Extracting Phoneme Shift Rules with a Multi-task Transformer

Automated Learning with Noisy Data

AGENDA

1. Motivation + Goals
2. Data
3. Transformer Model
4. Results
5. Conclusions
6. Future Work

MOTIVATION + GOALS

MOTIVATION

- Reliable data can be difficult to obtain, especially for linguistics tasks.
    - Different dialects, linguistic drift, dependence on experts / native speakers, etc.
- Neural nets have been shown to excel at tasks using noisy, diverse datasets.
- Transformers have shown excellent ability in language translation tasks.

GOALS

- The goal is to use a transformer neural network to analyze a language family based on a messy web-scraped list of cognate words.
    - I want to use a small transformer model for cognate translation, and add a second head to the model that predicts the specific phoneme mutation rules for that translated word pair.
    - Then I can compare these mutation rules and find the magnitude and direction of each language's phonetic drift over time, relative to all the other languages in the family tree.

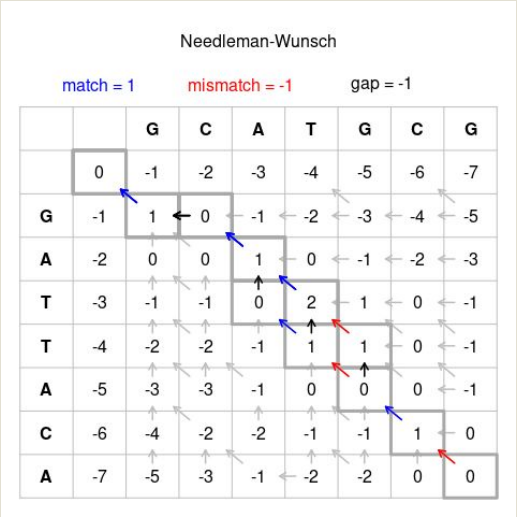
DATA + PREPROCESSING

- I began with a table of cognates that I found on Reddit (6,313 lines, Sparse)
  - Vulgar Latin → (Portuguese, Spanish, Catalan, French, Italian, Romanian)
- I found an open source model on hugging face that can do grapheme to phoneme conversion for all the relevant languages

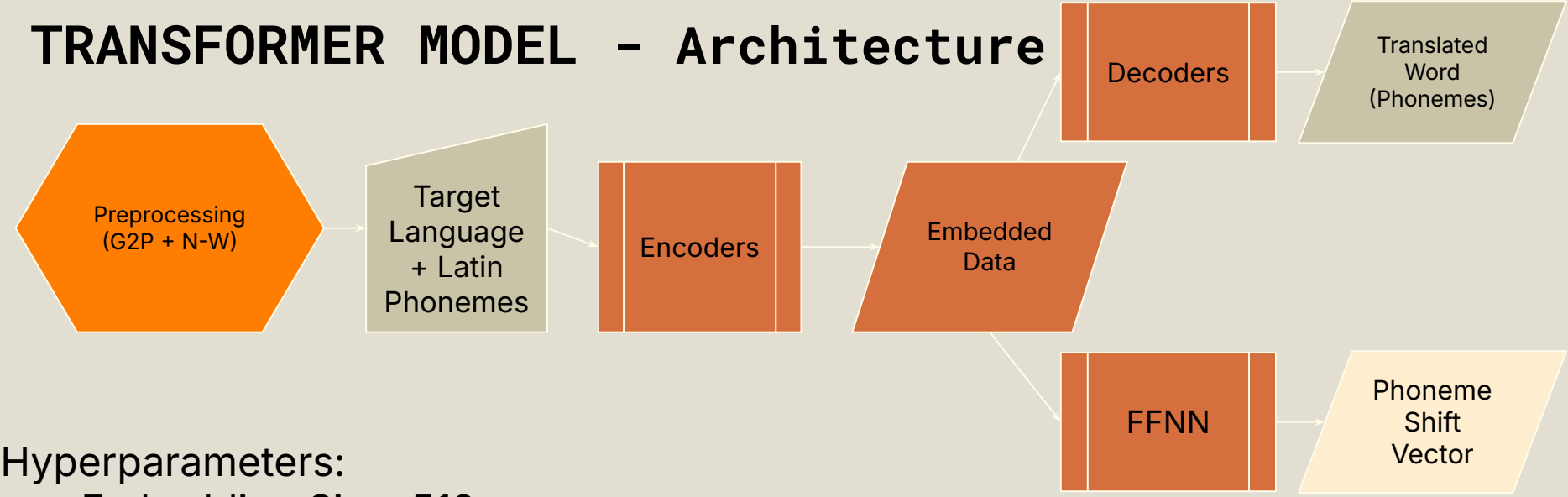
Language	Err. rate	Word err. rate
Lat-eccl	2.7%	10.6%
Spa	0.29%	1.6%
Fra	0.5%	2.2%
Ita	2.7%	19%
Rom	0.4%	3%

- I cut Portuguese and Catalan based on G2P word error rate >20% (6,253 lines)
- **SP**: 4,204 **FR**: 3,547 **ITA**: 4,394 **ROM**: 1,674 **TOTAL:** 13,819
- Final split [Train: 5,178 Validation: 575 Test: 500]

- I used Needleman-Wunsch for automatic string alignment of the phoneme strings generated by the G2P model
- `Bonu,bueno,bon,buono,bun,` `bɔ̃nu,bweno,bɔ̃,bwɔ̃no,bun,`
- `"["u->o", "\u02d0->e", "\u0254->w"]"`



TRANSFORMER MODEL - Architecture

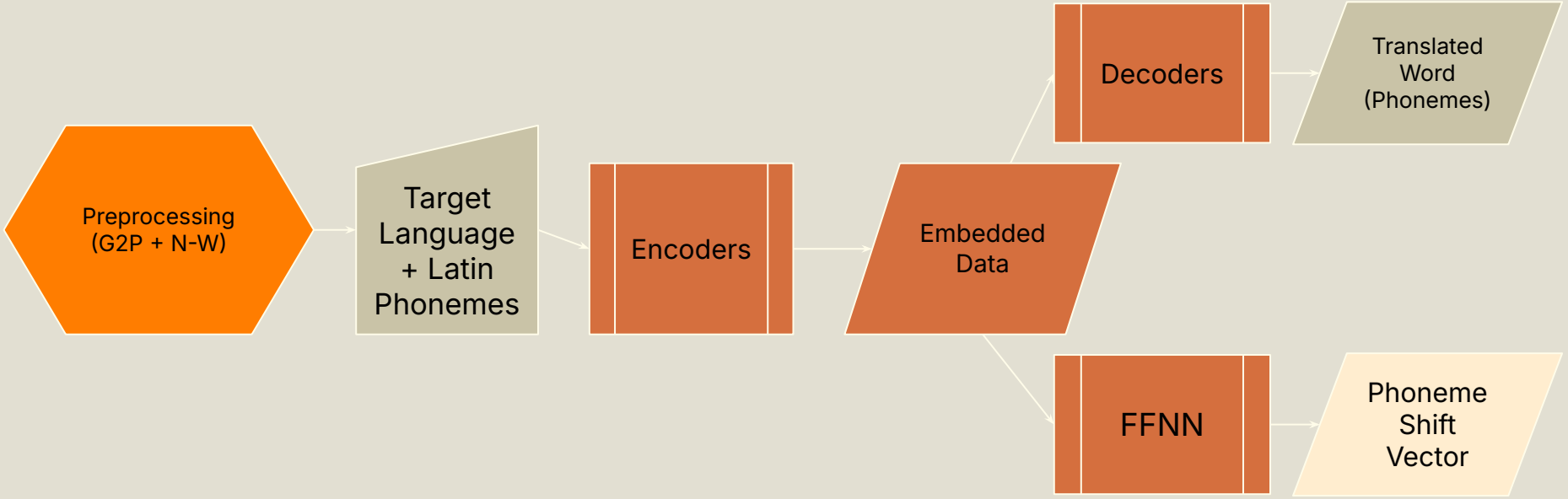


Hyperparameters:

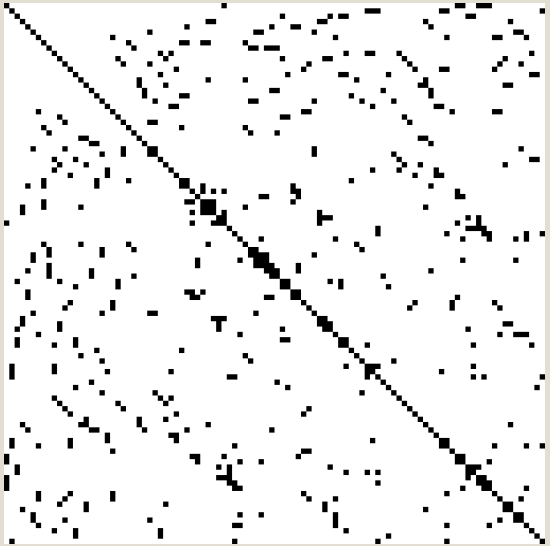
- Embedding Size: 512
- Encoder Size: 1024
- Encoders / Decoders: 3
- Attention Heads: 4
- Total Parameters: 16.5M (~65MB)

<u>Feed Forward Neural Net</u>

- Embedding (512) → 256
- ReLU + Dropout
- 256 → PSV (1068)


Preprocessing
(G2P + N-W)
Target
Language
+ Latin
Phonemes
Encoders
Embedded
Data
Decoders
Translated
Word
(Phonemes)
FFNN
Phoneme
Shift
Vector

TRANSFORMER MODEL - PSV



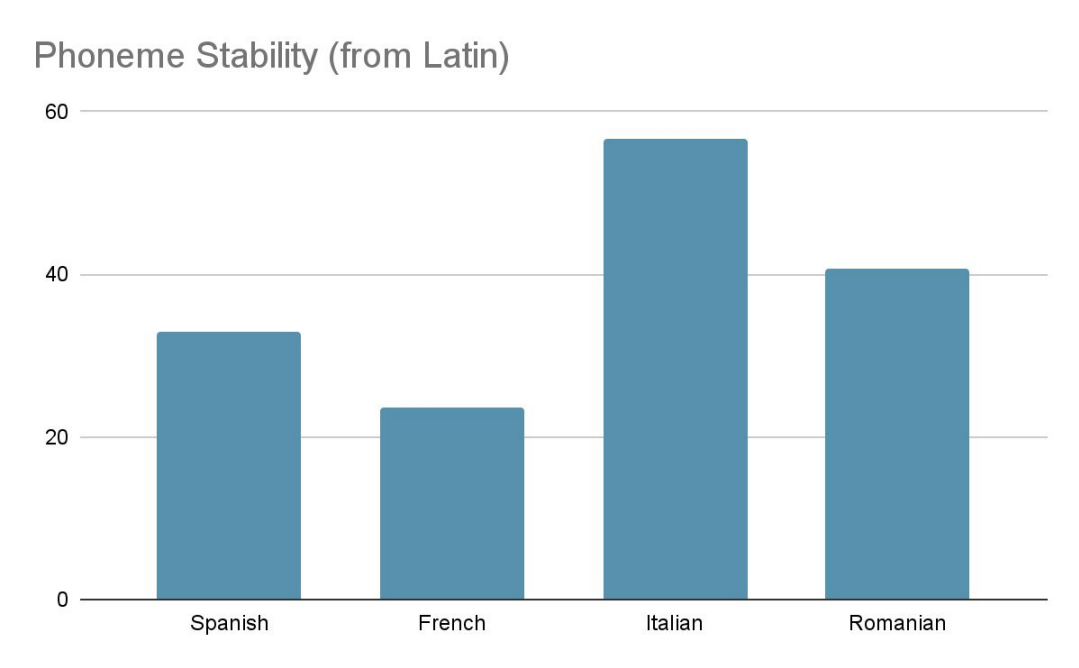
- There are 57 phonemes in the dataset.
  - This gives a 57×57 matrix of possible shifts that may occur to an input word, with 3249 entries total.
  - Most of these shifts never occur. In the dataset that I used, only 1068 distinct phoneme shifts occurred.
  - 67.1% of entries should always be 0.
  - In order to make the task easier for the model to learn, it is only required to predict distinct shifts / mutations that may occur. This helps to keep the model small and fast.
  - This 1068 vector from the model is what I refer to as the Phoneme Shift Vector (PSV).
  - The 57×57 sparse matrix is referred to as the Phoneme Shift Matrix (PSM).
- During training the loss for the model is weighted based on how frequently the predicted phoneme shift occurs throughout the dataset. More common phoneme shifts are assigned higher loss values. By this method, the model is taught to prioritize learning the rules for the most common phoneme shifts in the dataset.
- We do not track identity in the PSV

TRANSFORMER MODEL - Training

- After initial experimentation to find a starting point for the model I ran 31 training runs, systematically experimenting with hyperparameters of the model.
- I used a holdout validation set for early stopping and a reserved test set for final testing on the trained model.

IDX	EMBED_SIZE	NHEAD	ENC_LAYERS	DEC_LAYERS	DIM_FF	DROPOUT	WEIGHT_DECAY	11	LR	BATCH_SIZE	EPOCHS	PSV_LOSS_LAMBDA	pos_weight	22	Total Loss	Translation Loss	Rules (PSM) Loss	PSM F1 Score (Micro)	PSM F1 Score (Macro)
1	256	4	3	3	512	0.2	0		0.0002	32	50	1	100		1.2434	0.9739	0.2695	37.6%	6.7%
2	256	4	3	3	512	0.2	0		0.0002	32	85	0.25	365		1.777	0.9703	0.8067	26.1%	5.3%
3	256	4	3	3	512	0.2	0		0.0002	32	95	0.1	365		1.7262	0.9936	0.7326	25.7%	5.3%
4	256	4	3	3	512	0.2	0		0.0002	32	79	1	100		1.2549	0.9793	0.2756	37.4%	6.2%
5	256	4	3	3	512	0.2	0		0.0002	32	86	1	50		1.0898	0.9352	0.1546	45.6%	7.0%
6	256	4	3	3	512	0.2	0		0.0002	32	90	1	10		1.0375	0.9868	0.0507	60.2%	6.1%
7	256	4	3	3	512	0.2	0		0.0002	32	93	1	2		0.9741	0.9564	0.0177	49.9%	2.9%
8	256	4	3	3	512	0.2	0		0.0002	32	104	1	5		1.0128	0.9806	0.0322	60.7%	5.0%
9	512	4	3	3	1024	0.2	0		0.0002	64	60	1	10		1.0747	1.028	0.0467	58.7%	6.4%
10	512	4	3	3	1024	0.2	0		0.0001	32	76	1	10		1.0702	1.0201	0.05	59.7%	6.1%
11	256	8	6	6	512	0.2	0		0.0002	32	62	1	10		1.1273	1.0753	0.052	57.1%	4.8%
12	1024	4	3	3	2048	0.2	0		0.0002	32	242	1	10		1.7954	1.7274	0.0679	52.4%	3.3%
13	1024	4	3	3	2048	0.2	0		0.0001	32	65	1	10		1.1337	1.0855	0.0482	61.3%	6.5%
14	512	4	3	3	1024	0.2	0		0.0002	32	76	1	5		1.1182	1.0863	0.0319	61.5%	5.6%
15	512	8	3	3	1024	0.2	0		0.0001	32	59	1	10		1.1212	1.0734	0.0478	59.9%	6.6%
16	512	4	3	3	1024	0.2	1.00E-04		0.0002	32	174	1	10		0.7494	0.7001	0.0493	60.2%	3.9%
17	512	4	3	3	1024	0.2	1.00E-05		0.0002	32	67	1	10		1.0399	0.9966	0.0433	60.4%	5.7%
18	512	4	3	3	1024	0.2	1.00E-04		0.0002	32	169	2	10		0.7887	0.745	0.0437	61.9%	5.6%
19	512	4	3	3	1024	0.2	1.00E-04		0.0002	32	178	4	10		0.7782	0.7387	0.0395	64.6%	6.9%
20	512	4	3	3	1024	0.2	1.00E-04		0.0002	32	133	8	10		0.7883	0.7459	0.0425	64.1%	7.3%
21	512	4	3	3	1024	0.2	1.00E-04		0.0002	32	55	32	10		1.0578	1.0132	0.0446	59.6%	6.7%
22	512	4	3	3	1024	0.2	1.00E-04		0.0002	32	84	16	10		0.9612	0.9144	0.0469	62.1%	6.7%
23	512	4	3	3	1024	0.2	2.00E-04		0.0002	32	142	10	10		0.6539	0.6135	0.0405	66.5%	7.7%
24	512	4	3	3	1024	0.2	4.00E-04		0.0002	32	139	10	10		0.6162	0.5768	0.0394	65.1%	7.4%
25	512	4	3	3	1024	0.1	2.00E-04		0.0002	32	78	10	10		0.7158	0.6751	0.0407	62.8%	7.0%
26	512	4	3	3	1024	0.3	2.00E-04		0.0002	32	177	10	10		0.608	0.569	0.0389	66.3%	7.6%
27	512	4	3	3	1024	0.4	2.00E-04		0.0002	32	209	10	10		0.5795	0.5391	0.0404	66.7%	7.7%
28	512	4	3	3	1024	0.5	2.00E-04		0.0002	32	302	10	10		0.5697	0.5275	0.0422	65.3%	6.8%
29	512	4	3	3	1024	0.35	2.00E-04		0.00025	32	205	10	10		0.557	0.517	0.0401	66.9%	8.1%
30	1024	4	3	3	2048	0.35	2.00E-04		0.0001	32	88	10	10		1.0289	0.9866	0.0423	61.4%	6.3%
31	1024	8	6	6	2048	0.35	2.00E-04		0.00001	32	113	10	10		1.1089	1.0629	0.046	59.1%	4.8%

RESULTS - Statistical Analysis



$$Stability = \frac{Count\ of\ Exact\ Matches}{Total\ Length\ of\ Alignment}$$

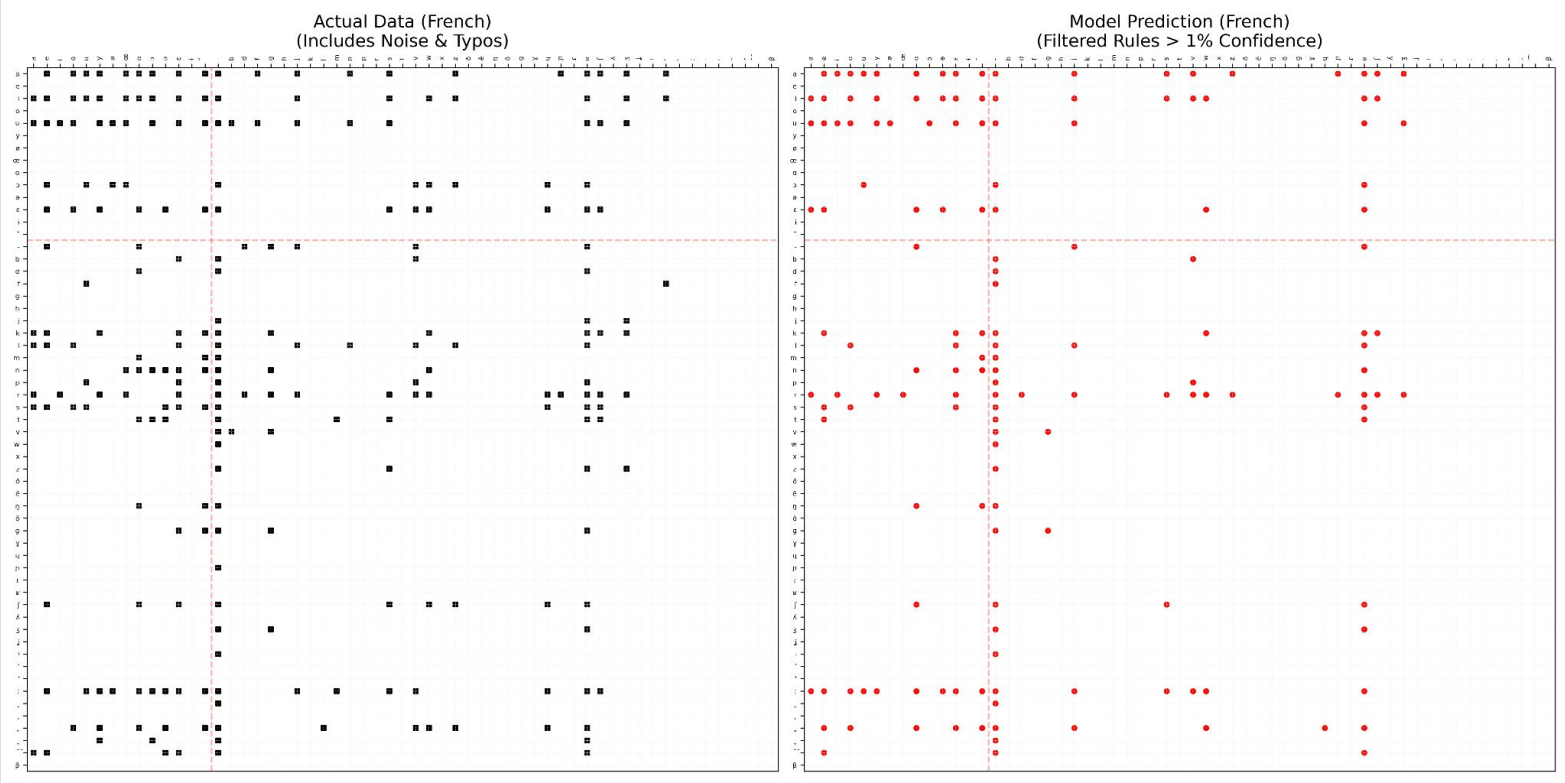
Italian = 56.7%

Romanian = 40.7%

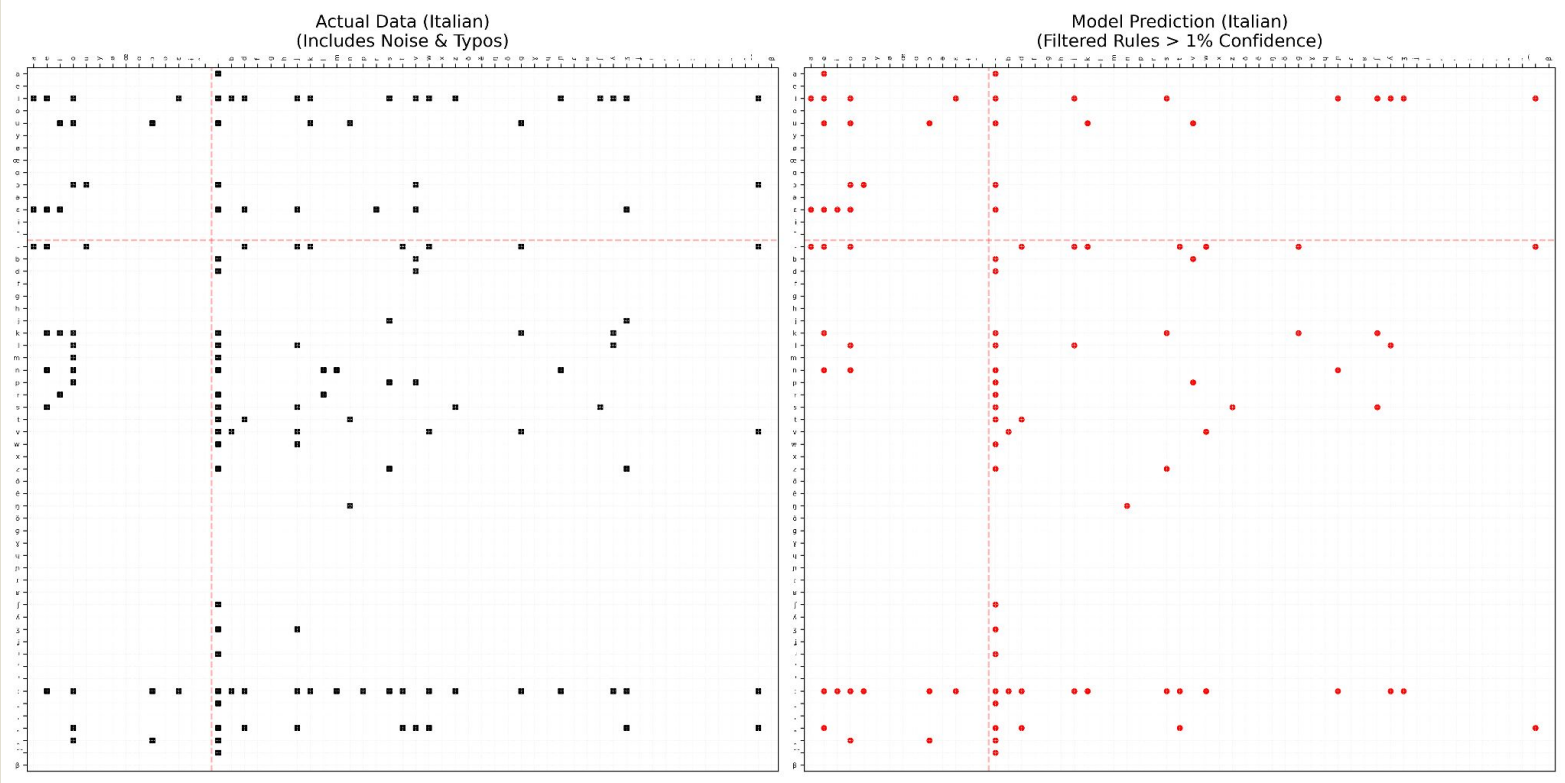
Spanish = 33.0%

French = 23.7%

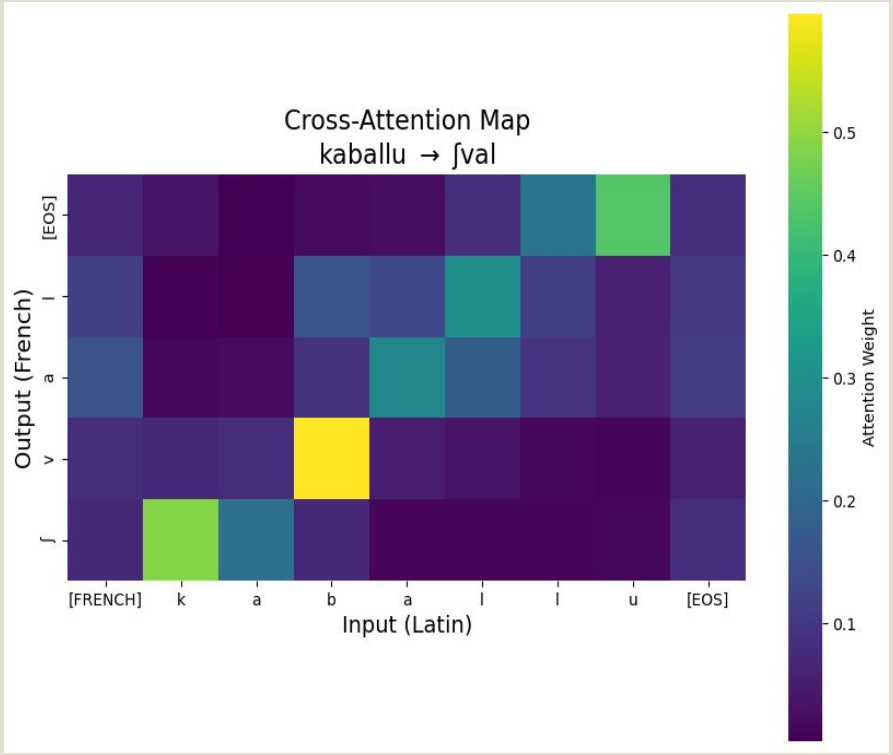
RESULTS - Sparse Matrix Reproduction

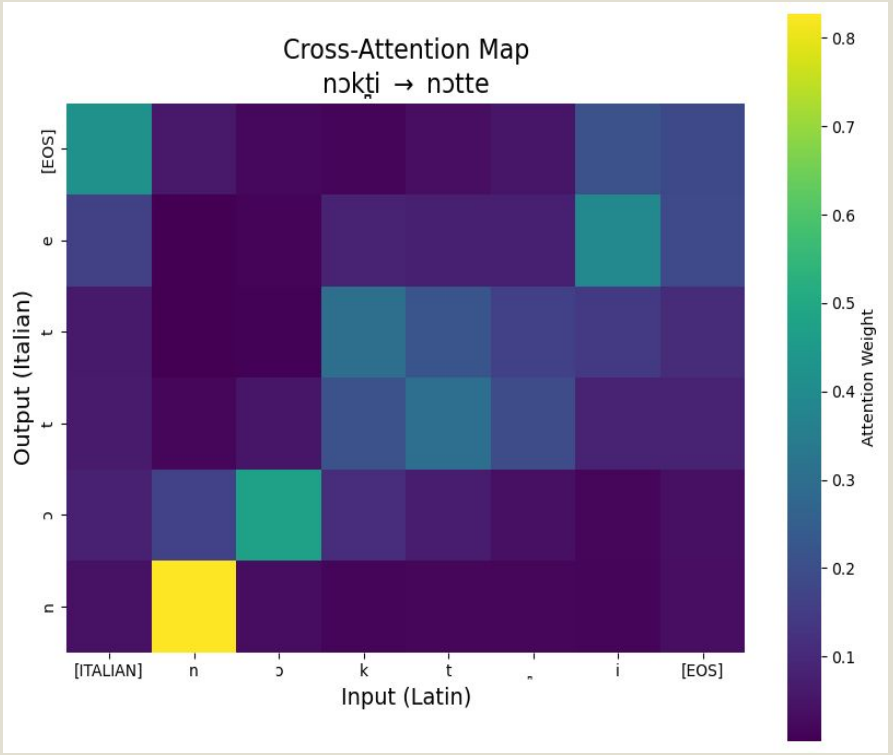


RESULTS - Sparse Matrix Reproduction

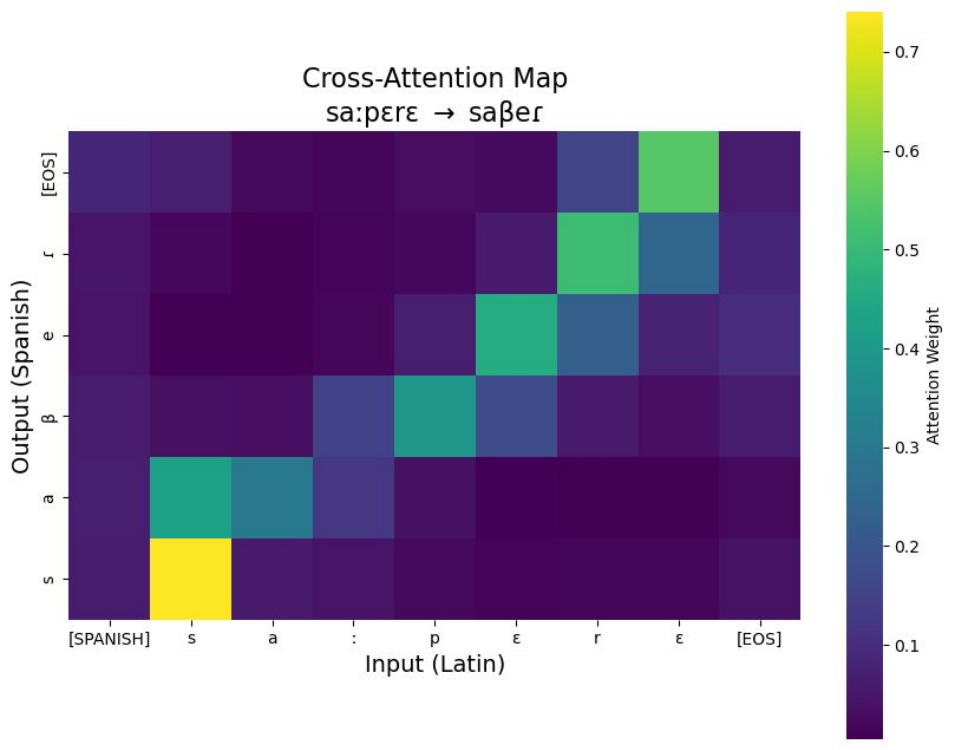


RESULTS - Attention





RESULTS - Attention + Context



<u>Testing predictive capability for p→b mutation between vowels (Intervocalic Lenition)</u>

- Start of Word (p...): 39 samples
- Middle of Word (...p...): 68 samples

Avg. Confidence of p→b mutation (Spanish)

- Start: 1.01%
- Middle: 1.81%

Avg. Confidence of p→b mutation (Romanian)

- Start: 0.09%
- Middle: 0.14%

RESULTS - Common Consonant Mutations

French

01. r→ʁ (Freq: 0.3472)
02. s→e (Freq: 0.1108)
03. r→j (Freq: 0.0947)
04. n→ɑ (Freq: 0.0930)
05. n→ɛ (Freq: 0.0621)
06. k→ʃ (Freq: 0.0435)
07. m→˜ (Freq: 0.0434)
08. n→˜ (Freq: 0.0345)
09. r→d (Freq: 0.0337)
10. r→ʒ (Freq: 0.0327)

Spanish

01. r→r (Freq: 0.3599)
02. r→e (Freq: 0.1865)
03. k→ɣ (Freq: 0.1348)
04. k→e (Freq: 0.1059)
05. →e (Freq: 0.0953)
06. l→ʎ (Freq: 0.0857)
07. →' (Freq: 0.0846)
08. r→o (Freq: 0.0646)
09. v→β (Freq: 0.0639)
10. v→b (Freq: 0.0612)

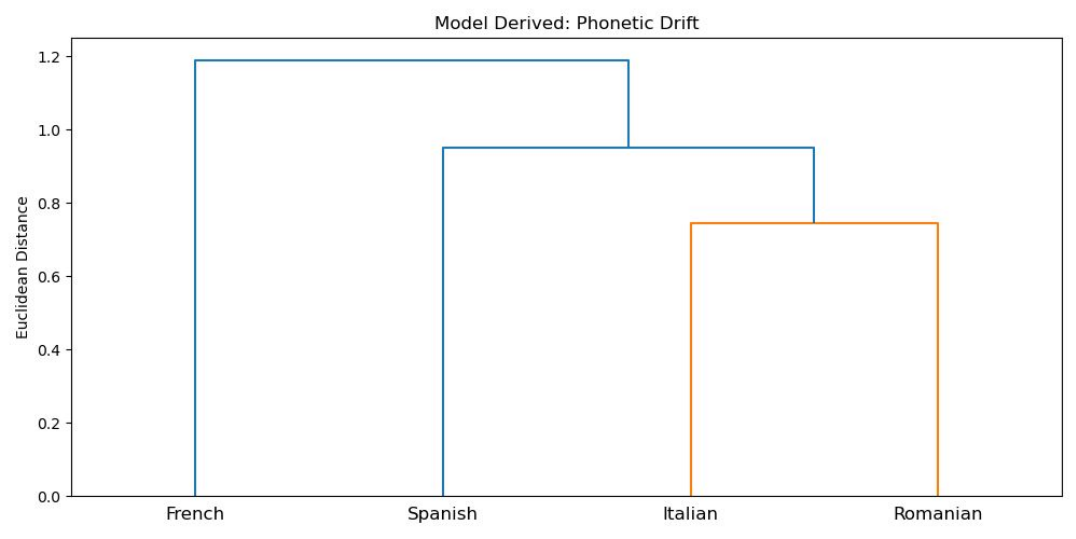
Italian

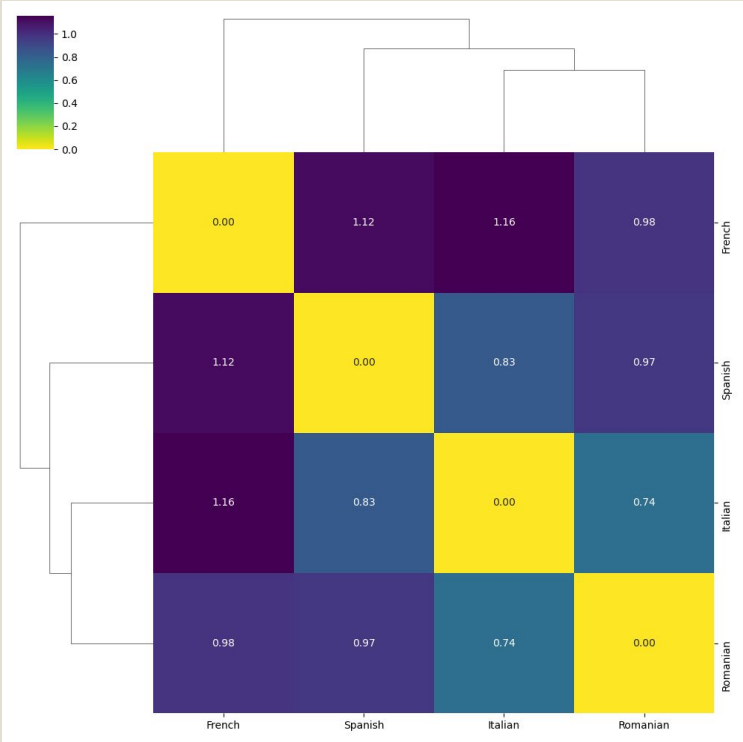
01. l→j (Freq: 0.1097)
02. k→g (Freq: 0.0742)
03. s→z (Freq: 0.0596)
04. b→v (Freq: 0.0483)
05. l→ʎ (Freq: 0.0387)
06. k→e (Freq: 0.0371)
07. n→ɲ (Freq: 0.0350)
08. →d (Freq: 0.0334)
09. ŋ→n (Freq: 0.0323)
10. →j (Freq: 0.0323)

Romanian

01. l→r (Freq: 0.0292)
02. z→o (Freq: 0.0280)
03. l→e (Freq: 0.0273)
04. →j (Freq: 0.0263)
05. k→p (Freq: 0.0233)
06. l→, (Freq: 0.0220)
07. ŋ→n (Freq: 0.0216)
08. v→b (Freq: 0.0203)
09. z→s (Freq: 0.0188)
10. s→ʃ (Freq: 0.0179)

RESULTS - Dendrogram + Heatmap

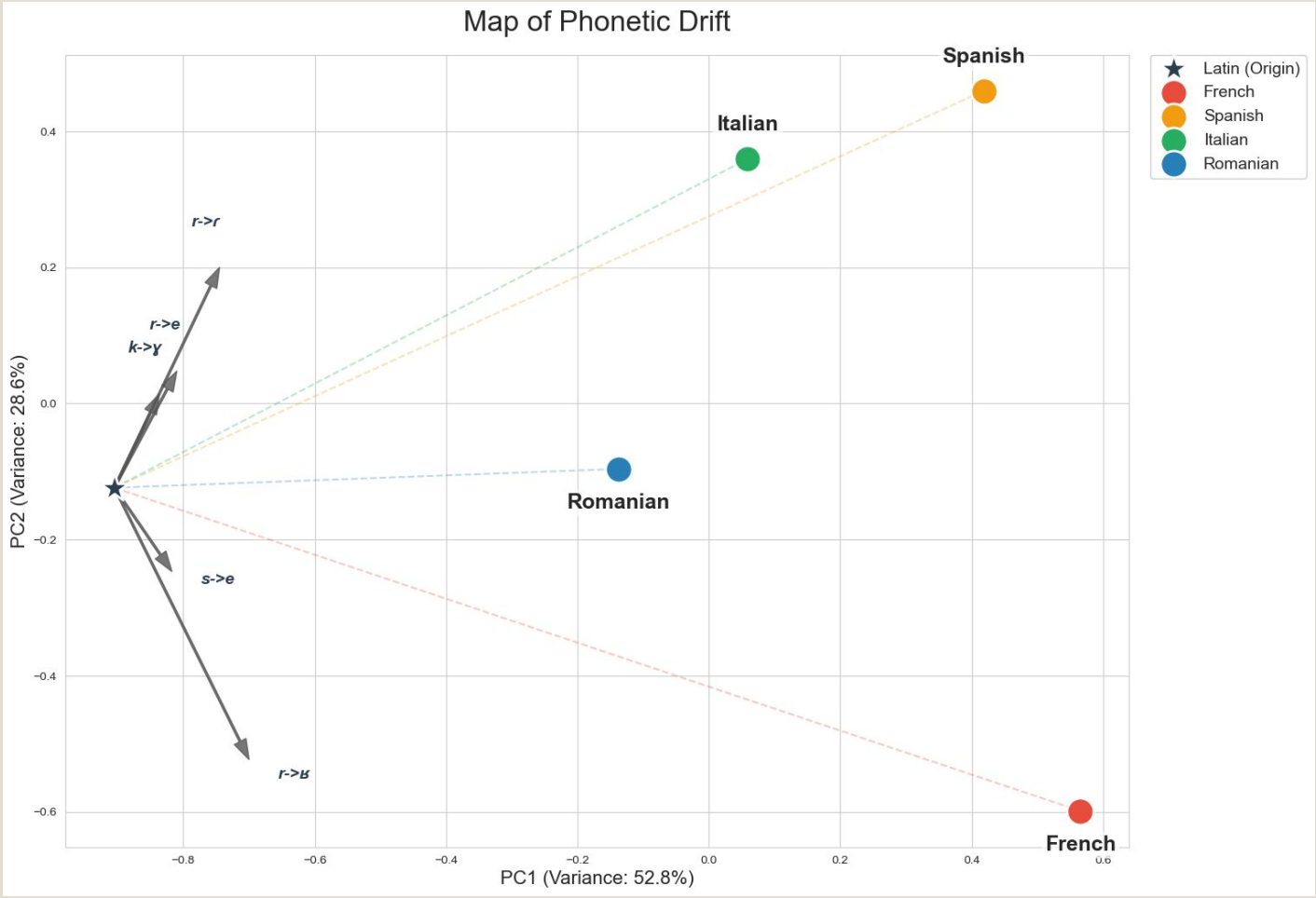




2D PCA

Variance Explained

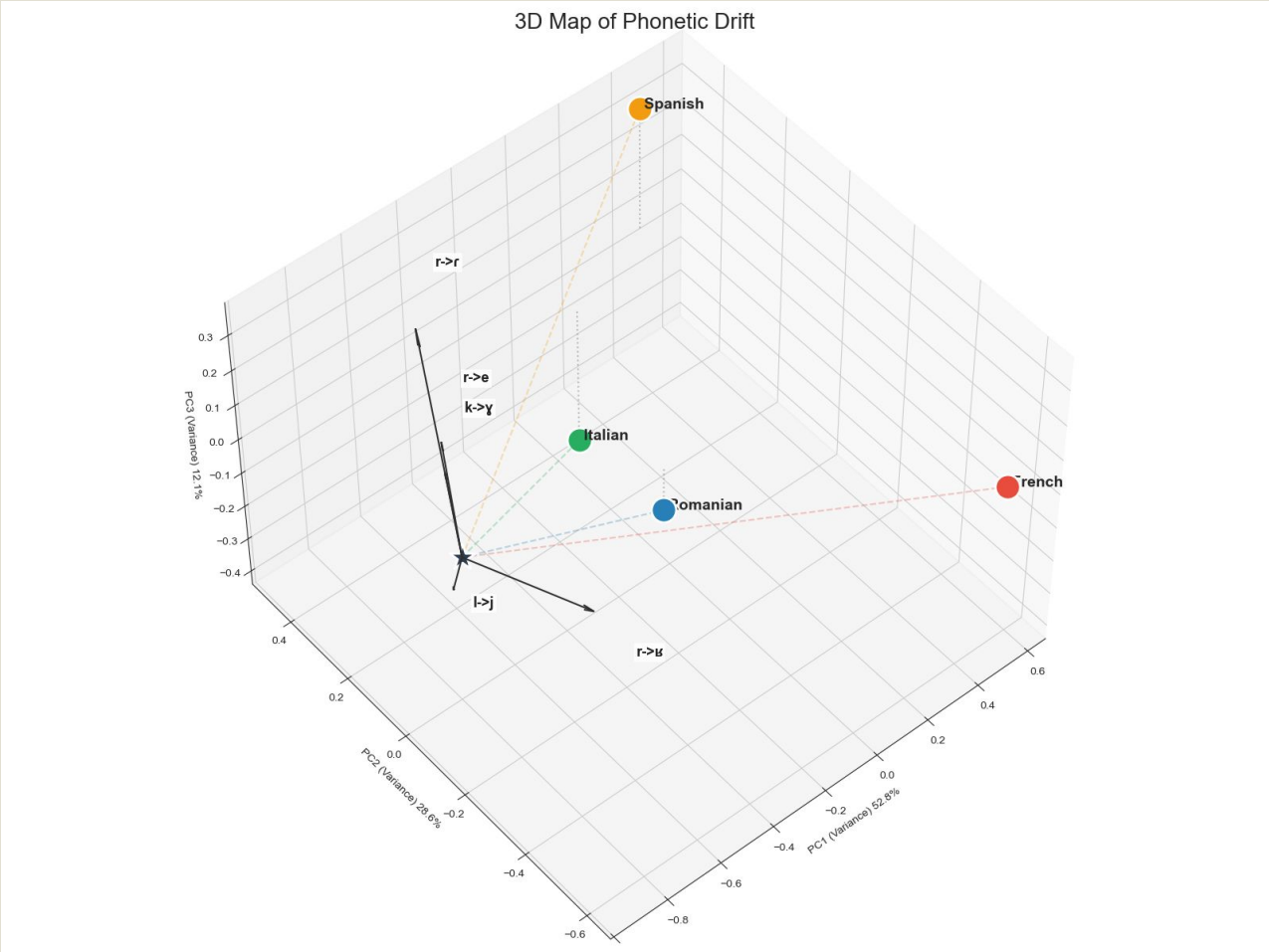
81.4%



3D PCA

Variance Explained

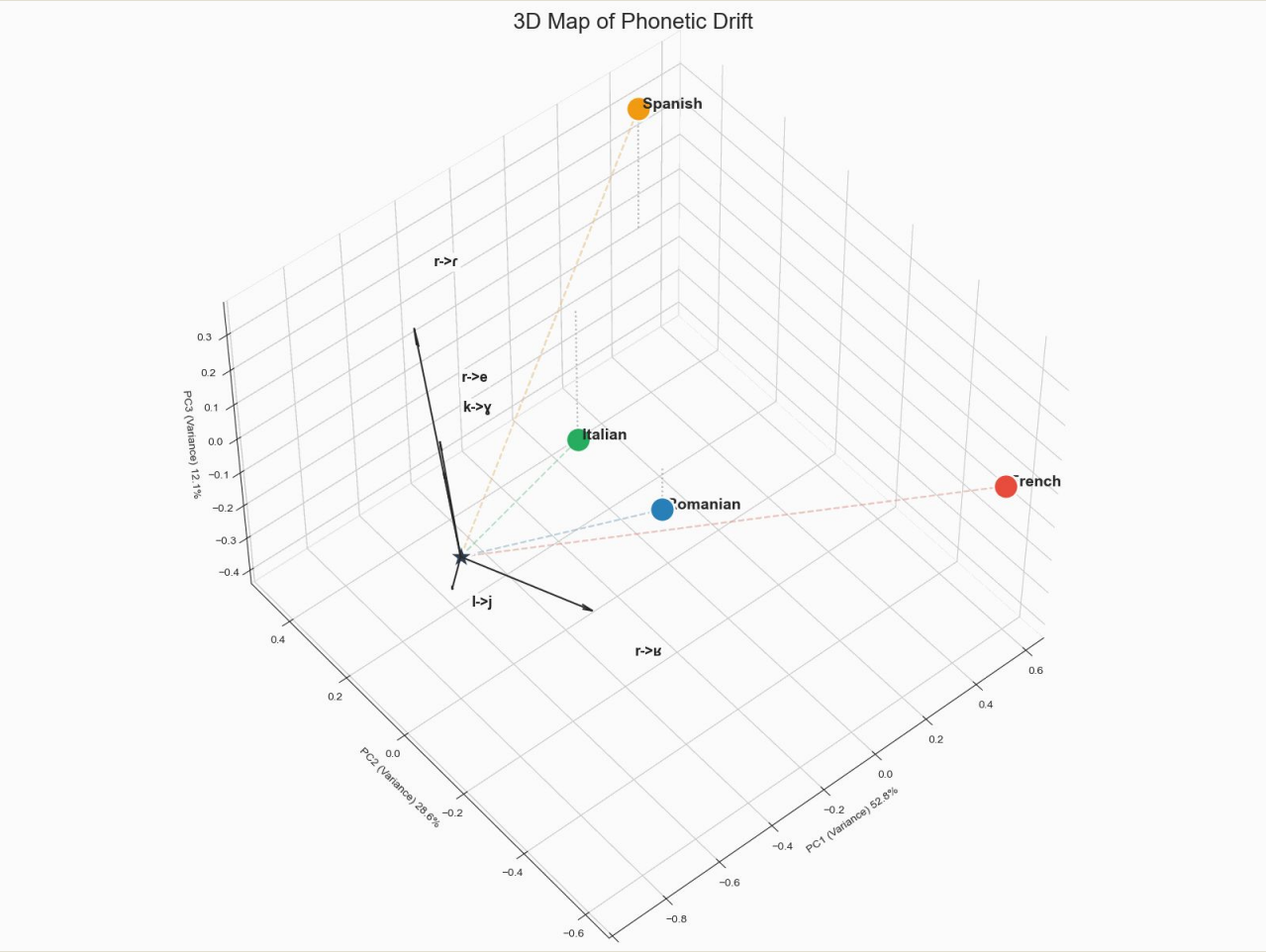
93.5%



3D PCA

Variance Explained

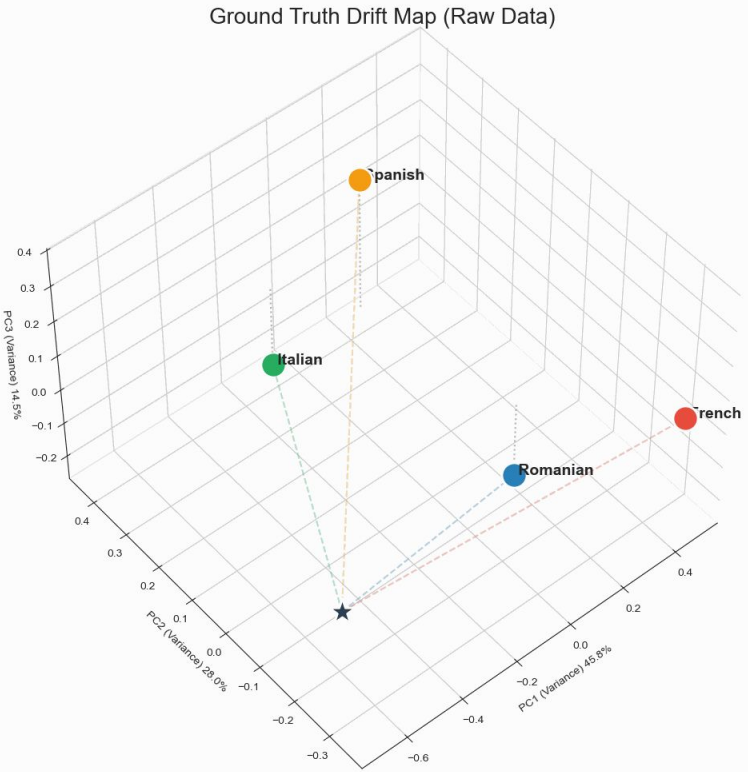
93.5%



3D PCA (Raw Data)

Variance Explained

88.3%



CONCLUSIONS

We have 2 things to consider:

- G2P + N-W
- Transformer model

A lot of the work in this project can be attributed to the pre-processing steps. The neural network is just learning the data that we feed it, but the transformer architecture does give us new ways to understand the data.

- Using cross-attention heatmaps we can gain insights about how languages evolve over time, what parts of the words influence each other, and overall trends in phoneme mutation in a language family.
- The transformer model naturally understands context, demonstrating an ability to apply real-world patterns that would be expected when queried with data from an unseen test set.
- The transformer model accurately reproduces the training data, while effectively filtering out noise.

FUTURE WORK

- **Consult with experts to verify results**
- **Assess utility of current work**
- **Better data for testing and training**
- Add GUI
    - Enter made up latin word (Plumbus) and select a target language
    - Auto-convert to phonemes with G2P model
    - Feed phoneme representation into transformer model
    - Output predicted translation, attention heatmap, and top x shifts / mutations predicted in the word by the PSV
- Expand Scope
    - More languages
    - Currently it's all rooted in latin. I could try to make a model that converts universally.
    - Currently it's just trained on cognates. It would be interesting to train on multiple entire dictionaries.
    - Scrape the web for more data.
- Agentic Framework
    - The training process was thorough but very systematic, it could probably be automated with AIDE (AI Driven Exploration) or another similar agentic script iteration tool.
- Add accents to existing text to speech programs?

FIN