The Cross-linguistic Phonological Frequencies (XPF) Corpus

Uriel Cohen Priva
Emily Strand
Shiying Yang
William Mizgerd
Abigail Creighton
Justin Bai
Rebecca Mathew
Allison Shao
Jordan Schuster
Daniela Wiepert

Department of Cognitive, Linguistic, and Psychological Sciences (CLPS)
Brown University
Providence, Rhode Island

2021

Preface

The Cross-linguistic Phonological Frequencies (XPF) Corpus is a collection of phonemic lexicons, or grammars, used for the primary purpose of studying information-theoretic universals. It was constructed within the Department of Cognitive, Linguistic, and Psychological Sciences (CLPS) at Brown University in the years 2018-2021. The composition of this corpus was funded by the National Science Foundation (NSF BCS-1829290).

This corpus would not have been possible without the tireless effort of a number of undergraduate students. We thank them for combing through and assessing existing documentation of languages' phonological systems, and then creating computational grammars to best reflect those languages using regular expressions. We offer special thanks to Rachel Gutman who made the creation of this corpus possible by collecting a pilot version, funded by Brown's UTRA program, which served as a proof of concept. We would also like to thank Nicholas Tomlin who created the initial JavaScript version of the main code. Finally, we would like to acknowledge Megan Kairiss, Madalyn Critz, and Delphine Morse Mahos for researching several languages that are now part of the corpus.

1 Information about the Corpus

1.1 Aim

One of the main goals of linguistics is to understand what universal trends persist across multiple unrelated languages (Greenberg, 1966). Resources such as the World Atlas of Language Structures (WALS, Dryer & Haspelmath, 2013), the UCLA Phonological Segment Inventory Database (UPSID, Maddieson, 1984), the Phonetics Information Base and Lexicon (PHOIBLE, Moran, McCloy, & Wright, 2014), and PBase (Mielke, 2008) have facilitated the discovery of many such trends, including ones that they were not originally designed to discover.

The growing availability of written language-use data made it possible to extend such questions beyond merely asking whether a phenomenon exists in a language (e.g. whether a particular language has a fixed word order) to asking how often and in what contexts a phenomenon occurs (e.g. whether word order minimizes dependency length, Futrell, Levy, & Gibson, 2020). One of the oldest universals in linguistics, Zipf's law of abbreviation (Zipf, 1935), belongs to the second category. The extension of such efforts to the segmental domain has been severely limited by the discrepancies that lie between written language and spoken language: Different languages employ different strategies to translate the representations of words to written form, and it is linguistically meaningless to draw conclusions from the distribution of particular characters.

The XPF corpus aims to bridge this gap. There are hundreds of languages whose written form can be translated back to its phonemic form, and when combined with existing corpora, such as the Crúbadán Corpus (Scannell, 2007), the phonemic representations can be used to answer questions that necessitate information about the fine grained distribution of sounds and their environments. The corpus comprises a set of rules that specify how alphabets can be translated to phonemic representations, language documentation summaries that justify particular correspondences, and Python and JavaScript code that can read a particular rule set, and use it to translate written input to its corresponding phonemic representation.

1.2 Name and abbreviation

The name of the corpus, Cross-linguistic Phonological Frequencies, should be abbreviated as XPF. The is meant to be pronounced as in *xylophone*, i.e. /z/, and the

is silent. Using American English standard vowel epenthesis rules, the recommended way to pronounce the abbreviation is therefore /zɪf/, as in the American English pronunciation of the name *Zipf*.

1.3 Distribution

The corpus is intended to be distributed under CC BY-NC-SA 4.0 license,[1] which means you may do the following:

Share Copy and redistribute the material in any medium or format

Adapt Remix, transform, and build upon the material

Under the following restrictions:

Attribution You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial You may not use the material for commercial purposes

ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

We reserve the right to provide the corpus under other licenses as well.

[1] Additional information about the license can be found here: https://creativecommons.org/licenses/by-nc-sa/4.0/.

The corpus is made available on GitHub (https://github.com/CohenPr-XPF/XPF), and is intended to be used for research purposes. We provide the following (all of which are described more in detail throughout the manual), where *langcode* stands for the BCP-47 code for each language (Phillips & Davis, 2009).[2]

- The documentation of each language (*langcode*.Rmd), see 1.4.1.
- The bibliography used for the documentation (*langcode*.bib).
- The rules to translate each language (*langcode*.rules), see 1.4.2.
- Sample word to IPA translations for each language (*langcode*.verify), see 1.5.1.
- The code responsible for implementing the translation rules (translate04.py - current version), see 1.5.
- A utility program for subsetting word frequencies of word lists (stopatn.sh), see 2.0.1.
- A utility program for obtaining informativity measures of segments within words (contextRep.py), see 2.1.
- A utility program for deriving summary statistics of language and frequency files (sumstats01.py), see 2.2.
- A summary table, which contains metainformation about the different languages (langslist.tsv), see 3.

1.4 Corpus Composition

The XPF Corpus consists of information for 201 languages, spanning 50 different language families. Table 1 displays the breakdown of the major language families represented in the corpus (i.e. those that have more than 5 associated languages), and Figure 1 displays the visual representation of the corpus' global spread. In terms of what's available for each language within the corpus, there are two main components: the language description and the corresponding phonemic grammar. These are described in 1.4.1 and 1.4.2 below.

As previously stated, our objective is to create phonologically rich data based on languages' orthographies. This may seem problematic; there are several aspects, such as stress and tone, that aren't generally indicated through orthography, which might compromise or limit certain types of research (Seifart, 2006). We gave up on representing stress in the corpus at an early stage, but we believe this information could be added by third parties in cases in which stress is predictable from written and phonemic forms.

In short, the corpus focuses on phonemic features, and may under-represent suprasegmental features. Moreover, our focus is put on the translation of words in isolation rather than in sentences

[2] con is a reserved name in Windows OS, and we therefore changed the *langcode* for Cofán to con_Cofan.

(see 1.5), so certain prosodic features like intonation that can extend phrasally are not available. The only suprasegmental we do account for is tone, and we only do so when it is clearly and systematically marked in the orthography (e.g. with diacritics like an acute accent (◌́), grave accent (◌̀), hook (◌̉), etc., as in Vietnamese, or with particular graphemes in identifiable positions such as with ⟨b⟩, ⟨j⟩, ⟨v⟩, etc., as in Hmong syllable codas).[3] Apart from this, we try to avoid ambiguity and conflation as much as possible, which is most crucially maintained in the language selection process.[4] The main criteria for selecting languages involved identifying those that were phonemically consistent, meaning they had one-to-one correspondences between the graphemes in their orthography and their phonemes. We were, however, unable to avoid this completely, which is discussed more in 1.4.1.

Table 1: The distribution of languages in the corpus. *Other* stands for language isolates, language groups that are represented by fewer than 5 languages, and all creoles.

Language family	Number of languages in language family
Arawakan	9
Austronesian	28
Indo-European	24
Mayan	12
Niger-Congo	7
Trans-New Guinea	24
Turkic	13
Other	84

1.4.1 Language Descriptions

Each language within the corpus has a language description (*`langcode`*`.Rmd` files) providing basic information, which was consolidated from existing documentation.[5] Each description includes the following sections:

1. Background: the language family and the geographic area in which it is spoken

[3]During the language selection process, if a language was indicated to have contrastive tone, yet it was unmarked, we "abandoned" the language (i.e. it wasn't included in the corpus). This method was applied to other languages that had several phonemic contrasts that were not recoverable from their alphabetic representation.

[4]The Crúbadán Corpus (an NSF-funded resource, Scannell, 2007) was used as the main point of reference for selecting the different languages.

[5]A reference list of all sources used in this corpus is provided here `TODO: provide`.

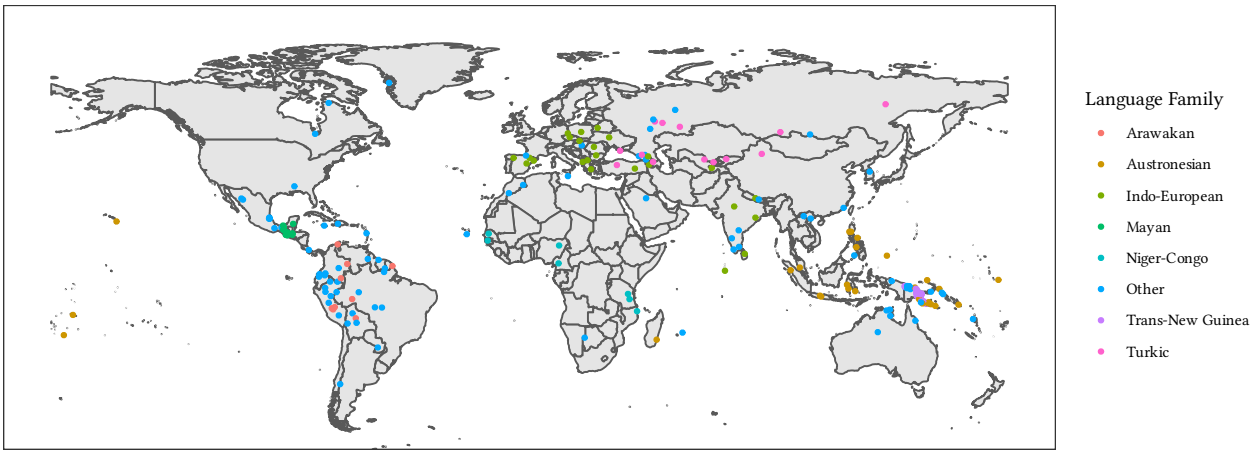

Figure 1: World Map of XPF Corpus Languages

2. Phonology[6]: the consonant, vowel, and tonal (if applicable) inventories
3. Alphabet/orthography: the grapheme to phoneme correspondences
4. Syllable structure (if available/relevant)
5. Lenition rules (e.g. degemination, voicing, spirantization, deletion)
6. Miscellaneous information or relevant phonological processes
7. References

Depending on the language, the aforementioned sections may include more detail than others. More detail generally accompanies languages that either had contradicting existing documentation (e.g. disagreement around native phonemes) or ambiguous orthographies (i.e. the alphabet was inexplicit, or the correspondences between graphemes and phonemes weren't clearly defined). In lieu of these occurrences, the more likely or the most corroborated descriptions were prioritized and justified. However, this does mean that our grammars may not agree with particular analyses or theoretical viewpoints.[7]

Although the language descriptions in their entirety are important, some sections are more crucial than others, especially in terms of constructing the phonemic grammars. The phonological inventories and alphabets are undoubtedly the most important for this aspect, given that we need the explicit grapheme to phoneme correspondences for the construction. The other sections mainly

[6]Although languages tend to have loanwords (Tadmor, 2009), resulting in the adoption of non-native phonemes, we strictly account for native phonemes. The motivation for this decision stems from the desire to capture a given language's native phonology with as little noise, or outside influence as possible. Moreover, it serves as an additional method of eliminating chances of phoneme conflation. Loanwords, therefore, may be mistranslated.

[7]All components of the corpus are made available, including the raw code. Therefore, should any of the grammars warrant modification due to alternative interpretations, human error, or alternative research purposes, changes can be easily made.

help give a more well rounded picture of each language, and the lenition section in particular provides possible points of analysis that could be used to observe cross-linguistic tendencies.

Given that the corpus is dependent on each language's grapheme to phoneme correspondences, we provide an example inventory in Table 2, which is reflective of the *alphabet/orthography* section in each of the language descriptions. This example inventory is representative of a phonemically consistent language, but as previously mentioned, we were unable to avoid phoneme conflation completely. This means that some languages (roughly 40) involve a one-to-two or a one-to-many correspondence for at least one grapheme. For example, Batak Karo, an Austronesian language, uses the grapheme ⟨e⟩ to represent /e/, /ɯ/, and /ə/ (Woollams, 1996). Languages with conflation are still provided with a grapheme to phoneme inventory, however, they are accompanied by a *comment* column that specifies the default correspondence that's represented in the constructed grammar. So, the *comment* in the Batak Karo example is: **/ə/: default in the rules**. The default correspondences are justified within the documentation, but in most cases, they prioritize the representation of the most (or what appears to be the most) frequently occurring phoneme given the options. Such languages were flagged as "compromised."

There were other issues that led languages to be flagged as compromised too (about 20 languages). One common reason was that certain phonemes weren't consistently marked in the orthography. Some languages, for example, do not mark or inconsistently mark the glottal stop (e.g. Nigerian Fulfulde: McIntosh, 1982; Mekeo: Chung, 2009; Tz'utujil: Dayley, 1985). This observation tends to be a point of debate around whether or not the unmarked or inconsistently marked phoneme is even a phoneme or not, but that's why we flag these languages in the first place. Other reasons for a language to be compromised include extremely inconsistent documentation, ambiguity related to the marking of diacritics (predominant in abugidas), and phonetically based orthographies (i.e. those that represent surface level or allophonic production).

For all 62 compromised languages, the compromising elements are presented at the top of the language descriptions. They are also documented in the file `langs-list.tsv`, either as a comment or regular expression, if applicable (see 3).

The complete list of available languages within the corpus is presented in Table 3, which specifies the language, the corresponding BCP-47 code, the language family, and the status (i.e. whether it's compromised or not).

Table 2: Grapheme to Phoneme Correspondences in Benabena

Grapheme	Phoneme
a	/ɑ/
b	/ɸ/
e	/e/
f	/f/
g	/ɣ/
h	/h/
i	/i/
k	/k/
l	/l̪/
m	/m/
n	/n/
o	/o/
p	/p/
s	/s/
t	/t/
u	/u/
y	/j/
’	/ʔ/

1.4.2 Language Grammars

Along with the descriptions, each language has a corresponding phonemic grammar (*lang-code*`.rules` files). These are the computational representations of the grapheme to phoneme correspondences that can be used to translate a language sample (e.g. a word list) into its phonemic form. The grammar for the correspondences presented in Table 2 is shown below.

```
# Benabena Rule Set
# Written by: Bill
# Last Updated: 2019-09-30
type,sfrom,sto,weight,precede,follow,comment
# Classes
class,apostrophe,['''’’],,,,
class,passthrough,[efhikmnopstu],,,,
```

```
# Individual Letters
sub,a,ɑ,4,,,
sub,b,ɸ,4,,,
sub,g,ɣ,4,,,
sub,l,ɭ,4,,,
sub,y,j,4,,,
sub,{apostrophe},ʔ,4,,,
# Misc. Rules
sub,({passthrough}),\1,0.1,,,
```

The most meaningful part of this grammar and any of the grammars within the corpus is the instructions, which are presented by the line: *type,sfrom,sto,weight,precede,follow,comment*. Each of the subsequent lines are thought of as rules, written through a series of regular expressions, outlined by these instructions. The bulk of these instructions depend on the first classification, which is *type*, so we explain the remaining instructions accordingly for each *type* below. ***type*** refers to the type of the rule in question. There are six different rule types and each one dictates how the remaining instructions are handled or what the input for those instructions should be (mainly important for *sfrom* and *sto*).

1. ***word*** rules: *word* rules are plain substitution rules. If a word from a language sample matches the *sfrom* column, it is replaced by the *sto* column, and avoids further processing. Thus, these rules only make use of the *sfrom* and *sto* columns.

   Example:

   ```
   word,cat,k æ t,,,,
   ```

2. ***pre*** rules: All translations from orthographic to phonemic representation occur in lowercase. For some languages, the processing of certain lowercase characters/graphemes goes wrong. These *pre*processing rules, however, strictly deal with this as a first step in the translation process as they explicitly translate problematic uppercase forms to the correct lowercase forms. Both the *sfrom* and *sto* columns are expected to contain orthographic input: *sfrom* with the uppercase form, and *sto* with the lowercase form. Translation is done by index: The first character in the *sfrom* column is replaced with the first character from the *sto* column, the second with the second, etc. This strategy makes it possible to perform only very simple substitutions, akin to `tr` in POSIX or `chartr()` in R, and is used very sporadically in the corpus. In most instances, it controls for the dotted Iı and dotless İi in certain Latin alphabets (e.g. Azerbaijani). *pre* rules only make use of *sfrom* and *sto*.

   Example:

```
pre,Iİ,ıi,,,,"uppercase to lowercase conversion",
```

3. ***match*** rules: *match* rules are plain substitution rules at the level of an individual letter. If a letter matches the *sfrom* column, it is replaced by the *sto* column, and avoids further processing. This particular rule type is only used in the constructed Korean grammar, with each rule being automatically generated, given the sheer number of possible stroke combinations within a syllable block.[8]

   Example:

```
"match","가","{G} {A}",1,"","","autogenerated from GA"
"match","각","{G} {A} {G}",1,"","","autogenerated from GAG"
"match","갂","{G} {A} {GG}",1,"","","autogenerated from GAGG"
```

4. ***sub*** rules: *sub* rules are the core part of the translation scheme, and rely on input from all of the instructions mentioned prior. Given orthographic input, a *sub* rule is applied if *sfrom* matches the grapheme in question, *precede* matches the preceding context, *follow* matches the following context, and the *weight* is the highest of all rules that concern that particular grapheme (the behavior is not defined if multiple rules with identical weights for the grapheme in question match). If no rule meets these requirements, the grapheme is translated to some non-IPA default, currently @. Irrespective of the explanations below, every property (except *weight*) can contain *class* rules (described below in 5).

   - ***sfrom*** represents one alphabetical letter (as defined by the utf8 standard).
   - ***sto*** represents zero or more phonemes (space-separated) that the letter should translate to. The phonemes are presented in IPA. In a handful of cases *sto* contains non-IPA outputs that are subsequently rewritten by *ipasub* rules (see below).
   - ***precede*** represents a regular expression that must match the preceding context. This expression ends with $ (added by the program), ensuring that the expression matches the context that immediately precedes *sfrom*. ^ in the *precede* column therefore designates the beginning of a word, leaving ^$ as an empty string.
   - ***follow*** similarly represents a regular expression that must match the following context. This expressions starts with ^, ensuring that it applies to the context immediately

[8] Each Hangul syllable block, of which there are 11,172, is encoded precompositionally, thus treated as an individual "letter". We did not opt for compositional encoding (i.e. encoding individual jamos, or alphabetic characters, based on initial, medial, or final position within a syllable block), as it is not the preferred method for most software nor is it as robust (Tan & Lim, 2011; *The unicode standard version 5.2.0*, 2009). The *match* rules could have been performed using *sub* rules (see below); however, their reliance on regular expressions is computationally expensive. Therefore, we opted for the computationally cheap method of direct matching.

following the letter in question. $ in the *follow* column therefore designates the end of a word, leaving ^$ as an empty string.

- ***weight*** represents the relative priority of the rule, with higher values overriding lower values. The goal is to designate rules that take priority over other rules. This is crucial in instances where one grapheme with a particular phoneme correspondence also makes up part of a digraph with a separate phoneme correspondence. For example, if we have a default rule for ⟨c⟩ (which translates to k), and a specific rule for ⟨c⟩ (which translates to /tʃ/ when ⟨h⟩ follows, representative of the digraph ⟨ch⟩), we want to make sure the specific rule applies rather than the lower priority rule when appropriate. Thus, we give the specific rule a higher weight.

Example:

```
sub,c,k,1,,,"Default rule for c"
sub,c,tʃ,2,,h,"Specific rule for c"
```

When handling digraphs or multigraphs, typically separate rules need to be added to control for all the letters that compose them. For instance, with the ⟨ch⟩ example above, given that ⟨c⟩ is in the *sfrom* column, it is directly translated to /tʃ/; however, ⟨h⟩ is merely used as a contextual identifier for the translation of ⟨c⟩, so we still need to control for the translation of ⟨h⟩. Specifically we need an additional rule to translate ⟨h⟩ to nothing as it is no longer relevant in this particular context. This would be represented as such: `sub,h,,2,c,,"clean-up"`.

5. ***class*** rules: *class* rules define internal substitution guidelines meant to improve readability or allow reuse to recurring elements. Only the *sfrom* and *sto* columns are used. *sfrom* defines the name of the class, and *sto* defines the value of the class. In terms of improving readability, *class* rules are specifically helpful for non-Latin alphabets, given that it is not practical to manually type these characters repeatedly in subsequent *sub* or *ipasub* rules (discussed below).

   Example:

   ```
   class,pi,π,,,,,
   sub,{pi},p,1,,,,
   ```

   Regarding reuse, *class* rules are beneficial in eliminating the redundancy of similarly behaving graphemes. For example, instead of writing the same *sub* rule three times for ⟨c⟩ such that it translates to /k/ preceding ⟨a⟩, ⟨o⟩, and ⟨u⟩, the similar contextual graphemes can be put into a *class*, resulting in only one *sub* rule (⟨c⟩ translates to /k/ preceding back vowels).

Example:

```
class,back_vowels,[aou],,,,
sub,c,k,2,,{back_vowels},"<c> translates to /k/ when preceding back vowels",
```

6. ***ipasub*** rules: *ipasub* rules are responsible for performing substitutions on the output IPA directly. That is, they are the final step within the translation process. They are intended to be used sparingly; however, given certain orthographies (e.g. those with a number of diacritics) or phonological inventories (e.g. those with phonemic consonant/vowel length), they may greatly increase readability. *sfrom* designates the input pattern, *sto* designates the replacement string, and *weight*, unlike *sub* rules, designates the order of application. These can be as complex as python3 allows, but should ideally be used to change what constitutes a phoneme.

   Example:

```
ipasub,({consonant}) \1,\1 ː,1,,,treat consonant gemination as a special phoneme (e.g. t t -
ipasub,({vowels}) \1,\1ː,1,,,treat vowel gemination as a distinct phoneme each time (e.g. i
ipasub, ʷ,ʷ,1,,,attaches labialization diacritic to phonemes
```

Phonemic length was handled computationally in two separate ways. If length was contrastive for consonants or vowels of particular natural classes (e.g. voiceless stops; tense vowels), the length diacritic (ː) was kept separate from those phonemes (e.g. /k ː/). However, if length was a contrastive feature of idiosyncratic phonemes, the diacritic was "attached" to the phoneme (e.g. /kː/). Most languages within the corpus that have this feature were coded under the former approach. We additionally mark naturally occurring gemination in some languages where it occurs (typically at syllable boundaries), but such phonemes that undergo this process are also space-separated from the diacritic. These two alternatives are exemplified above.

If a language can be represented by a number of alphabets (e.g. Cyrillic and Latin), it may have multiple rules files. Currently five languages have two separate rules files.

1.5 Translation Scheme

Once the computational grammars are specified, they can be tested with the developed translation scheme. The most current version of this translation scheme is `translate04.py`. Essentially, it iterates through the specified rules when presented with a list of words, and translates them to their phonemic form.

Rules can be tested using python3 (or python depending on the machine), once the name of the script (translate04.py), the language specific rules file, and the words to be translated are specified.

Using the rules file for Benabena (bef as the BCP-47 language code) as an example, we would type the following in the terminal:

```
python3 translate04.py -l bef.rules cat brick log
```

This is merely an example; the words chosen to be translated are not native to Benabena, so certain graphemes are not translatable and the output, which is presented below, is not representative of the language's phonology. For more informative results, words that are actually present within the selected language should be used.

```
cat     @ ɑ t
brick   ɸ @ i @ k
log     l o ɣ
```

The example above demonstrates the simplest method of implementing the translation program. However, the program has a number of parameters that make for more efficient and justifiable translations. These include:

-l specifies the rules file (as described above), which should be in csv format.

-v specifies the log level (higher → more information). In future releases the distinction between different log levels would be clearer.

-c specifies the verification file (described below in 1.5.1), which should be in csv or tsv format.

-r specifies a file from which words should be read for translation. Only the first word in every line is read, to facilitate a use case with a counts column.

This might look like:

```
python3 translate04.py -l bef.rules -c bef.verify.csv -r bef_words.txt
```

or, with piped input (using the Crúbadán Benabena word list):

```
bzcat bef.txt.bz2 | python3 translate04.py -l bef.rules -c bef.verify.csv -r-
```

Depending on the words within the text file, the output should be extremely similar to the output in the first example.

1.5.1 Grammar Verification

Considering that the grammars were all manually constructed, we needed to verify that they behaved how they were expected to. To do so, we created sample word to IPA translations for each language (*langcode*`.verify` files), and ran them against the translation scheme. Notification

of incorrectly translated words signified that the constructed grammar was somehow incorrect and in need of modification. Our main objective was to provide at least one word that tested the behavior of each phoneme within the language. Each verify file is in csv format, and specifies the word, the expected IPA (space-separated) output, and any additional comments (e.g. what phoneme is the focus). A sample of Benabena's verify file (`bef.verify.csv`) is provided below.

```
akaluya'a,ɑ k ɑ l̠ u j ɑ ʔ ɑ,"a"
bi'ehibe,ɸ i ʔ e h i ɸ e,"b"
etehi,e t e h i,"e"
falu'afu,f ɑ l̠ u ʔ ɑ f u,"f"
gegisa,ɣ e ɣ i s ɑ,"g"
```

Although all grammars have been verified, we still recommend involving some sort of verify file for any translation as a sort of sanity check.

2 Utility Programs

In addition to the general translation scheme, we provide a variety of utility programs that are likely to aid future research.

2.0.1 Evaluating Frequency Files

`stopatn.sh` is a script that reads word frequency files from standard input, and subsets specific word/frequency data based on set parameters. The program has the format `stopatn.sh n [min-freq]`, and accomplishes three things:

1. It removes any word that has a frequency lower than *minfreq* (defaults to one if there is no filtering).
2. It finds the frequency of the n^{th} word in decreasing frequency order.
3. It provides all words whose frequencies are at least as high as that of the n^{th} word.

The program reads word frequency lists from standard input. The input is assumed to have Crúbadán format (i.e. *word freq*). Here is an example using the Crúbadán word list for Benabena:

```
bzcat bef.txt.bz2 | ./stopatn.sh 5 2000
```

Therefore, after eliminating words that have frequencies less than 2000, the program finds the 5^{th} position (`lu 3819`), in descending order, while providing all words with frequencies that are at least as high:

```
huto 4741
```

```
luto 4367
yabe 4201
To 4083
lu 3819
```

In the example above, although the word at the 5th position has a unique frequency within the Benabena word list, we must reiterate that the program does not discriminate if there are multiple words with the same frequency at the n^{th} position. That is, all words at the specified position will be accounted for and included in the output. The program therefore controls for alphabetic bias unlike the standard `head -n` *n* command.

For instance, assuming that the text file input is:

```
A 5
B 4
C 4
D 3
```

`head -n 2` would produce `A 5` and `B 4`, but `stopatn.sh 2` would also produce `C 4` because `C` has the same frequency as `B`.

`stopatn.sh` assumes it is invoked under `bash`, and that POSIX `sort`, `head`, and `tail` are all available.

2.1 Segment Informativity Measures

`contextRep.py` is a utility code responsible for calculating informativity measures of segments within a word (Cohen Priva, 2008). It does so by comparing the likelihood of observing specific segments in particular contexts to the likelihood of observing all segments in those same particular contexts. This program is currently implemented by `sumstats01.py` (described below), and has no independent format.

2.2 Summary Statistics

`sumstats01.py` derives summary statistics of language and frequency files. Specifically, with the help of `translate04.py` and `contextRep.py`, it calculates and displays the frequency counts and informativity measures of a language's phonemes based on an inputted word list. In addition, it calculates the percentage of words that can't be translated. The format for sumstats01.py (`python3 sumstats01.py -l LANGRULES [-c CHECK] [-r READ] [-m MIN] [-N] [-A] [-@ MAX@]`) is extremely similar to that of translate04.py, but it has additional parameters:

-l specifies the rules file, which should be in csv format.

-c specifies the verification file, which should be in csv or tsv format.

-r specifies a file from which words should be read for translation. Only the first word in every line is read, to facilitate a use case with a counts column.

-m specifies the minimum frequency to consider (defaults to 1 if unspecified).

-N suppresses summary information.

-A enumerates all words and probabilities (not currently supported).

-@ specifies the number of @ words to include in the summary.

For Benabena, this might look like:

```
python3 sumstats01.py -l bef.rules -c bef.verify.csv -r bef_words.txt
```

Output from certain commands and scripts (e.g. `cat`, `bzcat`, and `stopatn.sh`) can also be piped into `sumstats01.py` for analysis:

```
bzcat bef.txt.bz2 | ./stopatn.sh 5 2000 | python3 sumstats01.py -l bef.rules -
c bef.verify.csv -r bef_words.txt
```

The output of the example immediately above is presented below:

```
seg     informativity   ,count
ɑ       0.8760839355957686      194761.0
i       0.9857145072876157      139055.0
o       1.0284684906591073      109887.0
e       1.5699446314044685      106487.0
n       1.674649351104794       86605.0
l       2.1032185229521403      81180.0
h       1.681832682370468       67522.0
u       1.4940892170154876      66549.0
t       2.225576768421279       59317.0
ʔ       1.516808735143856       56922.0
m       2.2653764701844668      54622.0
ɸ       2.7737975472008305      40291.0
ɣ       2.3609439860066157      36594.0
k       3.3596143633894267      29630.0
j       3.1931525642526615      27244.0
```

```
s       3.1362605877238 20090.0
f       2.9103088863125226      19055.0
p       2.421597163332841       16058.0
## Summary statistics:
##   processed (inc. skipped): 3938
##   skipped: 0
##   %@ words: 0.1
## Top missing:
##   ï → '@' (287)
##   ROMU → '@ o m u' (35)
```

3 Metainformation

The metainformation summary table (`langs-list.tsv`) provides summary information about all the languages. For each language, it contains:

`code` The BCP-47 language code.

`name` The language name.

`family` The language family.

`macroarea` The macroarea in which it belongs (Hammarström & Donohue, 2014).

`nrules` The number of rules files.

`rules, rules_2` The rules files associated with the language. We ranked these based on how successful the translations were, and referenced the corresponding verify and Crúbadán word lists appropriately (i.e. `rules` vs. `rules_2`, `verify` vs. `verify_2`). Specifically, we prioritized the rule sets that had the fewest untranslated words.

`verify, verify_2` The verify files for the rules.

`compromised` A regular expression representing the language's compromising element(s), typically conflation (only applicable for compromised languages).

`compromised_other` A comment that specifies the language's compromising element(s) if it can't be represented by a regular expression.

We also added information that is mostly relevant to our own research purposes and less so for others that intend to use the corpus. These include mainly pointers to word frequency lists, e.g. Crúbadán (Scannell, 2007), OpenSubtitles (Tiedemann, 2009), and corpora based on LDC's

IARPA Babel. They fall under the columns `crubadan`, `crubadan2` (which correspond to `rules` and `rules_2`), `opensubtitles`, and `ldc`. The word frequency lists are not available with the corpus because they are licensed under their respective sources.

4 For the Average User

The foundation of this corpus relies on translating words to their phonemic form depending on the language. One of our main goals was to enable future academic research. But what about the general public? We feel that academic research and the resources used to implement it can be unnecessarily complex for the general public, which diminishes intrigue. We further acknowledge that corpus research and all the computational work required to use it may be daunting to the average person. Considering that the core translation scheme of the XPF corpus might actually be of interest to a lot of people, especially perhaps to those who are just starting their education in linguistics or to those that are trying to teach or learn a new language, we have created a user friendly version of this on our website (https://cohenpr-xpf.github.io/XPF/Convert-to-IPA.html) where no manipulation of code is required. The website currently provides ways to translate alphabets to their phonemic representations, and to compare language use files to see which words are over-represented and under-represented.

Table 3: XPF Corpus Languages

Language	BCP-47 Code	Language Family	Status
Abau	aau	Sepik-Ramu	
Abkhaz	ab	Northwest Caucasian	
Akawaio	ake	Cariban	Compromised
Alamblak	amp	Sepik-Ramu	Compromised
Albanian	sq	Indo-European	
Amanab	amn	Trans-New Guinea	
Amele	aey	Trans-New Guinea	
Angor	agg	Trans-New Guinea	
Anjam	boj	Trans-New Guinea	
Ankave	aak	Trans-New Guinea	
Apalaí	apy	Cariban	
Apurinã	apu	Arawakan	
Arabela	arl	Zaparoan	
Arabic	ar	Afro-Asiatic	Compromised
Aragonese	an	Indo-European	
Armenian	hy	Indo-European	
Arosi	aia	Austronesian	
Asháninka	cni	Arawakan	Compromised
Asturian	ast	Indo-European	
Au	avt	Torricelli	
Awara	awx	Trans-New Guinea	Compromised
Aymara	ay	Aymaran	
Azerbaijani	az	Turkic	
Bargam	mlp	Trans-New Guinea	Compromised
Bashkir	ba	Turkic	
Basque	eu	Isolate	
Batak Karo	btx	Austronesian	Compromised
Belarusan	be	Indo-European	
Benabena	bef	Trans-New Guinea	
Bislama	bi	English Creole	
Bora	boa	Witotoan	
Borong	ksr	Trans-New Guinea	
Bribri	bzd	Chibchan	Compromised
Bugis	bug	Austronesian	
Bulgarian	bg	Indo-European	
Bunama	bdd	Austronesian	
Burarra	bvr	Australian	
Candoshi-Shapra	cbu	Isolate	
Cape Verdean Creole	kea	Portuguese Creole	
Carib	car	Cariban	
Catalan	ca	Indo-European	Compromised
Cavineña	cav	Tacanan	Compromised
Central Atlas Tamazight	tzm	Afro-Asiatic	Compromised
Central Bikol	bcl	Austronesian	Compromised
Cha'palaa	cbi	Barbacoan	
Chavacano	cbk	Spanish Creole	
Chayahuita	cbt	Cahuapanan	
Choctaw	cho	Muskogean	Compromised
Chol	ctu	Mayan	

Table 3: XPF Corpus Languages *(continued)*

Language	BCP-47 Code	Language Family	Status
Chuvash	cv	Turkic	
Ch'orti'	caa	Mayan	
Cofán	con	Isolate	Compromised
Colorado	cof	Barbacoan	Compromised
Crimean Tatar	crh	Turkic	
Cusco Quechua	quz	Quechuan	
Czech	cs	Indo-European	
Daga	dgz	Trans–New Guinea	
Dedua	ded	Trans-New Guinea	
Djambarrpuyngu	djr	Pama-Nyungan	
Erzya	myv	Uralic	Compromised
Ese	mcq	Trans-New Guinea	
Francisco León Zoque	zos	Mixe-Zoque	
Gapapaiwa	pwg	Austronesian	
Georgian	ka	Kartvelian	
Guarani	gn	Tupian	
Guayabero	guo	Guahiboan	
Guhu-Samane	ghs	Trans-New Guinea	
Haitian Creole	ht	French Creole	
Hawaiian	haw	Austronesian	
Hiligaynon	hil	Austronesian	
Hindi	hi	Indo-European	Compromised
Hmong	hmn	Hmong-Mien	
Huallaga Huánuco Quechua	qub	Quechuan	Compromised
Huarijío	var	Uto-Aztecan	
Huehuetla Tepehua	tee	Tepehua	
Hungarian	hu	Uralic	
Iduna	viv	Austronesian	
Ignaciano	ign	Arawakan	
Ilocano	ilo	Austronesian	
Indonesian	id	Austronesian	Compromised
Inga	inb	Quechuan	
Inuktitut	iu	Eskimo-Aleut	
Ixil	ixl	Mayan	Compromised
Ixtatán Chuj	cnm	Mayan	
Jamaican Creole	jam	English Creole	
Javanese	jv	Austronesian	
Jola-Fogny	dyo	Niger-Congo	Compromised
Kabardian	kbd	North Caucasian	
Kagulu	kki	Niger-Congo	
Kalaallisut	kl	Eskimo-Aleut	
Kannada	kn	Dravidian	
Karachay-Balkar	krc	Turkic	Compromised
Kayabí	kyz	Tupi	Compromised
Kazakh	kk	Turkic	Compromised
Kirghiz	ky	Turkic	
Kiribati	gil	Austronesian	
Komba	kpf	Trans-New Guinea	
Komi	kv	Uralic	

Table 3: XPF Corpus Languages *(continued)*

Language	BCP-47 Code	Language Family	Status
Korean	ko	Isolate	
Kuku-Yalanji	gvn	Pama-Nyungan	
Kunimaipa	kup	Trans-New Guinea	
Kwoma	kmo	Sepik-Ramu	Compromised
Macedonian	mk	Slavic	
Malagasy	mg	Austronesian	
Malayalam	ml	Dravidian	Compromised
Maldivian	dv	Indo-European	
Maltese	mt	Afro-Asiatic	Compromised
Mam	mam	Mayan	
Mamasa	mqj	Austronesian	
Manam	mva	Austronesian	
Mapos Buang	bzh	Austronesian	Compromised
Mapudungun	arn	Araucanian	Compromised
Mari	chm	Uralic	Compromised
Matsés	mcf	Panoan	Compromised
Mauwake	mhl	Trans-New Guinea	
Mekeo	mek	Austronesian	Compromised
Min Nan Chinese	nan	Sino-Tibetan	
Misima-Panaeati	mpx	Austronesian	Compromised
Modern Greek	el	Indo-European	
Moose Cree	crm	Algonquian	Compromised
Morisyen	mfe	French Creole	Compromised
Mountain Koiali	kpx	Trans-New Guinea	
Mufian	aoj	Torricelli	Compromised
Muna	mnb	Austronesian	Compromised
Mussau-Emira	emi	Austronesian	
Mwani	wmw	Niger-Congo	Compromised
Naasioi	nas	South Bougainville	
Nabak	naf	Trans-New Guinea	
Nahuatl	nhe	Uto-Aztecan	
Naro	nhr	Khoe-Kwadi	
Nehan	nsn	Austronesian	
Nepali	ne	Indo-European	Compromised
Nigerian Fulfulde	fuv	Niger-Congo	Compromised
Nobonob	gaw	Trans-New Guinea	
Nomaande	lem	Niger-Congo	
Nomatsiguenga	not	Arawakan	Compromised
Nunggubuyu	nuy	Gunwinyguan	
Ömie	aom	Trans-New Guinea	
Oriya	or	Indo-Aryan	Compromised
Ossetian	os	Indo-European	Compromised
Palauan	pau	Austronesian	
Palikúr	plu	Arawakan	Compromised
Pangasinan	pag	Austronesian	Compromised
Paumarí	pad	Arauan	
Pele-Ata	ata	isolate	
Piapoco	pio	Arawakan	
Pisaflores Tepehua	tpp	Totonacan	Compromised

Table 3: XPF Corpus Languages *(continued)*

Language	BCP-47 Code	Language Family	Status
Q'anjob'al	kjb	Mayan	
Qeqchi	kek	Mayan	Compromised
Rabinal Achi'	acr	Mayan	Compromised
Rawa	rwo	Trans-New Guinea	Compromised
Rikbaktsa	rkb	Macro-Ge	
Romanian	ro	Indo-European	
Rotokas	roo	North Bougainville	
Russia Buryat	bxr	Mongolic	
Saint Lucian Creole French	acf	French Creole	
Samoan	sm	Austronesian	Compromised
Sepik Iwam	iws	Sepik-Ramu	
Shilha	shi	Afro-Asiatic	
Shipibo Konibo	shp	Panoan	
Sinaugoro	snc	Austronesian	
Sinhala	si	Indo-European	
Slovak	sk	Indo-European	Compromised
Somba-Siawari	bmu	Trans-New Guinea	Compromised
South Tairora	omw	Trans-New Guinea	
Spanish	es	Indo-European	Compromised
Standard Malay	zsm	Austronesian	Compromised
Sunwar	suz	Sino-Tibetan	Compromised
Swahili	sw	Niger–Congo	Compromised
Tabasco Chontal	chf	Mayan	Compromised
Tajik	tg	Indo-European	
Tamil	ta	Dravidian	
Tarahumara	tac	Uto-Aztecan	
Tatar	tt	Turkic	
Telugu	te	Dravidian	
Tok Pisin	tpi	English Creole	
Tongan	to	Austronesian	
Totontepec Mixe	mto	Mixe-Zoque	
Turkish	tr	Turkic	
Tuvan	tyv	Turkic	
Tz'utujil	tzj	Mayan	Compromised
Tzotzil	tzo	Mayan	
Ukrainian	uk	Indo-European	
Upper Sorbian	hsb	Indo-European	
Usarufa	usa	Trans-New Guinea	
Uyghur	ug	Turkic	
Uzbek	uz	Turkic	
Vietnamese	vi	Austroasiatic	
Warlpiri	wbp	Pama-Nyungan	
Wayana	way	Cariban	
Wayuu	guc	Arawakan	
Wolof	wo	Niger-Congo	
Xicotepec de Juárez Totonac	too	Totonacan	Compromised
Yakut	sah	Turkic	Compromised
Yawa	yva	Yawa-Saweru	
Yiddish	yi	Indo-European	

Table 3: XPF Corpus Languages *(continued)*

Language	BCP-47 Code	Language Family	Status
Yine	pib	Arawakan	Compromised
Yucatec Maya	yua	Mayan	
Yucuna	ycn	Arawakan	
Yuracare	yuz	Isolate	
Zaza	zza	Indo-European	Compromised

References

Chung, J.-S. (2009). *Orthography paper for mekeo language in central province of papua new guinea*. SIL Language and Culture Archives.

Cohen Priva, U. (2008). Using information content to predict phone deletion. In N. Abner & J. Bishop (Eds.), *Proceedings of the 27th West Coast Conference on Formal Linguistics* (pp. 90–98). Somerville, MA: Cascadilla Proceedings Project.

Dayley, J. P. (1985). *Tzutujil grammar* (p. xvi+412). Berkeley; Los Angeles: University of California Press.

Dryer, M. S., & Haspelmath, M. (Eds.). (2013). *WALS online*. Retrieved from https://wals.info/

Futrell, R., Levy, R. P., & Gibson, E. (2020). Dependency locality as an explanatory principle for word order. *Language*, *96*(2), 371–412. https://doi.org/10.1353/lan.2020.0024

Greenberg, J. H. (1966). *Universals of language*. Joseph H. Greenberg.

Hammarström, H., & Donohue, M. (2014). Some principles on the use of macro-areas in typological comparison. *Language Dynamics and Change*, *4*(1), 167–187. https://doi.org/10.1163/22105832-00401001

Maddieson, I. (1984). *Patterns of sounds*. Cambridge: Cambridge University Press.

McIntosh, M. H. (1982). *Aspects of fulfulde syntax and morphology* (PhD thesis). University of London.

Mielke, J. (2008). *The emergence of distinctive features*. OUP Oxford.

Moran, S., McCloy, D., & Wright, R. (Eds.). (2014). *PHOIBLE online*. Retrieved from http://phoible.org/

Phillips, A., & Davis, M. (2009). *Tags for Identifying Languages*. https://doi.org/10.17487/RFC5646

Scannell, K. P. (2007). The Crúbadán project: Corpus building for under-resourced languages. *Building and exploring web corpora: Proceedings of the 3rd web as corpus workshop*, *4*, 5–15.

Seifart, F. (2006). *Essentials of language documentation* (J. Gippert, N. P. Himmelmann, & U. Mosel, Eds.). https://doi.org/10.1515/9783110197730

Tadmor, U. (2009). *Loanwords in the world's languages* (M. Haspelmath & U. Tadmor, Eds.). Retrieved from https://www.ebook.de/de/product/9194294/loanwords_in_the_world_s_languages.html

Tan, H. Y., & Lim, H. (2011). Unicode canonical decomposition for hangeul syllables in regular expression. *IEICE Transactions on Information and Systems*, *E94.D*(1), 146–154. https://doi.org/10.1587/transinf.E94.D.146

The unicode standard version 5.2.0. (2009). Mountain View, CA: The Unicode Consortium.

Tiedemann, J. (2009). News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, & R. Mitkov (Eds.), *Recent advances in natural language processing*: *Vol. V* (pp. 237–248). Borovets, Bulgaria: John Benjamins, Amsterdam/Philadelphia.

Woollams, G. (1996). *A grammar of karo batak, sumatra.* Pacific Linguistics.

Zipf, G. K. (1935). *The psycho-biology of language: An introduction to dynamic philology.* Houghton, Mifflin.