Model 1: A probabilistic model of segment borrowability

This report includes supplementary materials for:

```
Operationalizing borrowability: A case study from phonological segments
```

In [1]:

```
from collections import defaultdict, Counter
import pandas as pd
```

In [2]:

```
def load_cldf_dataset(path_to_values, path_to_languages):
    values = pd.read_csv(path_to_values)
    languages = pd.read_csv(path_to_languages)
    return pd.merge(left = values, right = languages, how="left",
                    left_on="Language_ID", right_on="ID")
```

A helper routine for computing typological frequencies.

When several inventories (doculects) are available for a single language, we collapse the inventories in them (i.e. take their union).

In [3]:

```
def get_frequencies_w_inventory_collapsing(dataset):
    glottocode_to_inventory = defaultdict(set)
    for row in dataset.itertuples():
        if not pd.isnull(row.Language_ID):
            glottocode_to_inventory[row.Language_ID].add(row.Value)
    print(f'{len(glottocode_to_inventory)} languages')
    frequencies_absolute = Counter()
    for segments in glottocode_to_inventory.values():
        for segment in segments:
            frequencies_absolute[segment] += 1
    frequencies_relative = {
        segment: count / len(glottocode_to_inventory)
        for segment, count in frequencies_absolute.items()
    }
    return frequencies_absolute, frequencies_relative
```

First, load the data from the CLDF format (Forkel et al. 2018) and combine the tables into single data frames including PHOIBLE (Moran and McCloy 2019) and SegBo (Grossman et al. 2020).

In [4]:

```
segbo = load_cldf_dataset('../data/segbo/cldf/values.csv',
                          '../data/segbo/cldf/languages.csv')
phoible = load_cldf_dataset('../data/phoible/cldf/values.csv',
                            '../data/phoible/cldf/languages.csv')
```

Number of different inventories in PHOIBLE:

In [5]:

```
n_phoible_inventories = len(phoible.Language_ID.unique())
n_phoible_inventories
```

Out[5]:

```
2177
```

Number of different inventories in SEGBO:

In [6]:

```
len(segbo.Language_ID.unique())
```

Out[6]:

```
498
```

Some languages in SEGBO are missing from PHOIBLE:

In [7]:

```
len(set(segbo.Language_ID) - set(phoible.Language_ID))
```

Out[7]:

```
199
```

We need to exclude them:

In [8]:

```
phoible_langs = set(phoible.Language_ID)
segbo = segbo.loc[ segbo.Language_ID.map(lambda gltc: gltc in phoible_langs) ]
```

In [9]:

```
len(segbo.Language_ID.unique())
```

Out[9]:

```
299
```

In [10]:

```
len(set(segbo.Language_ID) - set(phoible.Language_ID))
```

```
Out[10]: 0
```

We compute borrowability factors for a segment *s* (b_s) following the approach by Eisen (2019). We assume that the marginal probability of borrowing of *s* ($P_s(\text{borrowing})$) is equal to probability of contact between a language with this segment and a language lacking this segment ($P_s(\text{contact})$) multiplied by the conditional probability of borrowing of this segment in a contact situation ($P_s(\text{borrowing}|\text{contact})$):

$$P_s(\text{borrowing}) = P_s(\text{contact})P_s(\text{borrowing}|\text{contact})$$

We approximate $P_s(\text{borrowing})$ with the empirical relative frequency of borrowing (q_s) provided by SEGBO and PHOIBLE and assume, following Eisen (2009), that $P_s(\text{contact})$ can be estimated as a product of the relative typological frequency of a segment (f_s) and its comlement $(1 - f_s)$:

$$P_s(\text{contact}) \propto f_s(1 - f_s)$$

f_s ranges from 0 to 1, and in order for $P_s(\text{contact})$ to integrate to 1 on this interval, we need to introduce a normalisation constant equal to 6 to obtain valid probabilities:

$$P_s(\text{contact}) = 6f_s(1 - f_s)$$

Here and in Model 2, we define b_s to be $P_s(\text{borrowing}|\text{contact})$, which gives

$$q_s = 6f_s(1 - f_s)b_s$$
$$b_s = \frac{q_s}{6f_s(1 - f_s)}$$

f_s is equal to the number of occurrences of *s* in PHOIBLE divided by the number of distinct languages in PHOIBLE.

q_s is equal to the number of occurrences of *s* as a borrowed segment in SEGBO again divided by the number of distinct languages in PHOIBLE: languages without borrowed segments were not included in SEGBO, which therefore cannot be used as a source of negative data.

In [11]:

```
# We cannot use vanilla relative frequencies: for SEGBO, we need the number of distinct languages from PHOIBLE;
# for PHOIBLE see below.

(
    phoible_frequencies_absolute,
    _
) = get_frequencies_w_inventory_collapsing(phoible)

(
    segbo_frequencies_absolute,
    _
) = get_frequencies_w_inventory_collapsing(segbo)
```

```
2177 languages
299 languages
```

In [12]:

```
segbo_frequencies_relative = {
    segment: count_segbo / n_phoible_inventories
    for segment, count_segbo in segbo_frequencies_absolute.items()
}
```

Vanilla relative frequencies from PHOIBLE produce valid results in most cases, but problems arise with some rare segments. E.g., when a rare segment was borrowed from language A to language B, it may happen that language A then quickly loses it. As a result, this segment may have a higher frequency in SEGBO than in PHOIBLE, which makes the derivation ill-defined.

In order to avoid this issue we create two versions of absolute PHOIBLE frequencies -- one where the values are greater than or equal than in SEGBO and one where they are strictly greater (through Laplace smoothing) -- and then use these absolute frequencies to compute relative typological frequencies.

In [13]:

```python
phoible_greater_or_equal = {}
phoible_strictly_greater = {}
for segment, count_segbo in segbo_frequencies_absolute.items():
    if count_segbo >= phoible_frequencies_absolute[segment]:
        print(segment, count_segbo, phoible_frequencies_absolute[segment])
        phoible_greater_or_equal[segment] = count_segbo
        phoible_strictly_greater[segment] = count_segbo + 1
    else:
        phoible_greater_or_equal[segment] = phoible_frequencies_absolute[
            segment]
        phoible_strictly_greater[segment] = phoible_frequencies_absolute[
            segment] + 1
```

```
ʕ̞ 1 0
ʊai 1 0
ɸʷ 1 1
pʷʰ 1 1
tsʲʰ 1 1
d̤ʒ̤ 1 1
ɻ̤ 1 0
uə̯ 1 1
ɨə̯ 1 0
n̪ˤ 1 1
l̪ˤ 1 1
n̺d̺z̺ 1 0
ðˠ 1 1
```

In [14]:

```python
phoible_freqs_relative = {
    segment: count / n_phoible_inventories
    for segment, count in phoible_greater_or_equal.items()
}
phoible_freqs_relative_laplace = {
    segment: count / n_phoible_inventories
    for segment, count in phoible_strictly_greater.items()
}
for segment, f_s in sorted(phoible_freqs_relative.items(),
                           key=lambda el: el[1], reverse=True)[:10]:
    print(f'{segment}: {f_s}, {phoible_freqs_relative_laplace[segment]}')

# Smoothing has no effect on frequent segments.
```

```
m: 0.9701423977951309, 0.9706017455213597
k: 0.9205328433624254, 0.9209921910886542
j: 0.915480018373909, 0.9159393661001378
u: 0.9150206706476803, 0.915480018373909
a: 0.9108865411116215, 0.9113458888378503
p: 0.870463941203491, 0.8709232889297198
w: 0.864951768488746, 0.8654111162149747
n: 0.8474965548920533, 0.847955902618282
t: 0.7606798346348186, 0.7611391823610473
l: 0.7266881028938906, 0.7271474506201194
```

In [15]:

```
# Now we can compute borrowability scores using Eisen's formula with the normalisation constant

def borrowability_score(q_s, f_s):
    return q_s / f_s / (1 - f_s) / 6

borrowability_scores = {}
borrowability_scores_laplace = {}
for segment in segbo_frequencies_relative:
    borrowability_scores[segment] = {
        'Segment': segment,
        'Borrowability': borrowability_score (
            segbo_frequencies_relative[segment],
            phoible_freqs_relative[segment]
        ),
        'PHOIBLE_frequency_absolute': phoible_greater_or_equal[segment],
        'PHOIBLE_frequency_relative': phoible_freqs_relative[segment],
        'SEGBO_frequency_absolute': segbo_frequencies_absolute[segment],
        'SEGBO_frequency_relative': segbo_frequencies_relative[segment]
    }
    borrowability_scores_laplace[segment] = {
        'Segment': segment,
        'Borrowability': borrowability_score (
            segbo_frequencies_relative[segment],
            phoible_freqs_relative_laplace[segment]
        ),
        'PHOIBLE_frequency_absolute': phoible_strictly_greater[segment],
        'PHOIBLE_frequency_relative': phoible_freqs_relative_laplace[segment],
        'SEGBO_frequency_absolute': segbo_frequencies_absolute[segment],
        'SEGBO_frequency_relative': segbo_frequencies_relative[segment]
    }
```

In [16]:

```
borrowability_df = pd.DataFrame.from_dict(borrowability_scores).T.sort_values(by='Borrowability', ascending=False)
```

In [17]:

```
# Frequently borrowed segments
borrowability_df.loc[ borrowability_df.SEGBO_frequency_absolute >= 10 ]
```

Out[17]:

	Segment	Borrowability	PHOIBLE_frequency_absolute	PHOIBLE_frequency_relative	SEGBO_frequency_absolute	SEGBO_frequency_relative
f	f	0.031313	968	0.444649	101	0.046394
p	p	0.021048	1895	0.870464	31	0.01424
g	g	0.016306	1255	0.576481	52	0.023886
b	b	0.013562	1385	0.636197	41	0.018833
z	z	0.013523	682	0.313275	38	0.017455
ʒ	ʒ	0.013064	331	0.152044	22	0.010106
d̠ʒ	d̠ʒ	0.011803	640	0.293983	32	0.014699
v	v	0.011686	617	0.283418	31	0.01424
d	d	0.011638	1097	0.503904	38	0.017455
x	x	0.010998	411	0.188792	22	0.010106
h	h	0.010671	1258	0.577859	34	0.015618
ʃ	ʃ	0.010643	782	0.35921	32	0.014699
l	l	0.009251	1582	0.726688	24	0.011024
r	r	0.008882	1099	0.504823	29	0.013321
t̠ʃ	t̠ʃ	0.008795	916	0.420763	28	0.012862
o	o	0.008581	1446	0.664217	25	0.011484
ɾ	ɾ	0.008309	615	0.282499	22	0.010106
s	s	0.007337	1531	0.703261	20	0.009187
ɣ	ɣ	0.007197	327	0.150207	12	0.005512
ts	ts	0.006746	519	0.238401	16	0.00735
e	e	0.006693	1482	0.680753	19	0.008728
ŋ	ŋ	0.004061	1424	0.654111	12	0.005512
ʔ	ʔ	0.003222	846	0.388608	10	0.004593

In [18]:
```python
# Rare segments
borrowability_df.loc[ borrowability_df.SEGBO_frequency_absolute <= 2 ][:10]
```

Out[18]:

	Segment	Borrowability	PHOIBLE_frequency_absolute	PHOIBLE_frequency_relative	SEGBO_frequency_absolute	SEGBO_frequency_relative
d̤ʒ̤	d̤ʒ̤	0.166743	1	0.000459	1	0.000459
ʕ̩	ʕ̩	0.166743	1	0.000459	1	0.000459
uə̯	uə̯	0.166743	1	0.000459	1	0.000459
n̪ˤ	n̪ˤ	0.166743	1	0.000459	1	0.000459
l̪ˤ	l̪ˤ	0.166743	1	0.000459	1	0.000459
ʊai	ʊai	0.166743	1	0.000459	1	0.000459
ɻ̤	ɻ̤	0.166743	1	0.000459	1	0.000459
n̪d̪z̪	n̪d̪z̪	0.166743	1	0.000459	1	0.000459
ðˠ	ðˠ	0.166743	1	0.000459	1	0.000459
ɸʷ	ɸʷ	0.166743	1	0.000459	1	0.000459

In [19]:
```python
borrowability_df.to_csv('model_1_borrowability.csv', index=False)
```

In [20]:
```python
borrowability_laplace_df = pd.DataFrame.from_dict(borrowability_scores_laplace).T.sort_values(
    by='Borrowability', ascending=False)
```

In [21]:
```python
# Frequently borrowed segments
borrowability_laplace_df.loc[ borrowability_laplace_df.SEGBO_frequency_absolute >= 10 ]
```

Out[21]:

	Segment	Borrowability	PHOIBLE_frequency_absolute	PHOIBLE_frequency_relative	SEGBO_frequency_absolute	SEGBO_frequency_relative
f	f	0.031307	969	0.445108	101	0.046394
p	p	0.021112	1896	0.870923	31	0.01424
g	g	0.01631	1256	0.576941	52	0.023886
b	b	0.013569	1386	0.636656	41	0.018833
z	z	0.013512	683	0.313734	38	0.017455
ʒ	ʒ	0.013032	332	0.152503	22	0.010106
d̠ʒ	d̠ʒ	0.011793	641	0.294442	32	0.014699
v	v	0.011674	618	0.283877	31	0.01424
d	d	0.011638	1098	0.504364	38	0.017455
x	x	0.010977	412	0.189251	22	0.010106
h	h	0.010674	1259	0.578319	34	0.015618
ʃ	ʃ	0.010637	783	0.359669	32	0.014699
l	l	0.009261	1583	0.727147	24	0.011024
r	r	0.008882	1100	0.505282	29	0.013321
t̠ʃ	t̠ʃ	0.008793	917	0.421222	28	0.012862
o	o	0.008587	1447	0.664676	25	0.011484
ɾ	ɾ	0.008301	616	0.282958	22	0.010106
s	s	0.007344	1532	0.703721	20	0.009187
ɣ	ɣ	0.007179	328	0.150666	12	0.005512
ts	ts	0.006738	520	0.238861	16	0.00735
e	e	0.006698	1483	0.681213	19	0.008728
ŋ	ŋ	0.004063	1425	0.654571	12	0.005512
ʔ	ʔ	0.003221	847	0.389068	10	0.004593

In [22]:

```
# Rare segments
borrowability_laplace_df.loc[ borrowability_laplace_df.SEGBO_frequency_absolute <= 2 ][:10]
```

Out[22]:

	Segment	Borrowability	PHOIBLE_frequency_absolute	PHOIBLE_frequency_relative	SEGBO_frequency_absolute	SEGBO_frequency_relative
d̤ʒ̤	d̤ʒ̤	0.08341	2	0.000919	1	0.000459
ʕ̩	ʕ̩	0.08341	2	0.000919	1	0.000459
uə̯	uə̯	0.08341	2	0.000919	1	0.000459
n̪ˤ	n̪ˤ	0.08341	2	0.000919	1	0.000459
l̪ˤ	l̪ˤ	0.08341	2	0.000919	1	0.000459
ʊai	ʊai	0.08341	2	0.000919	1	0.000459
ɹ̤	ɹ̤	0.08341	2	0.000919	1	0.000459
n̪d̪z̪	n̪d̪z̪	0.08341	2	0.000919	1	0.000459
ðˠ	ðˠ	0.08341	2	0.000919	1	0.000459
ɸʷ	ɸʷ	0.08341	2	0.000919	1	0.000459

In [23]:

```
borrowability_laplace_df.to_csv('model_1_borrowability_laplace.csv', index=False)
```

References

Eisen, Elad. 2019. "The Typology of Phonological Segment Borrowing." Masters thesis, Jerusalem, Israel: Hebrew University of Jerusalem.

Forkel, Robert, Johann-Mattis List, Simon J. Greenhill, Christoph Rzymski, Sebastian Bank, Michael Cysouw, Harald Hammarström, Martin Haspelmath, Gereon A. Kaiping, and Russell D. Gray. 2018. "Cross-Linguistic Data Formats, Advancing Data Sharing and Re-Use in Comparative Linguistics." Scientific Data 5: 180205.

Grossman, Eitan, Elad Eisen, Dmitry Nikolaev, and Steven Moran. 2020. "SegBo: A Database of Borrowed Sounds in the World's Language." In Proceedings of the 12th Language Resources and Evaluation Conference, 5316–22.

Moran, Steven, and Daniel McCloy, eds. 2019. PHOIBLE 2.0. Jena: Max Planck Institute for the Science of Human History. https://doi.org/10.5281/zenodo.2562766.