Mockingbird at the 2022 SIGTYP Shared Task

Two Types of Models For the Prediction of Cognate Reflexes

Christo Kirov Richard Sproat Alexander Gutkin

Google Research

Introduction

The discovery of cognate correspondences is an old problem that goes back to at least the establishment of Indo-European by William Jones in 1786.

The SIGTYP 2022 Shared Task involves the recovery of missing cognate reflexes given a subset of their neighbors.

	English	German	Dutch
Set 1	dream	Traum	droom
Set 1	??beam??	Baum	boom

Experimental Conditions

- Ten language family sets for development.
- Ten “surprise” language families for final testing.
- ~100s of cognate sets available for training per family.
- Versions of each dataset at different levels of sparsity: [10%, 20%, 30%, 40%, 50%] of the cognates ablated.
- Evaluation metrics included: Edit Distance, B-Cubed F-Score, and BLEU

Two (Very) Different Approaches

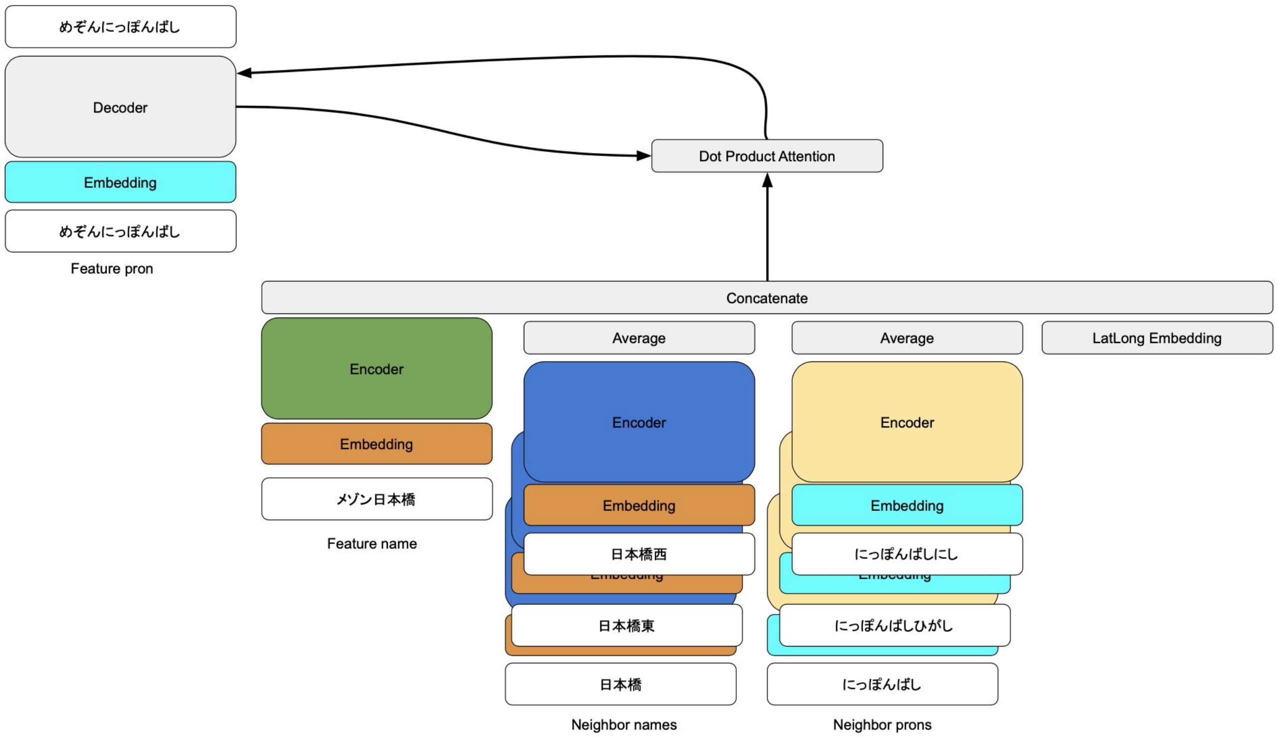
1. Extend Seq2Seq transformers (Vaswani 2017) to encode all available cognates in parallel.
   a. Able to capture arbitrary contextual cues.
   b. Complex models with many parameters.
2. Repurpose image inpainting CNN.
   a. Treat a complete set of cognates as an “image” with some rows of “pixels” missing.
   b. Ability to capture context limited by kernel size.
   c. Simple enough model to train/infer even on CPU.

Neighborhood Model

- Combine information from multiple parallel transformer encoders.
- Originally designed to determine pronunciations for Japanese *kanji* spellings of entities based on geo-proximity, adapted as follows:
  - Neighbors: Features in proximity ⇒ Other languages in cognate set
  - Kanji spellings ⇒ Language "codes"
  - Katakana pronunciations ⇒ IPA
- Submitted 3 variants (identical but trained for 25k, 35k, or 100k steps).
- Open source version based on Tensorflow Lingvo.*

* https://github.com/google-research/google-research/tree/master/cognate_inpaint_neighbors/neighbors

Neighborhood Model



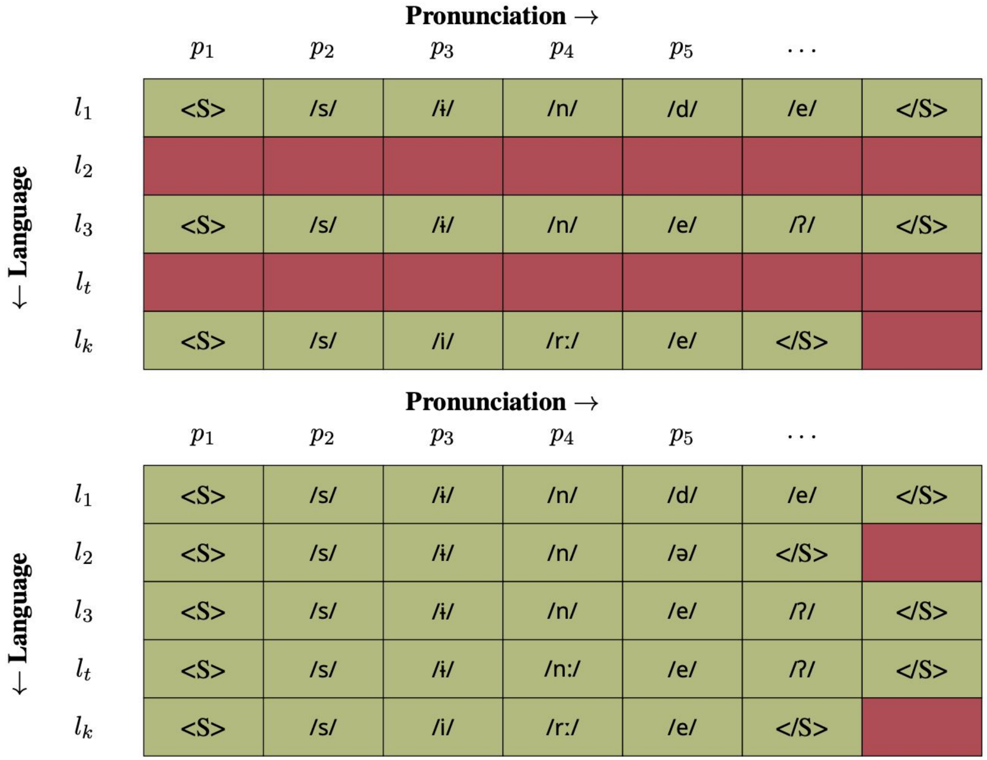
Neighborhood Model - Data Augmentation

- Not enough data to train.
- For each cognate neighborhood, generate 500 sub-neighborhoods by randomly removing a subset of existing cognates.
- For each (sub)neighborhood, randomly generate 10 new neighborhoods by using a pair unigram model trained on Levenshtein alignments between existing cognate pairs to hallucinate extra cognates.
- For ListSampleSize, 300 training samples becomes 4.2 million.

Inpainting Cognate “Images”

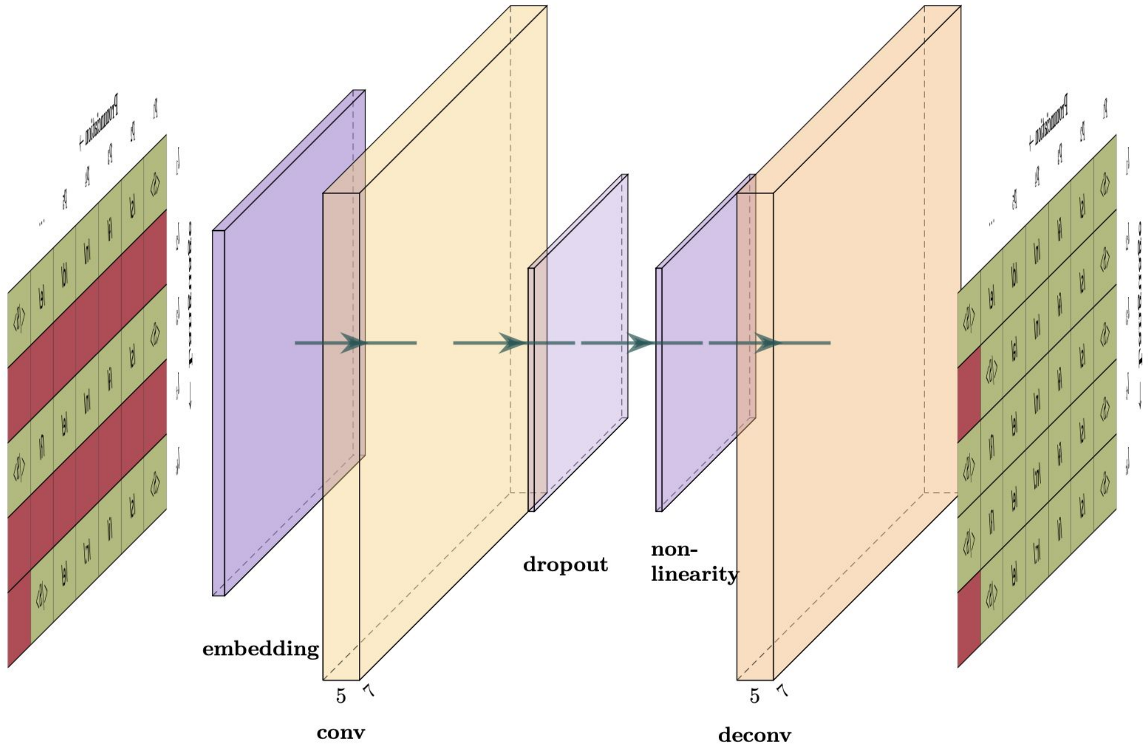
- Inspired by NVIDIA inpainting architecture (Liu et al., 2018).
- Simple CNN.
- Open-source: TF recipe.*

$$x' = \begin{cases} W^T(X \odot M)\frac{\mathrm{sum}(1)}{\mathrm{sum}(M)}, & \text{if } \mathrm{sum}(M) \geq 1 \\ 0, & \text{otherwise} \end{cases}$$



* https://github.com/google-research/google-research/tree/master/cognate_inpaint_neighbors/inpaint

Inpainting CNN Model



Inpainting - Low Resource Strategies

- No data hallucination, BUT:
- Split each set of training data into 10 random 80/20 train/dev sets.
  - Each individual dev set is small/biased, but direction of the bias varies.
- During each training step, drop a random subset of existing cognates (input dropout).
- Tune a model for each split using Vizier - a black-box optimizer.*
  - kernel width, dropout level, nonlinearity (tanh vs relu)
- For inference, ensemble all 10 tuned models via majority vote.

*https://cloud.google.com/ai-platform/optimizer/docs/overview

Results (Edit Distance - Lower is Better)

	Baseline	Inpainting	Neighbors (20k)
Dev Total	1.34 / 1.75	**1.05 / 1.40**	1.25 / 1.60
davletshinaztecan	2.07 / 2.09	1.87 / 1.69	2.04 / 2.29 (2.21 100k)
felekesemitic	1.46 / 2.90	1.29 / 2.85	1.68 / **2.33 (2.19 100k)**
hantganbangime	1.31 / 1.98	1.12 / 1.38	1.28 / 1.65
hattorijaponic	0.91 / 1.50	0.71 / 1.33	0.94 / 1.66 (1.50 100k)
listsamplesize	3.34 / 3.68	2.35 / 2.43	2.80 / 2.72
backstromnorthernpakistan	0.89 / 0.97	0.60 / 0.73	0.83 / 1.00 (0.93 100k)
mannburmish	1.98 / 2.33	1.55 / 2.02	1.74 / 2.41
castrosui	0.16 / 0.39	0.14 / 0.29	0.16 / 0.32
allenbai	0.72 / 0.76	0.55 / 0.64	0.58 / 0.78
abrahammonpa	0.55 / 0.94	0.34 / 0.66	0.47 / 0.87

Results (Edit Distance - Lower is Better)

	Baseline	Inpainting	Neighbors (20k)
Surprise Total	1.21 / 1.89	**0.92 / 1.42**	1.02 / 1.55
beidazihui	1.10 / 1.63	0.50 / 0.48	**0.48** / **0.45**
hillburmish	1.18 / 2.10	1.06 / 1.64	1.13 / 2.66 (1.80 100k)
bodtkhobwa	0.49 / 0.63	0.39 / 0.53	**0.25** / 0.56
bantubvd	1.12 / 1.99	0.89 / 1.53	1.01 / **1.29**
bremerberta	1.72 / 2.49	1.16 / 1.58	1.35 / 1.99 (1.85 30k)
deepadungpalaung	1.07 / 1.63	0.55 / 1.23	0.73 / 1.39
luangthongkumkaren	0.38 / 0.66	0.36 / 0.55	**0.26** / 0.56
birchallchapacuran	1.63 / 3.17	1.57 / 2.81	2.04 / **2.80**
wangbai	0.62 / 1/02	0.49 / 0.97	**0.48** / 1.05
kesslersignificance	2.77 / 4.06	2.23 / 2.85	2.49 / **2.77**

Concluding Remarks

- Significant variance across language families.
- Inpainting model has the best metrics on average, but loses out in several language families to the Neighborhood Transformer.
    - Smaller model paired with ensembling helps low-resource generalization, but can’t capture complex relationships.
- Neighborhood model shows particularly good sparse condition performance on the Semitic family, which has unique morphological characteristics (“templatic”).
    - Larger model can capture long-distance contextual cues, but may be more prone to overfitting
- Might be able to improve performance by trading low-resource strategies.

Thank You!