in the cluster.

4.3.5. Phonotactics Pipeline

The phonotactics pipeline uses the `CustomPhonotacticsModule`. As compared to the baseline pipeline which uses the `CV` syllable structure, This pipeline uses custom phonotactic rules. The rest of the pipeline is the same as the baseline pipeline.

4.4. LLMs and Embedders

We used both Local and Remote LLMs for our experiments. The use of Local LLMs is facilitated by the `LocalLLM` class, which is a wrapper to access local LLMs via the Ollama Library. The Remote LLMs are accessed via the `RemoteLLM` class, which supports two providers: OpenAI and Groq. The primary models used for the experiments were *deepseek-r1:14b* and OpenAI's *gpt-4o*.

For the embedding models, we primarily used the *all-MiniLM-L6-v2* model from HuggingFace.

4.5. Clustering

Wherever clustering is required, it is done with subclasses of the `ClusteringMethod` class which implement the `get_clusters` method. We used the *AgglomerativeClustering* class from the *sklearn* library for clustering the vocabulary, which we noticed gave the best results heuristically. The codebase also has implementations for *KMeans*, *DBScan*, *HDBScan*, etc.

4.6. Other tools

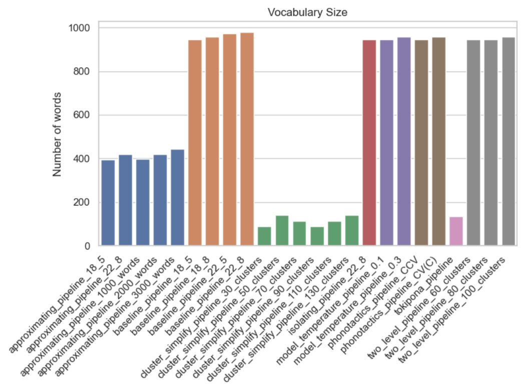
The source texts used for the experiments are accessed via the `AbstractSourceText` class, which is a wrapper for the source text. We provide multiple implementations of this class, and in the final experiments we used the `RaceC` class, which uses the articles from the Race-C dataset. Once translated, the translation pairs can be accessed via the `AbstractTranslationPairs` class.

For obtaining the most common words in english, we used the *wordfreq* python library [Spe25]. This library provides frequency lists for the most common words in over 40 languages, and is based on the Exquisite Corpus [25].

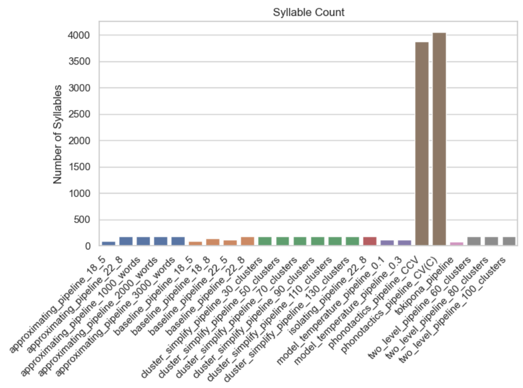
5. Results

This chapter presents the results of the experiments conducted to evaluate the performance of the different setups. The complete table of results is shown in Table A.1.

5.1. Vocabulary Statistics



(a) Vocabulary Size

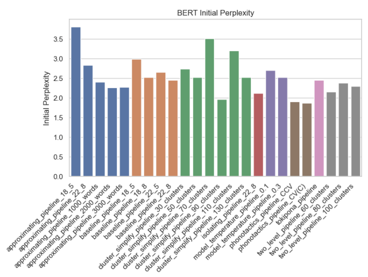


(b) Syllable Count

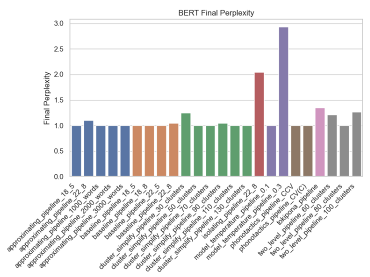
Figure 5.1.: Vocabulary Statistics.

Figure 5.1 shows the vocabulary size and syllable count for each setup. The vocabulary size is the number of unique words generated, while the syllable count is the total number of possible syllables in the language. The approximating and cluster simplify strategies have the smallest vocabulary size, along with Toki Pona. Syllable counts are based on the total number of phonemes and the phonotactic rules, and so the phonotactics pipeline has the highest syllable count by far.

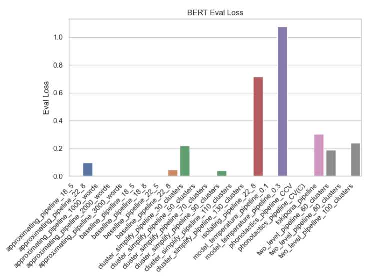
5.2. Bert Evaluation



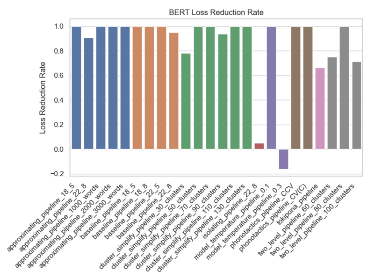
(a) BERT Initial Perplexity



(b) BERT Final Perplexity



(c) BERT Eval Loss

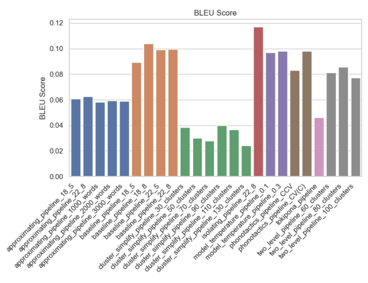


(d) BERT Loss Reduction Rate

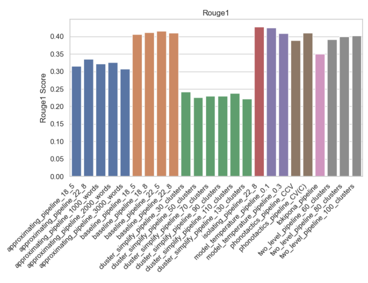
Figure 5.2.: BERT Evaluation

Figure 5.2 shows the BERT evaluation results. The initial perplexity is the perplexity of the model before training, i.e. how confused the model was about predicting masked words, before the fine tuning process. The final perplexity is the perplexity of the model after training, i.e. how confused the model was about predicting masked words, after the fine tuning process. The Eval Loss is the final evaluation loss from the model after training, and the loss reduction rate indicates how much the loss was reduced during training.

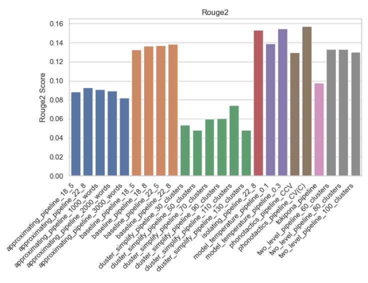
5.3. Translation Scores



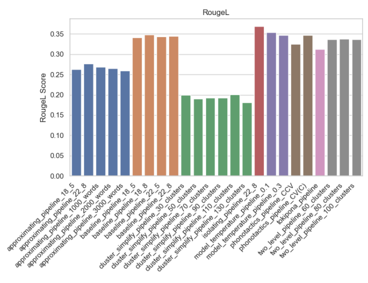
(a) BLEU Scores



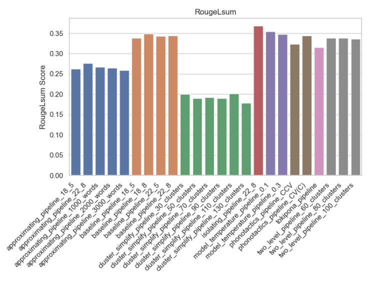
(b) Rouge 1 Scores



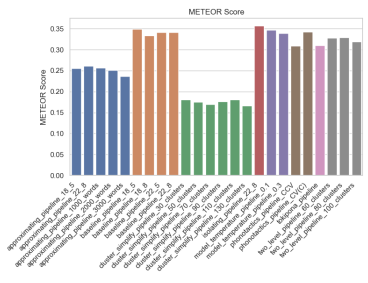
(c) Rouge 2 Scores



(d) Rouge L Scores



(e) Rouge L Sum Scores



(f) Meteor Scores

Figure 5.3.: Translation Scores

Figure 5.3 shows the translation scores for each setup. Once we translate and then detranslate the source text to and from the constructed language, we use the well known BLEU, ROUGE, and METEOR metrics to evaluate the quality of the translation. Higher scores indicate better translation quality. As we can see, the approximating and cluster simplify strategies have the lowest scores, which makes sense considering there would be some amount of information loss in the process of simplifying.

5.4. Race-C Evaluation

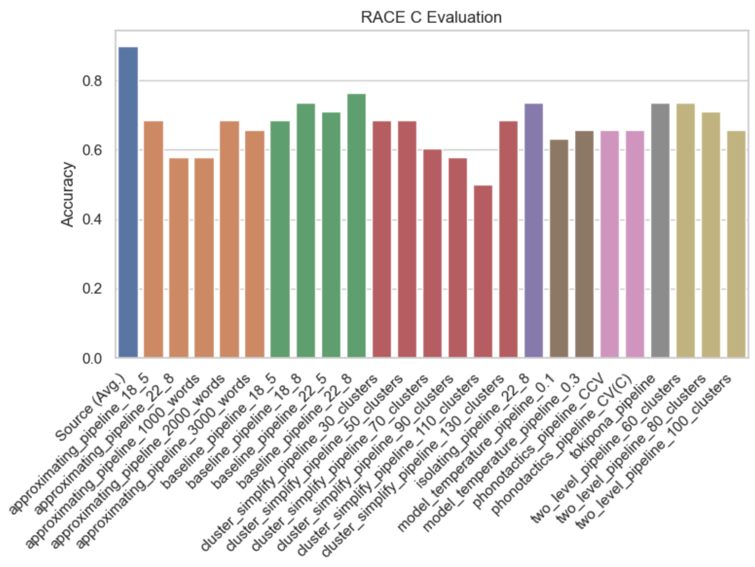


Figure 5.4.: Race-C Scores.

Figure 5.4 shows how well the model was able to answer the questions from the race-c evaluation, given the source text, or the detranslated text. As expected, the original text has the highest score. Interestingly, the simplified and approximating pipelines do not perform much less than the other pipelines compared to the sharp drop seen in the translation scores.

5.5. Zipf's Law Evaluation

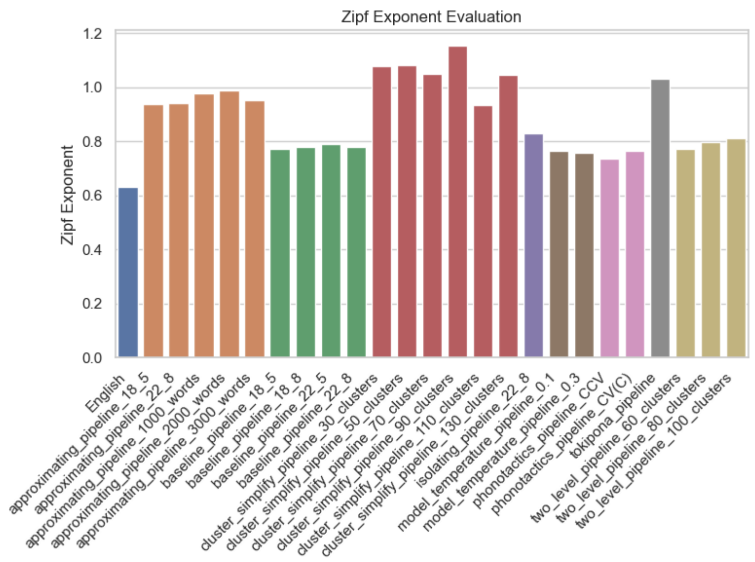


Figure 5.5.: Zipf's Exponent

Figure 5.5 shows the Zipf's exponent for each setup. The exponent is a measure of how well the word frequency distribution follows Zipf's law, which states that the frequency of a word is inversely proportional to its rank in the frequency table. A value of 1 indicates a perfect fit to Zipf's law, while a value of 0 indicates no fit at all. In our case, since the source text we use is small, we expect some deviation from the law. However, it can still be useful to compare different setups. In this case, we can note that the approximating and cluster simplify strategies get the closest to the law, along with Toki Pona. This makes sense, since the chance of rare words appearing in the text is lower, and so the distribution is more uniform. It also explains why English has such a low exponent, since in such a small text, the chance of rare words appearing is much higher.

6. Discussion, Conclusion and Future Work

In this chapter, we discuss the results of the experiments conducted in Chapter 5. We will discuss the effect of different parameters on the performance of the models, and how they relate to the original goals of this thesis. We then conclude with a summary of the findings and suggest future work that can be done.

6.1. Effect of Phoneme Count

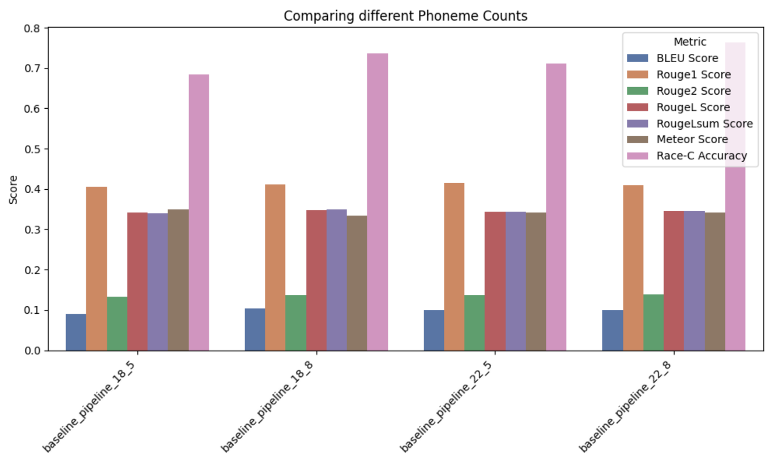


Figure 6.1.: Comparing the Effects of different phoneme counts

As we can see in Figure 6.1, the phoneme count does not seem to have a significant effect on the translation scores or the Race-C scores. This being the case, we can conclude that we can simplify a language by reducing the phoneme count without affecting the ability of the language to convey meaning.

6.2. Effect of Phonotactics

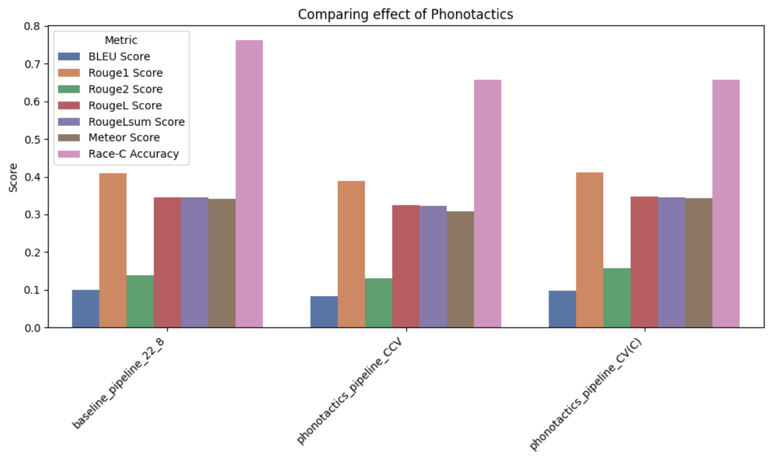


Figure 6.2.: Comparing the Effects of phonotactics

Figure 6.2 shows the effect of phonotactics on the translation scores and the Race-C scores. Again, we can see that the phonotactics do not seem to have a significant effect either of the metrics. We can therefore conclude that we can simplify a language by using simplifed phonotactic rules without affecting the ability of the language to convey meaning.

6.3. Effect of Grammar Rules

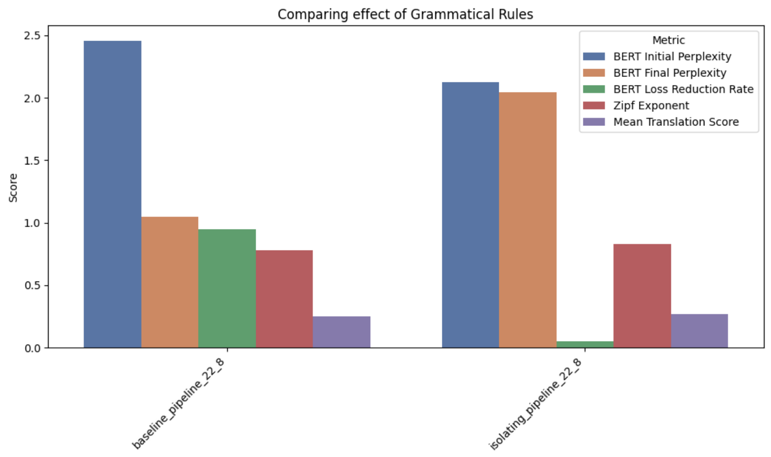
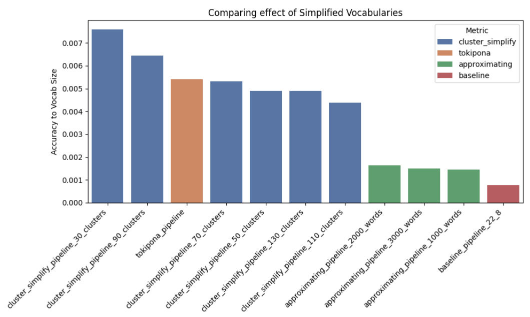


Figure 6.3.: Comparing different Grammatical Rules

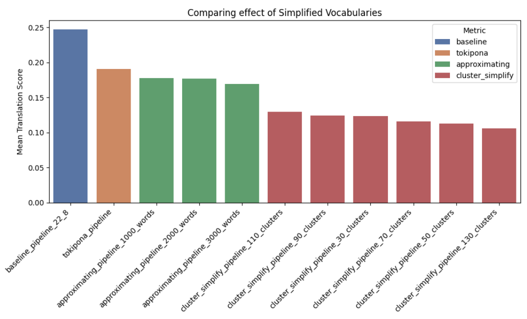
We implemented 2 different grammars, but from our results we did not find any significant difference between the Agglutinative and Isolating grammar modules, save for differences in the Final Perplexity in the Bert Evaluation. Further investigation is required to analyse the effects of grammatical features in constructed languages.

6.4. Effect of Vocabulary Generation method

Figure A.2 shows the effect of different simplifying vocabulary generation methods against our metrics. We can see that both methods result in a drop in the translation scores, and in the Race-C scores. We can consider the ratio of the Mean Translation Score (MTS) and the Race-C Accuracy to the vocabulary size.



(a) Effect of Vocabulary on Race-C Accuracy



(b) Effect of Vocabulary on MTS

Figure 6.4.: Effect of Simplified Vocabulary.

From Figure 6.4b, we can see that the drop in MTS Score is much more significant, with the baseline having the highest ratio. However, as we can see from Figure 6.4a, The cluster-simplify and approximating methods perform better in terms of accuracy to vocabulary size. For all practical purposes, we can argue that the performance on the Reading Comprehension dataset is a better proxy for the performance of the language, rather than the MTS scores, which heavily depend on n-gram similarity.

6.5. Effect of Language Model Temperature

We compared the effects of different model temperatures on the evaluations, with the same baseline pipeline. The baseline was run with a temperature value of `0.2`. From Figure 6.5, we can see that the higher model temperature seems to degrade the result of some scores, but does not seem to have a significant effect with a lower temperature. We can conclude that the results can be reproduced at different model temperatures, as long as the temperature is not too high.

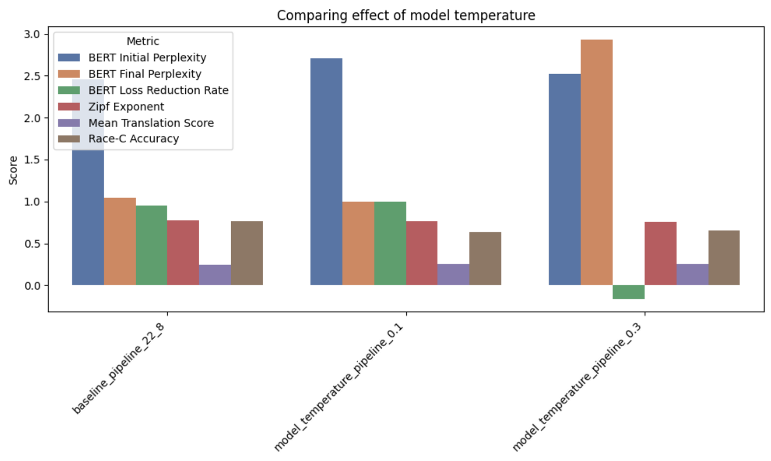


Figure 6.5.: Effect of Model Temperature

6.6. Conclusion

This thesis set out to investigate whether pre-trained Large Language Models (LLMs) can aid in the systematic construction of minimal, efficient languages that preserve communicative utility. By developing a modular pipeline that integrates phonology, phonotactics, grammar, vocabulary, and multiple evaluation strategies, we were able to design and assess a variety of constructed languages (ConLangs) under different constraints.

The results demonstrate that significant simplifications are possible in multiple linguistic subsystems, such as reducing phoneme inventory and grammar complexity, without substantially degrading performance on machine translation or comprehension tasks. In particular, reading comprehension scores remained relatively robust even in highly simplified setups, indicating that core meaning can be preserved despite a reduction in language complexity. Additionally, the adherence to Zipf's Law in many generated languages suggests that even artificial languages can naturally align with linguistic patterns found in human languages.

Our experiments show that LLMs can effectively support language generation and evaluation, acting as tools to explore the trade-off between efficiency and expressiveness in language design. The modularity of our framework allows for targeted ablation studies, making it a flexible foundation for future research in computational linguistics and language optimization.

In essence, this work provides strong evidence that LLMs can assist in creating interpretable, compact, and efficient constructed languages, opening new avenues for both linguistic theory and applied language technology.

6.7. Future Work

The results of this thesis show that it is possible to generate simplified languages that are still able to convey meaning. However, there are still many open areas for future work. Firstly, we can explore other different language generation methods, including more or less complex grammatical rules, or different vocabulary generation methods. In particular, further research is required in analyzing the effects of different grammatical structures.

In addition, further research is required on metrics to evaluate the generated languages. In particular, there is more to explore in terms of simplicity measures, which are complex to define and measure.

7. Author Contributions

The codebase used for this thesis was developed by me in collaboration with Sara Visconti. I setup the initial repository and abstract classes for the modules and pipelines as well as the data classes for the language, which was then reviewed and refactored by Sara. Both of us modified or added to these modules and classes as needed based on experiment requirements. We also implemented specfiic modules for the experiments.

1. **Phonemes**: I implemented the RandomPhonemeModule, MostCommonPhonemeModule and LanguagePhonemeModule. I also implemented the BasicAlphabetModule.
2. **Phonotactics**: I implemented the BasicPhonotacticsModule, CustomPhonotacticsModule and SyllableBuilderModule.
3. **Grammar**: I implemented the BasicGrammarModule, BaselineAgglutinativeGrammarModule, and BaselineIsolatingGrammarModule. These modules built on top of Sara's Implementations of grammar features.
4. **Vocabulary**: I implemented the FixedVocabularyModule, FromSourceVocabularyModule, ClusterSimplifyVocabularyModule, ClusterTwoLevelVocabularyModule and ApproximatingVocabularyModule. I also used the Mapping modules implemented by Sara.
5. **Conlang**: I implemented the LLM based Source Text Translation Module.
6. **Evaluation**: I implemented the CompressionEvaluator, DetranslationEvaluator, HTML Summary Generator and RaceCEvaluator. I used Sara's Implementation of Zipfs Law Evaluator and BertEvaluator. Sara also added the various metrics to the DetranslationEvaluator.
7. **LLMs**: I implemented the LocalLLM and OpenAI implementation of the Remote LLM.

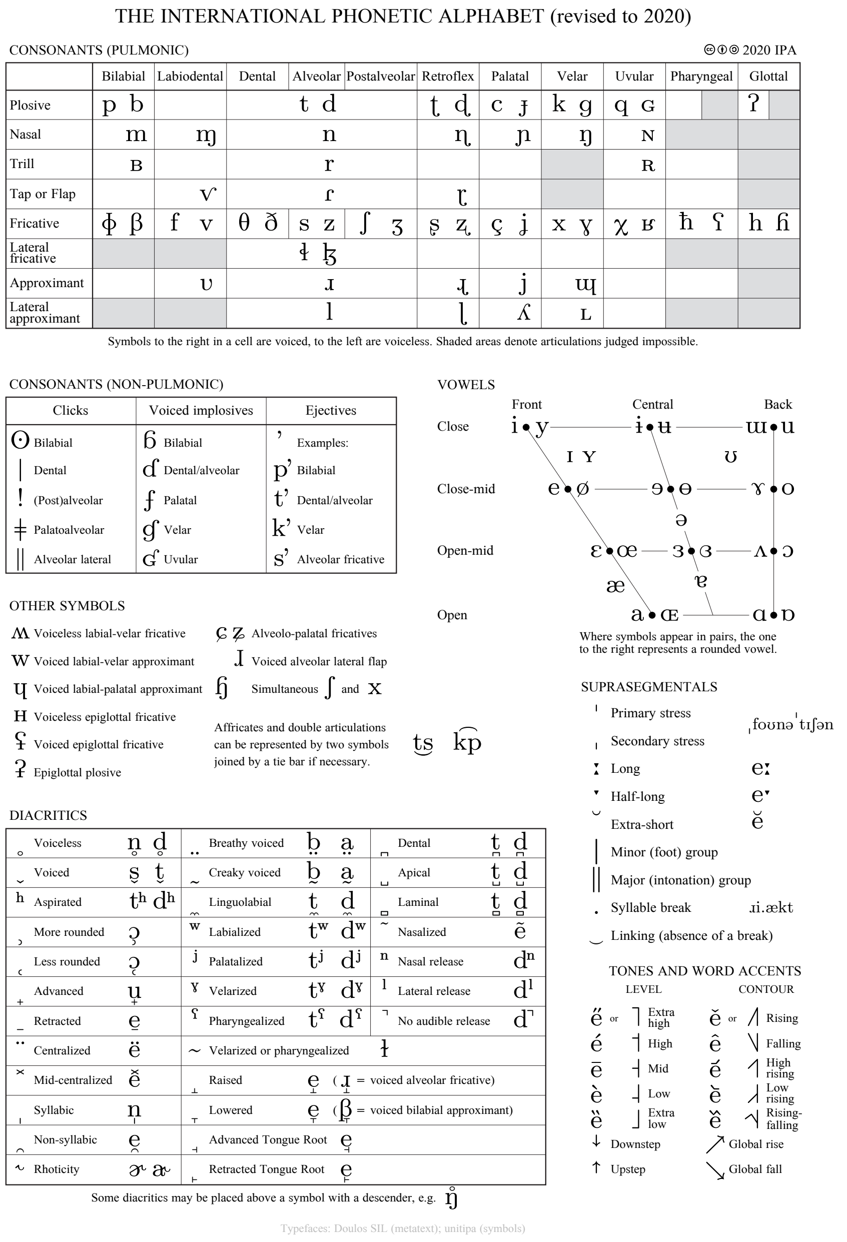
We jointly developed the conceptual design and methodology. Other contributions from Sara Visconti are not reported here since they are not mentioned in this work.

A. Appendix

A.1. Extended Results

Label	VS	Syll. Count	BIP	BFP	EL	LRR	CR	BLEU	R1	R2	RL	RLsum	Meteor	RC	Zipf	RC/VS	MTS	MTS/VS
AP 18_5	395	90	3.805036	1.000701	0.000701	0.999475	0.382916	0.060660	0.315455	0.088494	0.263348	0.262436	0.255022	0.684211	0.939288	0.001732	0.176122	0.000446
AP 22_8	418	176	2.828593	1.101228	0.096426	0.907263	0.361459	0.062460	0.335964	0.092946	0.277229	0.275981	0.261639	0.578947	0.942390	0.001385	0.184187	0.000441
AP 1000	398	176	2.401913	1.000247	0.000247	0.999718	0.345869	0.058229	0.322852	0.091146	0.268410	0.267654	0.257057	0.578947	0.977553	0.001455	0.177847	0.000447
AP 2000	420	176	2.267986	1.000324	0.000324	0.999604	0.338485	0.059252	0.325953	0.089243	0.264856	0.264440	0.251265	0.684211	0.989893	0.001629	0.176597	0.000420
AP 3000	443	176	2.279193	1.000346	0.000346	0.999580	0.341297	0.058849	0.308112	0.082064	0.259167	0.259282	0.236497	0.657895	0.951189	0.001485	0.169419	0.000382
BP 18_5	944	90	2.984567	1.003452	0.003446	0.996848	0.413533	0.089393	0.406293	0.132522	0.341110	0.338671	0.348798	0.684211	0.771769	0.000725	0.240591	0.000255
BP 18_8	959	144	2.524507	1.000322	0.000322	0.999652	0.382578	0.103775	0.412069	0.136566	0.348190	0.349100	0.334057	0.736842	0.778133	0.000768	0.248789	0.000259
BP 22_5	974	110	2.651095	1.000346	0.000346	0.999645	0.413514	0.099102	0.415278	0.137157	0.343517	0.343490	0.341473	0.710526	0.791547	0.000729	0.247064	0.000254
BP 22_8	978	176	2.455257	1.047953	0.046839	0.947855	0.374543	0.099540	0.409906	0.138335	0.344875	0.344601	0.341364	0.763158	0.778184	0.000780	0.247344	0.000253
CSP 30	90	176	2.735671	1.245957	0.219904	0.781490	0.314786	0.038296	0.242180	0.053529	0.200019	0.199877	0.181215	0.684211	1.078153	0.007602	0.123780	0.001375
CSP 50	140	176	2.527449	1.000060	0.000060	0.999935	0.320828	0.030046	0.225745	0.048193	0.189754	0.189834	0.174738	0.684211	1.082406	0.004887	0.112778	0.000806
CSP 70	114	176	3.507812	1.000395	0.000395	0.999685	0.299885	0.027665	0.229800	0.060093	0.192322	0.191943	0.170063	0.605263	1.049119	0.005309	0.115696	0.001015
CSP 90	90	176	1.963381	1.043078	0.042176	0.937486	0.305162	0.039848	0.230607	0.060676	0.192008	0.190578	0.176884	0.578947	1.152982	0.006433	0.123850	0.001376
CSP 110	114	176	3.205752	1.000393	0.000393	0.999662	0.318047	0.036439	0.237634	0.073999	0.201005	0.200997	0.181172	0.500000	0.935489	0.004386	0.129379	0.001135
CSP 130	140	176	2.529813	1.000399	0.000399	0.999570	0.309554	0.024032	0.222267	0.047936	0.181579	0.178963	0.166774	0.684211	1.047524	0.004887	0.105612	0.000754
IP 22_8	944	176	2.123571	2.046694	0.716226	0.048962	0.383174	0.116954	0.427392	0.153538	0.368515	0.368095	0.356711	0.736842	0.828702	0.000781	0.268101	0.000284
MTP 0.1	944	110	2.704528	1.000228	0.000228	0.999770	0.413414	0.096900	0.425194	0.139300	0.354264	0.354305	0.347425	0.631579	0.766435	0.000669	0.251037	0.000266
MTP 0.3	959	110	2.522915	2.931649	1.075565	-0.162252	0.413831	0.098095	0.408808	0.154705	0.347259	0.348118	0.338973	0.657895	0.759286	0.000686	0.251692	0.000262
PTP CCV	944	3872	1.901847	1.000213	0.000213	0.999668	0.379260	0.083152	0.388853	0.129627	0.324554	0.323035	0.308993	0.657895	0.735222	0.000697	0.226711	0.000240
PTP CV(C)	959	4048	1.874292	1.000236	0.000236	0.999625	0.362352	0.098214	0.410678	0.157076	0.347063	0.344817	0.342890	0.657895	0.765674	0.000686	0.252633	0.000263
TKP	136	82	2.454759	1.351960	0.301555	0.664203	0.319333	0.046147	0.349945	0.097848	0.312278	0.315641	0.310078	0.736842	1.030834	0.005418	0.190831	0.001403
TLP 60	944	176	2.153737	1.209560	0.190257	0.752013	0.375062	0.081251	0.391477	0.133255	0.336282	0.338883	0.328016	0.736842	0.770698	0.000781	0.232618	0.000246
TLP 80	944	176	2.378547	1.000316	0.000316	0.999636	0.370774	0.085548	0.400041	0.133011	0.337966	0.338431	0.328805	0.710526	0.797924	0.000753	0.235639	0.000250
TLP 100	959	176	2.294700	1.270623	0.239508	0.711646	0.374313	0.077206	0.403058	0.130378	0.336102	0.335990	0.318717	0.657895	0.811762	0.000686	0.229476	0.000239

Table A.1.: Summary of the Results



THE INTERNATIONAL PHONETIC ALPHABET (revised to 2020)

CONSONANTS (PULMONIC) 2020 IPA

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k ɡ	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Symbols to the right in a cell are voiced, to the left are voiceless. Shaded areas denote articulations judged impossible.

CONSONANTS (NON-PULMONIC)

Clicks		Voiced implosives		Ejectives	
ʘ	Bilabial	ɓ	Bilabial	ʼ	Examples:
ǀ	Dental	ɗ	Dental/alveolar	pʼ	Bilabial
ǃ	(Post)alveolar	ʄ	Palatal	tʼ	Dental/alveolar
ǂ	Palatoalveolar	ɠ	Velar	kʼ	Velar
ǁ	Alveolar lateral	ʛ	Uvular	sʼ	Alveolar fricative

VOWELS

	Front		Central		Back
Close	i • y		ɨ • ʉ		ɯ • u
		ɪ ʏ		ʊ	
Close-mid	e • ø		ɘ • ɵ		ɤ • o
			ə		
Open-mid	ɛ • œ		ɜ • ɞ		ʌ • ɔ
		æ	ɐ		
Open	a • ɶ				ɑ • ɒ

Where symbols appear in pairs, the one to the right represents a rounded vowel.

OTHER SYMBOLS

ʍ Voiceless labial-velar fricative	ɕ ʑ Alveolo-palatal fricatives
w Voiced labial-velar approximant	ɺ Voiced alveolar lateral flap
ɥ Voiced labial-palatal approximant	ɧ Simultaneous ʃ and x
ʜ Voiceless epiglottal fricative	
ʢ Voiced epiglottal fricative	Affricates and double articulations can be represented by two symbols joined by a tie bar if necessary. t͡s k͡p
ʡ Epiglottal plosive	

DIACRITICS

̥	Voiceless	n̥ d̥	̤	Breathy voiced	b̤ a̤	̪	Dental	t̪ d̪
̬	Voiced	s̬ t̬	̰	Creaky voiced	b̰ a̰	̺	Apical	t̺ d̺
ʰ	Aspirated	tʰ dʰ	̼	Linguolabial	t̼ d̼	̻	Laminal	t̻ d̻
̹	More rounded	ɔ̹	ʷ	Labialized	tʷ dʷ	̃	Nasalized	ẽ
̜	Less rounded	ɔ̜	ʲ	Palatalized	tʲ dʲ	ⁿ	Nasal release	dⁿ
̟	Advanced	u̟	ˠ	Velarized	tˠ dˠ	ˡ	Lateral release	dˡ
̠	Retracted	e̠	ˤ	Pharyngealized	tˤ dˤ	̚	No audible release	d̚
̈	Centralized	ë	̴	Velarized or pharyngealized	ɫ			
̽	Mid-centralized	e̽	̝	Raised	e̝ (ɹ̝ = voiced alveolar fricative)			
̩	Syllabic	n̩	̞	Lowered	e̞ (β̞ = voiced bilabial approximant)			
̯	Non-syllabic	e̯	̘	Advanced Tongue Root	e̘			
˞	Rhoticity	ɚ a˞	̙	Retracted Tongue Root	e̙			

Some diacritics may be placed above a symbol with a descender, e.g. ŋ̊

SUPRASEGMENTALS

ˈ	Primary stress	ˌfoʊnəˈtɪʃən
ˌ	Secondary stress	
ː	Long	eː
ˑ	Half-long	eˑ
̆	Extra-short	ĕ
|	Minor (foot) group	
‖	Major (intonation) group	
.	Syllable break	ɹi.ækt
‿	Linking (absence of a break)	

TONES AND WORD ACCENTS

LEVEL			CONTOUR		
e̋ or	˥	Extra high	ě or	˩˥	Rising
é	˦	High	ê	˥˩	Falling
ē	˧	Mid	e᷄	˦˥	High rising
è	˨	Low	e᷅	˩˨	Low rising
ȅ	˩	Extra low	e᷈	˧˦˧	Rising-falling
ꜜ	Downstep		↗	Global rise	
ꜛ	Upstep		↘	Global fall	

Typefaces: Doulos SIL (metatext); unitipa (symbols)

Figure A.1.: IPA Chart, available under a Creative Commons Attribution-Sharealike 3.0 Unported License. Copyright © 2018 International Phonetic Association.

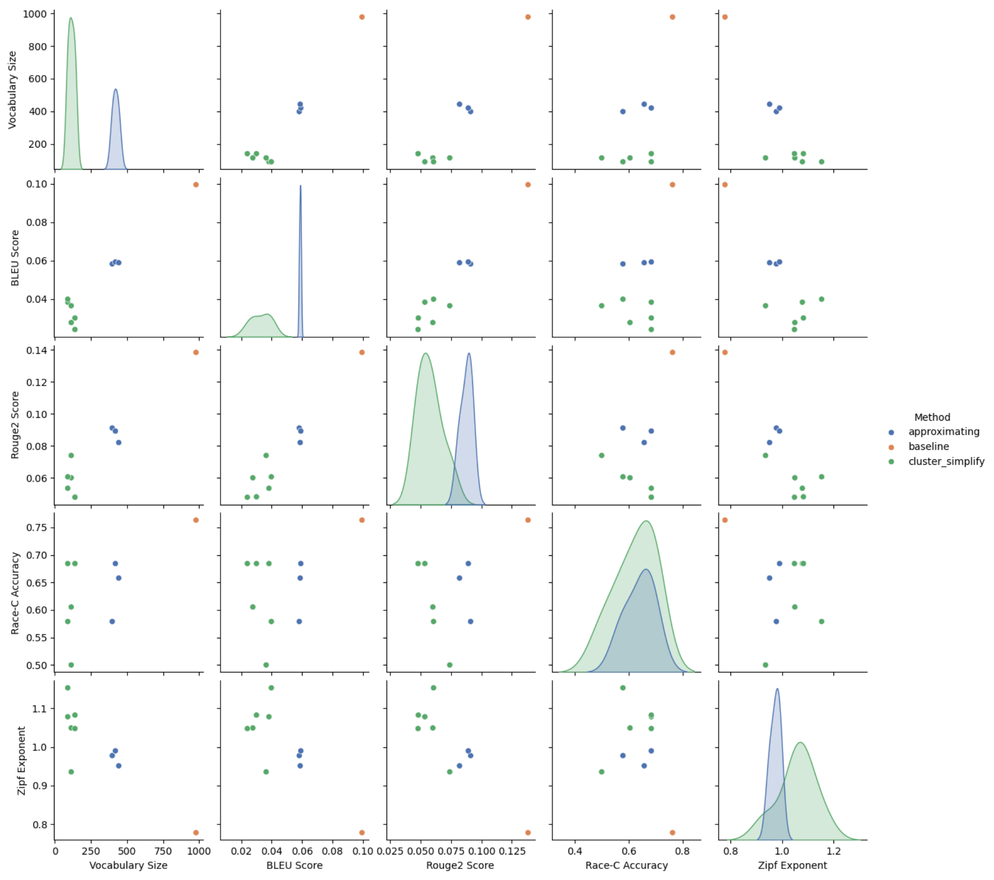


Figure A.2.: Comparing different Simplifying vocabulary generation methods with the baseline.

Abbreviations

List of Figures

List of Tables

Bibliography

[24] “Ithkuil.” In: *Wikipedia* (Aug. 2024).

[25] *LuminosoInsight/Exquisite-Corpus*. Luminoso Technologies, Inc. Jan. 2025.

[99] *Handbook of the International Phonetic Association : A Guide to the Use of the International Phonetic Alphabet*. 1. publ. Cambridge u.a.: Cambridge Univ. Press, 1999.

[BCV14] Y. Bengio, A. Courville, and P. Vincent. *Representation Learning: A Review and New Perspectives*. Apr. 2014. DOI: `10.48550/arXiv.1206.5538`. arXiv: `1206.5538 [cs]`.

[BL05] S. Banerjee and A. Lavie. “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments.” In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ed. by J. Goldstein, A. Lavie, C.-Y. Lin, and C. Voss. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 65–72.

[Bro+20] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. *Language Models Are Few-Shot Learners*. July 2020. DOI: `10.48550/arXiv.2005.14165`. arXiv: `2005.14165 [cs]`.

[CMS13] R. J. G. B. Campello, D. Moulavi, and J. Sander. “Density-Based Clustering Based on Hierarchical Density Estimates.” In: *Advances in Knowledge Discovery and Data Mining*. Ed. by J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu. Berlin, Heidelberg: Springer, 2013, pp. 160–172. ISBN: 978-3-642-37456-2. DOI: `10.1007/978-3-642-37456-2_14`.

[DB20] Ł. Dębowski and C. Bentz. “Information Theory and Language.” In: *Entropy (Basel, Switzerland)* 22.4 (Apr. 2020). ISSN: 1099-4300. DOI: `10.3390/e22040435`.

[Dee+25] DeepSeek-AI, D. Guo, D. Yang, et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. https://arxiv.org/abs/2501.12948v1. Jan. 2025.

[Dev+19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by J. Burstein, C. Doran, and T. Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: `10.18653/v1/N19-1423`.

[Est+96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, Aug. 1996, pp. 226–231.

[FH22] R. Futrell and M. Hahn. "Information Theory as a Bridge between Language Function and Language Form." In: *Frontiers in Communication* 7 (2022). ISSN: 2297-900X. DOI: `10.3389/fcomm.2022.657725`.

[Gib+19] E. Gibson, R. Futrell, S. P. Piantadosi, I. Dautriche, K. Mahowald, L. Bergen, and R. Levy. "How Efficiency Shapes Human Language." In: *Trends in Cognitive Sciences* 23.5 (May 2019), pp. 389–407. ISSN: 1364-6613. DOI: `10.1016/j.tics.2019.02.003`.

[Ha10] R. Ha. "Cost-Benefit Analysis in Animal Behavior." In: Jan. 2010, pp. 402–405.

[Har54] Z. S. Harris. "Distributional Structure." In: *WORD* (Aug. 1954). ISSN: 0043-7956.

[HS97] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory." In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`.

[Jai10] A. K. Jain. "Data Clustering: 50 Years beyond K-means." In: *Pattern Recognition Letters*. Award Winning Papers from the 19th International Conference on Pattern Recognition (ICPR) 31.8 (June 2010), pp. 651–666. ISSN: 0167-8655. DOI: `10.1016/j.patrec.2009.09.011`.

[JM25] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. 2025.

[Kan+17] J. Kanwal, K. Smith, J. Culbertson, and S. Kirby. “Zipf’s Law of Abbreviation and the Principle of Least Effort: Language Users Optimise a Miniature Lexicon for Efficient Communication.” In: *Cognition* 165 (Aug. 2017), pp. 45–52. ISSN: 1873-7838. DOI: `10.1016/j.cognition.2017.05.001`.

[Lai+17] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. “RACE: Large-scale ReAding Comprehension Dataset From Examinations.” In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Ed. by M. Palmer, R. Hwa, and S. Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 785–794. DOI: `10.18653/v1/D17-1082`.

[Lev22] N. Levshina. *Communicative Efficiency: Language Structure and Use*. Cambridge: Cambridge University Press, 2022.

[Lin04] C.-Y. Lin. “ROUGE: A Package for Automatic Evaluation of Summaries.” In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81.

[Llo82] S. Lloyd. “Least Squares Quantization in PCM.” In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pp. 129–137. ISSN: 1557-9654. DOI: `10.1109/TIT.1982.1056489`.

[LLY19] Y. Liang, J. Li, and J. Yin. “A New Multi-choice Reading Comprehension Dataset for Curriculum Learning.” In: *Proceedings of The Eleventh Asian Conference on Machine Learning*. PMLR, Oct. 2019, pp. 742–757.

[Mik+13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed Representations of Words and Phrases and Their Compositionality.” In: *Advances in Neural Information Processing Systems*. Vol. 26. Curran Associates, Inc., 2013.

[Mil94] G. A. Miller. “WordNet: A Lexical Database for English.” In: *Human Language Technology: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 8-11, 1994*. 1994.

[MM19] S. Moran and D. McCloy, eds. *Phoible 2.0*. Jena: Max Planck Institute for the Science of Human History, 2019.

[Pap+02] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. “BLEU: A Method for Automatic Evaluation of Machine Translation.” In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL ’02. USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: `10.3115/1073083.1073135`.

[Pet+18] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. "Deep Contextualized Word Representations." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Ed. by M. Walker, H. Ji, and A. Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: `10.18653/v1/N18-1202`.

[Pet15] D. J. Peterson. *The Art of Language Invention : From Horse-Lords to Dark Elves, the Words behind World-Building*. New York, New York: Penguin Books, 2015. Chap. ix, 292 pages ; 22 cm. ISBN: 978-0-14-312646-1 0-14-312646-6.

[Pia14] S. T. Piantadosi. "Zipf's Word Frequency Law in Natural Language: A Critical Review and Future Directions." In: *Psychonomic Bulletin & Review* 21.5 (Oct. 2014), pp. 1112–1130. ISSN: 1531-5320. DOI: `10.3758/s13423-014-0585-6`.

[PSM14] J. Pennington, R. Socher, and C. Manning. "GloVe: Global Vectors for Word Representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by A. Moschitti, B. Pang, and W. Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: `10.3115/v1/D14-1162`.

[RKR20] A. Rogers, O. Kovaleva, and A. Rumshisky. *A Primer in BERTology: What We Know about How BERT Works*. Nov. 2020. DOI: `10.48550/arXiv.2002.12327`. arXiv: `2002.12327 [cs]`.

[Ros10] M. Rosenfelder. *The Language Construction Kit*. Yonagu Books, 2010. ISBN: 978-0-9844700-0-6.

[Sch21] C. Schreyer. "Constructed Languages." In: *Annual Review of Anthropology* 50.Volume 50, 2021 (2021), pp. 327–344. ISSN: 1545-4290. DOI: `10.1146/annurev-anthro-101819-110152`.

[Sha51] C. E. Shannon. "Prediction and Entropy of Printed English." In: *The Bell System Technical Journal* 30.1 (Jan. 1951), pp. 50–64. ISSN: 0005-8580. DOI: `10.1002/j.1538-7305.1951.tb01366.x`.

[Spe25] E. R. L. (Speer). *Rspeer/Wordfreq*. Apr. 2025.

[TS07] R. Trask and P. Stockwell. *Language and Linguistics: The Key Concepts*. Key Concepts Series. Routledge, 2007. ISBN: 978-0-415-41359-6.

[Vas+23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. Aug. 2023. DOI: `10.48550/arXiv.1706.03762`. arXiv: `1706.03762 [cs]`.

[Wu+16] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. Oct. 2016. DOI: `10.48550/arXiv.1609.08144`. arXiv: `1609.08144 [cs]`.

[Yul20] G. Yule. *The Study of Language*. 7th ed. Cambridge: Cambridge University Press, 2020.

[Zen+20] C. Zeng, S. Li, Q. Li, J. Hu, and J. Hu. *A Survey on Machine Reading Comprehension: Tasks, Evaluation Metrics and Benchmark Datasets*. Oct. 2020. DOI: `10.48550/arXiv.2006.11880`. arXiv: `2006.11880 [cs]`.

[Zha+25] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen. *A Survey of Large Language Models*. Mar. 2025. DOI: `10.48550/arXiv.2303.18223`. arXiv: `2303.18223 [cs]`.