Lemmas Matter, But Not Like That: Predictors of Lemma-Based Generalization in Morphological Inflection

Sarah Payne and **Jordan Kodner**
Department of Linguistics & Institute for Advanced Computational Science
Stony Brook University, Stony Brook, NY, USA
{first.last}@stonybrook.edu

Abstract

Recent work has shown that overlap – whether a given lemma or feature set is attested independently in train – drives model performance on morphological inflection tasks. The impact of lemma overlap, however, is debated, with accuracy drops from 0% to 30% reported between seen and unseen test lemmas. In this paper, we introduce a novel splitting algorithm designed to investigate predictors of accuracy on seen and unseen lemmas. We find only an 11% average drop from seen to unseen test lemmas but show that the number of lemmas in train has a much stronger effect on accuracy on unseen than seen lemmas. We also show that the previously reported 30% drop is inflated due to the introduction of a near-30% drop in the number of training lemmas from the original splits to the novel splits. These results help us better understand the factors affecting morphological generalization by neural models.

1 Introduction

Morphological inflection – the task of producing an inflected form given a lemma and a set of morphosyntactic features – is a fundamental problem in NLP, with both practical and scientific implications. On the practical side, morphological inflection is a crucial tool for pipelining, particularly for low-resourced and highly-inflected languages for which end-to-end systems remain impractical or infeasible (Bender, 2009; Oflazer and Saraçlar, 2018; Wiemerslage et al., 2022; Arnett and Bergen, 2025; Rapacz and Smywiński-Pohl, 2025, i.a.). On the scientific side, morphological inflections are often complex transformations at the word and sub-word level (Chandlee, 2017). Investigating how neural systems handle these transformations, and the circumstances under which they are able to generalize them, provides a window into their linguistic generalization capabilities (Kodner et al., 2023b).

In recent years, the SIGMORPHON shared tasks on morphological inflection have reported generally high accuracy across many languages (Cotterell et al., 2016, 2017, 2018; McCarthy et al., 2019; Vylomova et al., 2020; Pimentel et al., 2021; Kodner et al., 2022; Goldman et al., 2023). Work seeking to understand this success has investigated predictors of model performance in terms of generalization along different dimensions. Two aspects of this are **lemma overlap**: whether the lemma in a test (`lemma, features, inflected form`) triple is also attested in train, and **feature overlap**, whether the feature set in a test triple is also attested in train. In both cases, entire *triples* do not reappear between train and test, rather the test triple contains a lemma or feature set seen in a distinct train triple. The effect of overlap is measured by comparing accuracy on test triples containing seen lemmas or feature sets to those containing the unseen ones (Goldman et al., 2022; Kodner et al., 2022); it thus probes the ability of neural models to *generalize* to novel lemmas or feature sets.

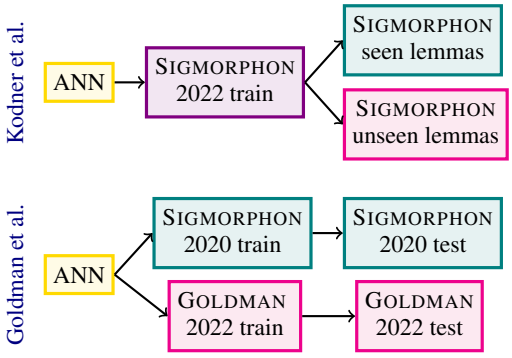

Figure 1: A schematization of the train and test methods of Kodner et al. (2022) and Goldman et al. (2022).

Previous work on lemma overlap has drawn conflicting conclusions. Goldman et al. (2022) compare model performance on the original SIGMORPHON 2020 splits – in which triples were partitioned uniformly at random – to their "lemma-splits," in which *entire paradigms* are partitioned uniformly at random, and thus all lemmas occur only in train or only in test. Goldman et al. report

a 30% average decrease in accuracy from the models trained and tested on the SIGMORPHON splits to those trained and tested on their lemma-splits. However, Kodner et al. (2022) and Kodner et al. (2023b) find no performance drop on unseen lemmas. Their setup involves a single training split, with a test set containing at most 50% triples with attested features; this also results in a significant portion of test triples with unseen lemmas. Kodner et al. train a single model per language and find no difference in accuracy between seen and unseen lemmas in their test set. Both experimental designs are conceptualized in Fig. 1. Crucially, while any drop in test accuracy can be attributed directly to differences in the test items in Kodner et al.'s setup, they could be attributed either to differences in the test items *or* the train sets in Goldman et al.'s.

In contrast to lemma overlap, previous work on feature overlap has unequivocally shown that neural models struggle to generalize to unseen feature sets: Kodner et al. (2022) found that all systems perform better on test triples with attested feature sets than those with unattested ones, with the gap in accuracy ranging from 32% to as high as 79%. These findings were replicated by Kodner et al. (2023a), who performed a more in-depth analysis on 6 languages. Kodner et al. (2023b) created linguistically-informed language-specific probes, and found that 2 of the 3 models which they tested only succeeded on the simplest feature-based generalization, while the third failed on all probes.

In this paper, we aim to understand the causes of the conflicting results on lemma overlap and further investigate what factors impact models' ability to generalize to novel lemmas. Because of the consistent findings regarding feature overlap, we hold it constant; our goal is to understand lemma and feature overlap separately so that future work may investigate their interaction. In our first experiment (§3), we introduce a novel splitting algorithm which fixes a single train set and ensures that 50% of test triples contain seen lemmas. We find an average accuracy drop of 11%, between those reported by Kodner et al. and Goldman et al.. More importantly, we show that the number of training lemmas has a much stronger effect on accuracy on unseen than seen lemmas. In our second experiment (§4), we replicate Goldman et al. to investigate causes of our differing results, and show that the number of lemmas in train is a stronger predictor of generalization to unseen lemmas than the number of triples. Since Goldman et al.'s lemma-splitting held triple count roughly constant but introduced a near-30% drop in the number of lemmas from SIGMORPHON to their lemma-based splits, they exaggerated the true impact of lemma overlap. These results are a step towards a more complete and interpretable picture of the dimensions across which neural models succeed and fail at generalization.

2 Background: Defining Overlap

In morphological inflection tasks, models are exposed to triples of `(lemma, feature set, inflected form)` in train. During evaluation, they are given a `(lemma, feature set)` pair as input and expected to correctly predict the corresponding inflected form. For example, given (`walk`, V;PAST), a model should output `walked`. In iterations of the SIGMORPHON shared task before 2022, triples are partitioned uniformly at random into train or test. While this entails that no triple occurring in train will occur in test, as both Goldman et al. (2022) and Kodner et al. (2022) note, it ignores the fact that lemmas or feature sets that appear in train may reappear during test separately. Consider a simple example, adapted from Kodner et al. (2023b):

(1) Example train set:

t0:	see	seeing	V;V.PTCP;PRS
t1:	sit	sat	V;PST

(2) Example test set:

e0:	see	V;PST
e1:	sit	V;NFIN
e2:	eat	V;PST
e3:	run	V;PRS;3;SG

Though none of the triples in train (1) re-appear in test (2), the lemmas and feature sets seen in train *do* reappear individually. Consider `e0`, for which both the lemma and feature set are attested separately in train, and `e3`, for which neither are. One might expect it to be easier for a model trained on (1) to generate the correct result for `e0` than for `e3`. This is the insight behind studies of overlap: test pairs with novel lemmas or novel feature sets require a system to generalize along different dimensions, and evaluation measures that control specifically for these overlaps can better probe models' ability to perform these generalizations. Kodner et al. (2023b) define four types of overlap:

`both` both the lemma and feature set are attested separately in train (`e0` in Example 2).

`lemmaOnly` only the lemma is attested in train, and the feature set is novel (`e1` in Example 2).

`featsOnly` only the feature set is attested in train, and the lemma is novel (`e2` in Example 2)

neither neither the lemma nor the feature set is attested in train (e3 in Example 2)

We define **lemmaSeen** to be any evaluation triple with a lemma attested in train (i.e., lemmaSeen = lemmaOnly ∪ both) and **lemmaNovel** to be any evaluation triple with a lemma unattested in train (i.e., lemmaNovel = featsOnly ∪ neither).

3 Exp. 1: The Effect of Lemma Overlap

In this section, we test the impact of lemma overlap with a novel splitting algorithm that ensures that 50% of test triples contain seen lemmas. Like Kodner et al., our setup ensures a single training set per language, but like Goldman et al., we focus on lemma overlap rather than feature set overlap.

3.1 Splitting Algorithm

Our algorithm takes in a set of TRIPLES and a set of the unique LEMMAS contained in these triples. It begins by randomly selecting half[1] of the lemmas as our OVERLAPLEMMAS. It then iteratively adds more lemmas to OVERLAPLEMMAS until OVERLAPTRIPLES, the number of triples containing a lemma in OVERLAPLEMMAS, is at least equal to the sum of the train size, fine tune [2] size, and half of the test size. Once OVERLAPTRIPLES is sufficiently large, we sample TRAIN and FINETUNE from it, as well as half of our test set; the remaining half is sampled from TRIPLES \ OVERLAPTRIPLES. This results in a single train set and a test set where exactly half of test triples contain a lemma seen in train. This algorithm is visualized in Figure 2, and pseudocode is provided in Appendix A.[3]

3.2 Languages

Following Goldman et al., we use the data from the SIGMORPHON 2020 shared task (Vylomova et al., 2020), which contains more languages than SIGMORPHON 2022. We select a 7,000 /1,000 / 2,000 train/fine-tune/test split to follow both the 70% / 10% / 20% splitting of Goldman et al. and the 7,000 training size used by Kodner et al.. To satisfy our split sizes, we retain only languages with at least 10,000 raw triples. This leaves us with

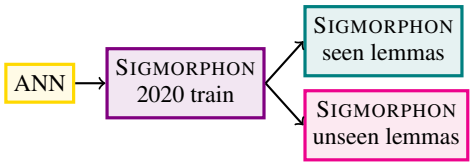

Figure 2: A schematization of our setup for Exp. 1

36 languages across 7 families; these are listed in Appendix A. We create 5 random (downsampled) train-test splits since the performance of ANNs on morphological tasks is known to vary across splits (Liu and Prud'hommeaux, 2022; Kodner et al., 2023b). Across all splits, feature overlap is near-ceiling, with an average of 99.5% of test triples containing attested features ($\sigma = 1.17\%$). Though we report the following analyses on *all* test triples, they do not differ significantly if done only with those test triples containing attested feature sets.

3.3 Neural Models

We select three neural models for our evaluation, spanning a diverse range of architectures:

CHR-TRM (Wu et al., 2021) is a character-level transformer used as a baseline in SIGMORPHON 2021 and 2022, and the only model used by both Goldman et al. and Kodner et al.. We use the hyperparameters suggested by the original authors for small training conditions.[4]

CLUZH (Wehrli et al., 2022) is a character-level transducer used by Kodner et al., who consider two variants with consistent hyper-parameters across languages. We use the beam decoding variant, which consistently performs better in their experiments, with a beam size of 4 and the large training hyperparameters.[5]

DeepSpin (Peters and Martins, 2020) is a multilingual RNN used by Goldman et al.. It is composed of 2 biLSTM encoders with bi-linear gated attention using sparsemax (Martins and Astudillo, 2016) and one unidirectional LSTM decoder.[6]

We exclude CULING (Liu and Hulden, 2020) due to its similarity to CHR-TRM and Goldman et al.'s base LSTM since they do not fully report results on this model.

3.4 Results

For each architecture, language, and seed, we calculate SEEN and NOVEL accuracy on lemmaSeen

[1] We also tried sampling starting from 1 lemma until the requisite number was met, and results were similar.

[2] Often called "validation." We call this set "fine tune" to clearly distinguish its purpose, avoiding a commom confusion, cf. (van der Goot, 2021).

[3] Our code splits, and neural model outputs can be found at: https://github.com/paynesa/lemmas-matter

[4] https://github.com/shijie-wu/neural-transducer

[5] https://github.com/slvnwhrl/il-reimplementation

[6] https://github.com/deep-spin/sigmorphon-seq2seq

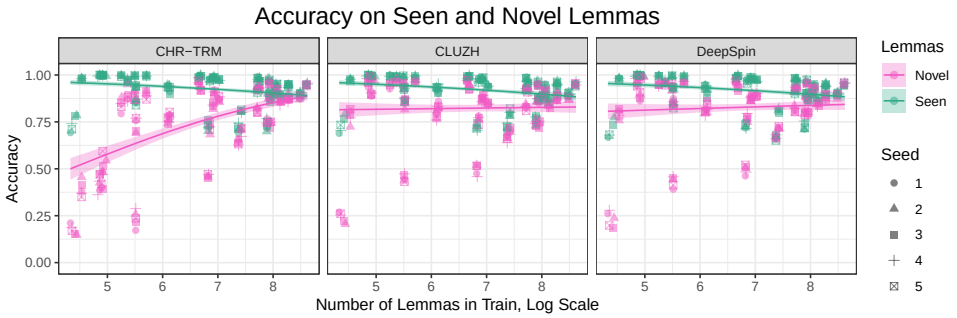

Figure 3: The ACCURACY ∼ LEMMAS beta regression models for SEEN and NOVEL accuracy.

Model	SEEN	NOVEL	DROP	Reported
CHR-TRM	92.594	75.786	-16.808	-37.00
CLUZH	92.138	83.221	-8.917	—
DEEPSPIN	91.946	83.828	-8.118	-14.00
Macro-Avg	92.226	80.945	-11.281	-30.00

Table 1: Accuracy on seen and unseen lemmas by neural architecture, averaged across seeds & languages.

and `lemmaNovel` test triples respectively; DROP is the difference between the two. The averages of these values by architecture are given in Table 1, as well as those reported by Goldman et al. (2022). Notably, our macro-averaged accuracy drop is just a third of that reported by Goldman et al.; this difference is most notable for CHR-TRM. Full results by language and architecture are in Appendix A.

Though we fix training size, the number of unique lemmas in the training data varies across languages. To investigate the effect of training lemmas on SEEN and NOVEL accuracy, we fit two beta regression models, implemented via the `mcgv` package in R with `family = betar` and the default logit linking function. We choose beta regression because our response variables are in the interval (0, 1) and follow a beta distribution (Appendix A), and we log-scale LEMMAS, our predictor, to obtain a better model fit. The fitted ACCURACY ∼ LEMMAS models are given in Fig. 3. We note that NOVEL accuracy falls off much more steeply as the number of training lemmas decreases than SEEN accuracy; this is most striking for CHR-TRM.

Interestingly, the SEEN β coefficients are all significantly negative (-0.120^* for CHR-TRM, -0.130^* for CLUZH and -0.098^* for DEEPSPIN), suggesting that performance on seen lemmas actually *decreases* as the number of training lemmas increases. This may be a function of feature overlap: since training size is fixed, languages with more training lemmas will have smaller paradigms available in train, which may lead to a greater proportion of test triples containing novel features. Indeed, there is a significant negative relationship between the number of training lemmas and the proportion of test triples with attested features in our data (Pearson's $r = -0.407^*$, Spearman's $\rho = -0.423^*$, Kendall's $\tau_B = -0.332^*$). Since feature overlap is known to play a key role in model performance (Kodner et al., 2022), this may explain the relationship we observe for SEEN accuracy. Further, the number of times a model is exposed to a single lemma in train *regardless* of feature overlap may affect its ability to generalize that lemma to other parts of its paradigm. Both of these points are closely tied to the notion of *paradigm saturation*, or the proportion of its paradigm that a lemma is attested in during train (Chan, 2008; Lignos and Yang, 2016). Future work should further investigate the interplay between paradigm saturation, lemma overlap, and feature overlap in predicting models' generalization abilities.

In contrast, NOVEL accuracy significantly increases as a function of the number of training lemmas ($\beta = 0.558^*$ for CHR-TRM, 0.148^* for CLUZH, 0.190^* for DEEPSPIN). The combination of the negative coefficients for SEEN accuracy and the positive ones for NOVEL accuracy further mean that there is a significant relationship between DROP and LEMMAS for all architectures ($\beta = -0.524^*$ for CHR-TRM, -0.174^* for CLUZH, and -0.194^* for DEEPSPIN). Despite this significant relationship however, the overall effect of lemma overlap is still far less than that reported by Goldman et al. at only 11% on average.

4 Exp. 2: Explaining the Discrepancy

The difference between our results and those of Goldman et al. (2022) are partly explainable by the relationship between training lemmas and accuracy drop outlined above. Since we fix our train size, we exclude from our analysis languages with minis-

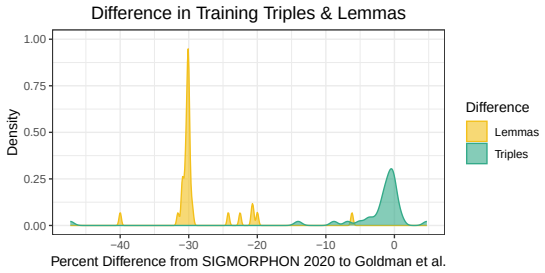

Figure 4: The distribution of percent differences in triple and lemma counts between the SIGMORPHON 2020 to the Goldman et al. training data.

SIGMORPHON		CHR-TRM	CLUZH	DEEPSPIN
TRIPLES	β	0.283*	0.360*	0.226*
	AIC	**-17170.90**	**-15843.95**	**-17413.73**
	BIC	**-17167.85**	**-15840.90**	**-17410.68**
LEMMAS	β	0.156	0.270*	0.129
	AIC	-17056.13	-15712.94	-17356.09
	BIC	-17053.08	-15709.89	-17353.04
GOLDMAN		CHR-TRM	CLUZH	DEEPSPIN
TRIPLES	β	0.643*	0.396*	0.286*
	AIC	-3694.535	**-10671.66**	**-12273.43**
	BIC	-3691.483	**-10668.61**	**-12270.37**
LEMMAS	β	0.811*	0.370*	0.192
	AIC	**-3798.311**	-10473.82	-12163.35
	BIC	**-3795.258**	-10470.76	-12160.30

Table 2: β coefficients (asterisks indicate $p < 0.05$) and AIC & BIC values (the better predictor is bolded) for the accuracy beta regression models.

cule numbers of training lemmas; Goldman et al. include these languages in their analysis. Indeed, the median number of training lemmas in our data is 1,015 vs. just 141 for Goldman et al., despite comparable means (1,625 and 2,038, respectively).

However, since Goldman et al. train two separate models for each language (Fig. 1), the resulting differences in performance could also be attributable to differences in the training data itself. Even if Goldman et al. had used an identical splitting algorithm across the two train-test splits, the performance of morphological ANNs varies across random seeds (Liu and Prud'hommeaux, 2022, i.a.) and initializations (e.g. Corkery et al., 2019). Moreover, since their train sets were derived from separate splitting algorithms, they show substantial distributional differences which may be relevant to performance. Firstly, though Goldman et al. retain the train-test ratios from SIGMORPHON 2020, the number of triples in their train splits do not match exactly since they split on *entire paradigms*. They report an average decrease in training triples of 3.5%, but there is a great deal of variation ($\sigma = 6.341\%$), with one language (Ludic) experiencing a 47.3% decrease. What's more, Goldman et al.'s lemma-splitting strategy results in a 28.846% ($\sigma = 5.221\%$) average drop in the number of training lemmas between the SIGMORPHON 2020 splits and their splits, which they do not report. The cause of this drop is intuitive: if we assume that nearly all available lemmas will appear at least once in the SIGMORPHON training data, then since Goldman et al. sample 70% for train, we expect about a 30% decrease in the number of training lemmas. Fig. 4 shows the distributions of train triple and lemma count decreases.

Implicit in the conclusions drawn by Goldman et al. (2022) is the assumption that the decrease in the number of training lemmas does not meaningfully influence their comparison, and thus that the number of training lemmas is not a predictor of model performance or performance drop. The evidence presented in §3 already suggests that this is not the case, but to further investigate this point, we replicate a subset of Goldman et al.'s results and investigate predictors of raw accuracy and accuracy drop. We show that the number of training lemmas, both measured directly and in terms of the decrease between the two training sets, has a significant effect on accuracy and is the best predictor of accuracy drop across architectures. We further show that Goldman et al.'s reported accuracy drops are substantially higher than they would have been had they held lemma counts constant.

4.1 Setup

We focus our replication on three language families: Niger-Congo, Uralic, and Romance. These were chosen for the range of SIGMORPHON data sizes: Niger-Congo data are typically quite small, while Uralic are large, and Romance contains a mix. Languages are listed in Appendix B. We retain the neural architectures from §3.3.

4.2 Accuracy and Accuracy Drop

To differentiate our results on the Goldman et al. splits from the results reported by Goldman et al., we denote our results as GOLDMAN. We calculate GOLDMAN accuracy and SIGMORPHON accuracy for each language and neural architecture, as well as the DROP in accuracy between these two. Since Goldman et al. did not fix train size in their study, we consider both TRIPLES, the number of unique training triples, and LEMMAS, the number of unique training lemmas, as possible predictors for raw accuracy. For DROP, we consider each of these predictors as measured on the GOLDMAN

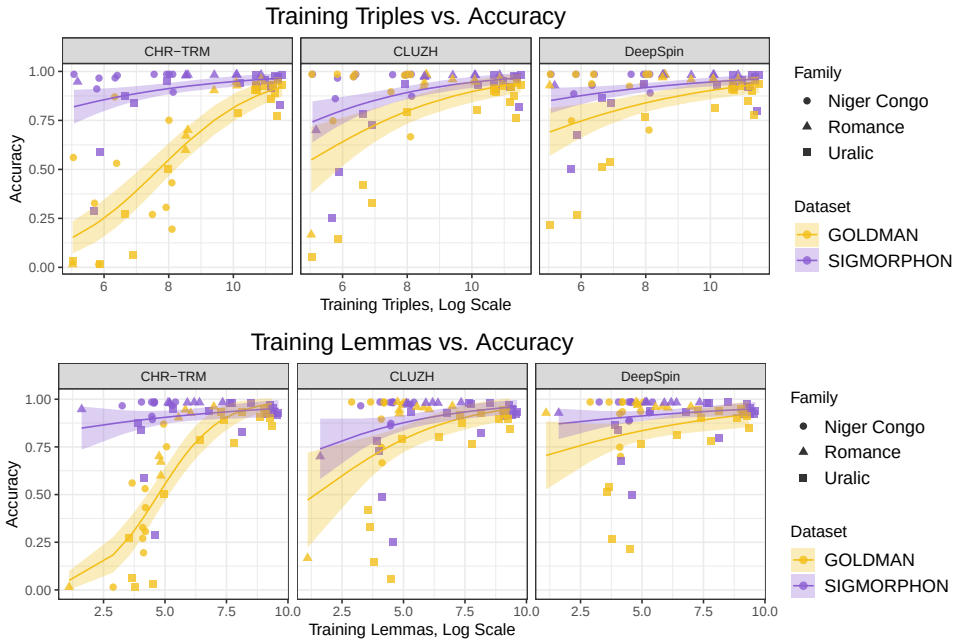

Figure 5: The fitted ACCURACY ∼ TRIPLES and ACCURACY ∼ LEMMAS beta regression models.

data, as well as LEMMA DROP, the unscaled difference in the number of lemmas from the SIGMORPHON to the GOLDMAN training data.[7] Following §3, for each data set and neural architecture, we fit a beta regression for each predictor, all log-scaled.

The fitted models for raw accuracy are in Fig. 5. Similarly to §3, we note that both TRIPLES and LEMMAS have a much stronger effect on GOLDMAN accuracy than SIGMORPHON. This is mirrored by the β coefficients, which are given in Table 2, along with AIC and BIC, two measures of the goodness of model fit that balance model performance, in terms of log likelihood, with model complexity. For both measures, lower values indicate a better model. Since we used a logit linking function in our beta regression model, our β coefficients are *log-odds interpretable*: a one-unit increase in the predictor increases the log-odds of the predicted mean proportion by β. As such, larger values of β correspond to stronger (or steeper, in terms of slope) influences on accuracy. In line with Fig. 5, β coefficients are consistently larger for GOLDMAN than SIGMORPHON accuracy. For CHR-TRM on GOLDMAN, LEMMAS is the better predictor of accuracy, as evidenced by the significant β coefficients and lower AIC and BIC values. In all other cases, TRIPLES is the better predictor.

[7]One could also consider scaling LEMMA DIFFERENCE as a proportion decrease. However, because these scaled values are so closely clustered around the mean, there is little relation between them and DROP.

Model	S ACC	G ACC	DROP	Reported
CHR-TRM	93.451	61.067	-32.383	-37.00
DEEPSPIN	94.036	86.377	-7.660	-14.00
CLUZH	91.222	81.790	-9.432	—
Macro-Avg	92.903	76.411	-16.492	-30.00
Family	**S ACC**	**G ACC**	**DROP**	**Reported**
Niger-Congo	97.470	76.162	-21.308	-39.00
Romance	97.999	85.478	-12.521	-28.00
Uralic	87.502	72.035	-15.467	-23.00
Macro-Avg	94.323	77.891	-16.432	-30.00

Table 3: SIGMORPHON (S ACC) and GOLDMAN (G ACC) accuracy and DROP by architecture and language family compared to Goldman et al. reported drops.

For each dataset and neural architecture, we fit an additional beta regression model using both TRIPLES and LEMMAS as predictors. Since our two predictors are highly co-linear ($r = 0.904$, $\rho = 0.891$, $\tau_b = 0.717$ for SIGMORPHON; $r = 0.892$, $\rho = 0.891$, $\tau_b = 0.722$ for GOLDMAN), we use the better predictor as the *reference category*, so the effect of the second predictor is only measured as that *beyond* the effect of the first. An ANOVA test determines that the second predictor does not have a significant effect beyond the first for any architecture on the GOLDMAN data. For SIGMORPHON, LEMMAS has a significant effect beyond that of TRIPLES for CHR-TRM and DEEPSPIN. Full model details and outputs are in Appendix B.

We now turn from raw accuracy to accuracy DROP, which Goldman et al. reported in their original study. Table 3 lists our observed accu-

Model	Triples			Lemmas			Lemma Drop		
	Coeff.	AIC	BIC	Coeff.	AIC	BIC	Coeff.	AIC	BIC
Chr-Trm	0.554*	1550.002	1553.055	0.904*	**1016.286**	**1019.338**	0.816*	1193.401	1196.454
Cluzh	0.239*	5855.958	5859.011	0.287*	**5823.695**	**5826.747**	0.300*	5831.558	5834.611
DeepSpin	0.238*	7018.613	7021.666	0.210*	7027.519	7030.572	0.253*	**6992.300**	**6995.352**

Table 4: β coefficients (asterisks indicate $p < 0.05$) and AIC and BIC values (the better predictor is bolded) for the three predictors of Drop.

Model	Constant	Decrease	Diff
Chr-Trm	25.689	30.585	4.896
DeepSpin	6.681	7.521	0.840
Cluzh	7.010	8.855	1.845
Macro-Avg	13.127	15.654	2.527

Table 5: Predicted Constant/Decrease acc. drops

racy drops by neural architecture and by language family, compared to those reported by Goldman et al.. We observe an average decrease of 45% from the values reported by Goldman et al. to our setup. This discrepancy is explainable in two ways. First, Cluzh outperformed Goldman et al.'s base LSTM, which brought down average performance on unseen lemmas. Second, the families we omitted from our replication contain many languages with extremely small training sets, for which we expect there to be larger drops given Fig. 5.

We also perform a beta regression with each predictor for Drop, and find that the best predictor for all three architectures depends on lemmas: either Lemmas or Lemma Drop (Table 4). Given the collinearity between these predictors, however (Lemmas and Lemma Drop: $r = 0.986$, $\rho = 0.978$, $\tau_b = 0.945$), we wish to test whether the addition of either of the other predictors significantly increases in the predictive power of the beta regression model. As for Accuracy, we train a third beta regression model for each architecture with all three predictors, again using the best predictor as the reference category, with the other two added in order of increasing AIC. For all three architectures, an ANOVA test indicates that addition of neither the second nor third predictor led to a significant increase in model performance.

In sum, though Triples is often an important predictor of raw accuracy, Lemmas or Lemma Drop are the key predictors of Drop, which was the original focus of Goldman et al. (2022). As such, the drop in training lemmas from the Sigmorphon to Goldman data can be expected to significantly affect the drop in accuracy between these datasets, a confound which favors the increased accuracy drop reported by Goldman et al..

4.3 Quantifying the Confound

Our results thus far demonstrate that the number of training lemmas plays a crucial role in predicting the drop in accuracy between Sigmorphon and Goldman: train sets with fewer lemmas exhibit a larger drop. As we saw above, however, the Goldman setup creates train sets with fewer lemmas than their Sigmorphon counterparts even when their size in triples is held constant. Consider, for example, a language with 1000 training lemmas in Sigmorphon. Because accuracy will fall off for the Goldman data as the number of training lemmas decreases, the drop in accuracy between the 1000-lemma Sigmorphon sample and a 1000-lemma Goldman sample will be significantly less than that between a 1000-lemma Sigmorphon sample and the 700-lemma Goldman sample predicted by the near-30% decrease in the number of training lemmas.

We can further formalize this intuition using the beta regression models from §4.2. For values of Lemmas from 10 to 25,000, we calculate two types of accuracy drops. Decrease represents Goldman et al.'s actual approach in which train sets averaged nearly 30% fewer lemmas than their Sigmorphon 2020 counterparts. It is calculated by subtracting the predicted Goldman accuracy on $0.7L$ lemmas from the predicted Sigmorphon accuracy on L lemmas. Constant drop is meant to model the results had train lemmas been held constant, and is calculated by subtracting the predicted Goldman accuracy on L lemmas from the predicted Sigmorphon accuracy on L lemmas.

Fig. 6 shows the differences between the Constant and Decrease predicted drops for each of the neural architectures as a function of the number of training Lemmas plotted on top of the actual distribution of training lemmas in the Goldman and Sigmorphon data. Notably, the distribution of train sizes peaks at the point where Goldman et al.'s Decrease lemma-splitting approach overpredicts accuracy drop by the greatest degree. As such, the original study's confounds mutually reinforced one another to increase accuracy drops.

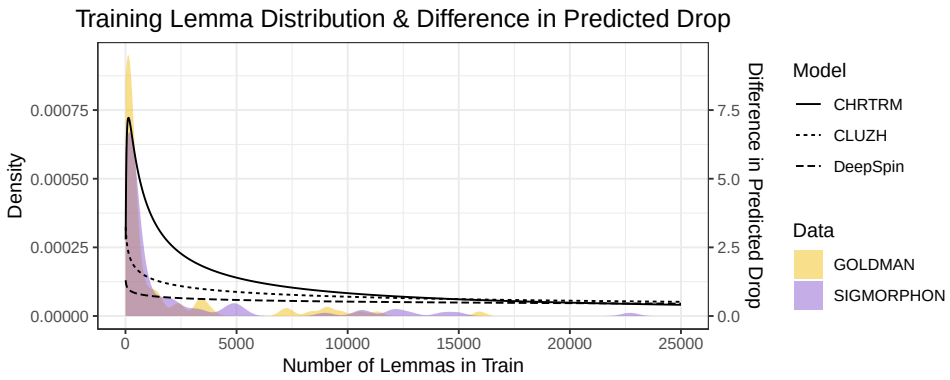

Figure 6: The difference between the CONSTANT and DECREASE predicted drop as a function of the number of training lemmas, plus the distribution of training lemmas in the SIGMORPHON and GOLDMAN data.

Table 5 gives the average predicted CONSTANT and DECREASE accuracy drops for each architecture for the actual values of training lemmas in the SIGMORPHON and GOLDMAN data. The average difference in drop is 2.5%, with the maximum (for CHR-TRM) estimated at 4.9%. This means that if Goldman et al. (2022) had held the number of training lemmas constant between their overlapping and non-overlapping splits, they would have found an effect of lemma overlap several percentage points smaller than the one they reported.

5 Discussion

The findings presented here suggest a more nuanced view of the role of lemma overlap than proposed by either Goldman et al. (2022) or Kodner et al. (2022). With our novel splitting algorithm (§3), we find that all neural architectures achieve substantially higher accuracy on seen than unseen lemmas, unlike Kodner et al., but the difference is much lower (11%), about a third of that reported by Goldman et al. reported (30%). More importantly, however, we show that the number of lemmas in train has a much stronger effect on accuracy on test triples containing unseen lemmas than seen ones.

While there are several incidental differences between our study and the two prior studies, we established one principled difference which resulted in the especially large performance drop which Goldman et al. reported: the number of lemmas in the training set (absolute or relative to the original set) is the most important predictor for accuracy drop, with larger accuracy drops occurring for languages with fewer train lemmas. Goldman et al. tested the effect of lemma overlap with two distinct training sets, one which allowed for lemma overlap, and one that both banned overlap *and* was on average 30% smaller in terms of lemma count; this was a confound which inflated accuracy drop. As we found in §4.3, Goldman et al.'s data was also highly skewed towards small training sizes where the impact of this confound was at its greatest.

Our own study of lemma overlap was set up similarly to Kodner et al. (2022) and Kodner et al. (2023b) in that we trained models on a single training set and tested them on distinct test sets (Fig. 2). This eliminates the potential for confounds such as the training lemma count. Unlike Kodner et al., who found no effect of lemma overlap, however, we found a substantial drop. One possible reason for this discrepancy is the role of feature attestation: in our study, we ensured a 50% seen/unseen lemma split in test. Doing so on at least 10,000 triple data sets caused feature set overlap to rise to 99.5%, thus controlling for this additional factor; indeed, our analysis does not significantly change when conducted only over test triples containing seen feature sets. Kodner et al., on the other hand, fixed feature set overlap to a maximum of 50% and did not control for lemma overlap explicitly.

It is possible that this discrepancy emerges as an artifact of the feature-based splitting algorithm or of the interaction between lemma and feature overlap. If, for example, lemma overlap has a small but significant effect for triples with seen features (what our results show) but does not play a role in triples with unseen features, this could help to explain the results of Kodner et al.. Now that we have established a better understanding of the effect of lemma overlap on triples with seen feature sets, future work should further investigate the interaction between lemma overlap and feature overlap. As discussed in §3, future work should also investigate the effect of the *number* of times a given lemma

or feature set has been seen in train rather than making a binary attested-unattested distinction.

The findings presented here have implications for our understanding of subword-level generalization in different classes of neural network and are thus an important contribution interpretability for these models. Intuitively, generalizing morphological inflections to unseen lemmas is a more challenging task than generalization to seen lemmas, since it requires generalization from the lemma and feature sets only. Without additional evidence from other inflected forms of the lemma, we cannot know, for example, if it follows a regular or irregular pattern.

While all models performed similarly to each other on languages with the largest numbers of training lemmas (Fig. 3), they diverged in their ability to generalize on languages with the smallest numbers of training lemmas. CHR-TRM, a character transformer trained on a single language, struggled the most to generalize to unseen lemmas on the smallest training sets, conforming with our expectations about the data hungry nature of transformers. On the other hand, DEEPSPIN, a multilingual model with a much simpler BiLSTM architecture, much more readily generalized to unseen lemmas even at the smallest training sizes and was nearly immune to performance drop.

6 Conclusion

In this paper, we have demonstrated that the number of lemmas in the training data significantly affects the performance of neural models on unseen lemmas, but not on seen lemmas. This result entails that neural models struggle to generalize to novel lemmas when trained on small data, an important step to understanding how different classes of model generalize at a subword level. Beyond just understanding how models generalize or fail to, inflection still remains an important element in NLP for morphologically rich low-resource and endangered languages, as evidenced by contributions to recent workshops on NLP for under-resourced, endangered, and indigenous language families.[8]

In addition to the main results concerning generalization to seen and unseen lemmas, this paper provides a methodological contribution. We find that careful experimental design lets us not only reassess top line numbers claimed in prior work (namely, Goldman et al. 2022), but also provides an explanation for the causes of previous discrepancies and allows us to identify confounds. We evaluated the role of lemma overlap while controlling for a number of potential confounds by holding the training set constant. We augmented the reporting of top-level accuracy numbers with statistical analysis modeling the role of confounding factors on performance. It is likely that these approaches will prove useful in future work assessing patterns in morphological generalization. In particular, now that a separate understanding of the roles of feature overlap and lemma overlap has been established, future work is well-positioned to investigate their interaction. While overlap has been quantified in terms of binary attestation in train, future work may also consider the *number* of times a given lemma or feature set is attested in order to gain a more complete picture of factors affecting generalization.

[8] e.g., ComputEL, AmericasNLP, RAIL, ArabicNLP

Acknowledgements

We are grateful to Sandy Abu El Adas, Mark Aronoff, Smeet Chheda, Jeff Heinz, Scott Nelson, Owen Rambow, and the anonymous reviewers for their feedback on this work. SP gratefully acknowledges support from the Institute for Advanced Computational Science Graduate Research Fellowship and the National Science Foundation Graduate Research Fellowship Program under NSF Grant No. 2234683. Experiments were performed on the SeaWulf HPC cluster maintained by RCC and IACS at Stony Brook University and made possible by NSF grant No. 1531492.

Limitations

We discuss two key limitations. Firstly, any experiment of this nature must generalize from the specific neural architectures and languages tested to draw more general claims. In §4, we discussed the effects our inclusion of CLUZH and omission of Goldman et al.'s base LSTM, but future work should investigate more neural architectures, as well as more language samples as they become available in future SIGMORPHON shared tasks. Secondly, while we have conducted an extensive comparison between our results and those of Goldman et al., we have yet to conduct a similar comparison with Kodner et al. since the latter paper also deals with feature overlap. Now that we have a better understanding of the factors affecting lemma overlap, however, we are better positioned to investigate its interaction with feature overlap, which we plan to do in future work.

References

Catherine Arnett and Benjamin Bergen. 2025. Why do language models perform worse for morphologically complex languages? In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6607–6623, Abu Dhabi, UAE. Association for Computational Linguistics.

Emily M. Bender. 2009. Linguistically naïve != language independent: Why NLP needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 26–32, Athens, Greece. Association for Computational Linguistics.

Erwin Chan. 2008. *Structures and distributions in morphological learning*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Jane Chandlee. 2017. Computational locality in morphological maps. *Morphology*, 27(4):599–641.

Maria Corkery, Yevgen Matusevych, and Sharon Goldwater. 2019. Are we there yet? encoder-decoder neural networks as cognitive models of English past tense inflection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3868–3877, Florence, Italy. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sabrina J. Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27, Brussels. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared Task—Morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Omer Goldman, Khuyagbaatar Batsuren, Salam Khalifa, Aryaman Arora, Garrett Nicolai, Reut Tsarfaty, and Ekaterina Vylomova. 2023. SIGMORPHON–UniMorph 2023 shared task 0: Typologically diverse morphological inflection. In *Proceedings of the 20th SIGMORPHON workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 117–125, Toronto, Canada. Association for Computational Linguistics.

Omer Goldman, David Guriel, and Reut Tsarfaty. 2022. (un)solving morphological inflection: Lemma overlap artificially inflates models' performance. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 864–870, Dublin, Ireland. Association for Computational Linguistics.

Jordan Kodner, Salam Khalifa, Khuyagbaatar Batsuren, Hossep Dolatian, Ryan Cotterell, Faruk Akkus, Antonios Anastasopoulos, Taras Andrushko, Aryaman Arora, Nona Atanalov, Gábor Bella, Elena Budianskaya, Yustinus Ghanggo Ate, Omer Goldman, David Guriel, Simon Guriel, Silvia Guriel-Agiashvili, Witold Kieraś, Andrew Krizhanovsky, Natalia Krizhanovsky, Igor Marchenko, Magdalena Markowska, Polina Mashkovtseva, Maria Nepomniashchaya, Daria Rodionova, Karina Scheifer, Alexandra Sorova, Anastasia Yemelina, Jeremiah Young, and Ekaterina Vylomova. 2022. SIGMORPHON–UniMorph 2022 shared task 0: Generalization and typologically diverse morphological inflection. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 176–203, Seattle, Washington. Association for Computational Linguistics.

Jordan Kodner, Salam Khalifa, and Sarah Ruth Brogden Payne. 2023a. Exploring linguistic probes for morphological generalization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8933–8941, Singapore. Association for Computational Linguistics.

Jordan Kodner, Sarah Payne, Salam Khalifa, and Zoey Liu. 2023b. Morphological inflection: A reality check. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6082–6101, Toronto, Canada. Association for Computational Linguistics.

Constantine Lignos and Charles Yang. 2016. Morphology and language acquisition. *The Cambridge handbook of morphology*, 743764.

Ling Liu and Mans Hulden. 2020. Leveraging principal parts for morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 153–161, Online. Association for Computational Linguistics.

Zoey Liu and Emily Prud'hommeaux. 2022. Data-driven model generalizability in crosslinguistic low-resource morphological segmentation. *Transactions of the Association for Computational Linguistics*, 10:393–413.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning*, pages 1614–1623. Proceedings of Machine Learning Research.

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sabrina J. Mielke, Jeffrey Heinz, Ryan Cotterell, and Mans Hulden. 2019. The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244, Florence, Italy. Association for Computational Linguistics.

Kemal. Oflazer and Murat. Saraçlar. 2018. *Turkish Natural Language Processing*. Theory and Applications of Natural Language Processing. Springer International Publishing, Cham, Switzerland.

Ben Peters and André F. T. Martins. 2020. One-size-fits-all multilingual models. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 63–69, Online. Association for Computational Linguistics.

Tiago Pimentel, Maria Ryskina, Sabrina J. Mielke, Shijie Wu, Eleanor Chodroff, Brian Leonard, Garrett Nicolai, Yustinus Ghanggo Ate, Salam Khalifa, Nizar Habash, Charbel El-Khaissi, Omer Goldman, Michael Gasser, William Lane, Matt Coler, Arturo Oncevay, Jaime Rafael Montoya Samame, Gema Celeste Silva Villegas, Adam Ek, Jean-Philippe Bernardy, Andrey Shcherbakov, Aziyana Bayyr-ool, Karina Sheifer, Sofya Ganieva, Matvey Plugaryov, Elena Klyachko, Ali Salehi, Andrew Krizhanovsky, Natalia Krizhanovsky, Clara Vania, Sardana Ivanova, Aelita Salchak, Christopher Straughn, Zoey Liu, Jonathan North Washington, Duygu Ataman, Witold Kieraś, Marcin Woliński, Totok Suhardijanto, Niklas Stoehr, Zahroh Nuriah, Shyam Ratan, Francis M. Tyers, Edoardo M. Ponti, Grant Aiton, Richard J. Hatcher, Emily Prud'hommeaux, Ritesh Kumar, Mans Hulden, Botond Barta, Dorina Lakatos, Gábor Szolnok, Judit Ács, Mohit Raj, David Yarowsky, Ryan Cotterell, Ben Ambridge, and Ekaterina Vylomova. 2021. SIGMORPHON 2021 shared task on morphological reinflection: Generalization across languages. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–259, Online. Association for Computational Linguistics.

Maciej Rapacz and Aleksander Smywiński-Pohl. 2025. Low-resource interlinear translation: Morphology-enhanced neural models for Ancient Greek. In *Proceedings of the First Workshop on Language Models for Low-Resource Languages*, pages 145–165, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rob van der Goot. 2021. We need to talk about train-dev-test splits. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4485–4494, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ekaterina Vylomova, Jennifer White, Elizabeth Salesky, Sabrina J. Mielke, Shijie Wu, Edoardo Maria Ponti, Rowan Hall Maudslay, Ran Zmigrod, Josef Valvoda, Svetlana Toldova, Francis Tyers, Elena Klyachko, Ilya Yegorov, Natalia Krizhanovsky, Paula Czarnowska, Irene Nikkarinen, Andrew Krizhanovsky, Tiago Pimentel, Lucas Torroba Hennigen, Christo Kirov, Garrett Nicolai, Adina Williams, Antonios Anastasopoulos, Hilaria Cruz, Eleanor Chodroff, Ryan Cotterell, Miikka Silfverberg, and Mans Hulden. 2020. SIGMORPHON 2020 shared task 0: Typologically diverse morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 1–39, Online. Association for Computational Linguistics.

Silvan Wehrli, Simon Clematide, and Peter Makarov. 2022. CLUZH at SIGMORPHON 2022 shared tasks on morpheme segmentation and inflection generation. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 212–219, Seattle, Washington. Association for Computational Linguistics.

Adam Wiemerslage, Miikka Silfverberg, Changbing Yang, Arya McCarthy, Garrett Nicolai, Eliana Colunga, and Katharina Kann. 2022. Morphological processing of low-resource languages: Where we are and what's next. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 988–1007, Dublin, Ireland. Association for Computational Linguistics.

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2021. Applying the transformer to character-level transduction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1901–1907, Online. Association for Computational Linguistics.

A Experiment 1

Algorithm 1 Our splitting algorithm for Expt. 1

Require: LEMMAS (all lemmas in data), TRIPLES (all triples in data), `trainsize`, `ftunesize`, `testsize`

OVERLAPLEMMAS $\leftarrow$ `random`(LEMMAS, $\lfloor 0.5\times$ | LEMMAS |$\rfloor$)
OVERLAPTRIPLES $\leftarrow$ $\{(\mathcal{L}, F, I) \in$ TRIPLES $\mid \mathcal{L} \in$ OVERLAPLEMMAS$\}$
while |OVERLAPTRIPLES| $<$ $\lceil$`trainsize` + `ftunesize` + $0.5 \times$ `testsize`$\rceil$ **do**
  OVERLAPLEMMAS $\leftarrow$ OVERLAPLEMMAS $\cup$ `random`(LEMMAS $\setminus$ OVERLAPLEMMAS, 1)
  OVERLAPTRIPLES $\leftarrow$ $\{(\mathcal{L}, F, I) \in$ TRIPLES $\mid \mathcal{L} \in$ OVERLAPLEMMAS$\}$
end while
TRAIN $\leftarrow$ `random`(OVERLAPTRIPLES, `trainsize`)
FINETUNE $\leftarrow$ `random`(OVERLAPTRIPLES $\setminus$ TRAIN, `ftunesize`)
TESTO $\leftarrow$ `random`(OVERLAPTRIPLES $\setminus$ (TRAIN$\cup$FINETUNE), $\lfloor 0.5\times$ `testsize`$\rfloor$)
TESTN $\leftarrow$ `random`(TRIPLES $\setminus$ OVERLAPTRIPLES, $\lceil 0.5 \times$ `testsize`$\rceil$)
TEST $\leftarrow$ TESTO $\cup$ TESTN
return TRAIN, FINETUNE, TEST

FAMILY	LANGUAGE	
Germanic	Old English	(ang)
	Danish	(dan)
	German	(deu)
	English	(eng)
	Icelandic	(isl)
	Dutch	(nld)
	Norwegian Nyorsk	(nno)
	Norwegian Bokmål	(nob)
	Swedish	(swe)
Indo-Aryan	Hindi	(hin)
	Sanskrit	(san)
	Urdu	(urd)
Iranian	Persian	(fas)
Oto-Manguean	San Pedro Amuzgos	(azg)
	Mezquital Otomi	(ote)
	Sierra Otomi	(otm)
	Chichimeca-Jonaz	(pei)
Romance	Catalan	(cat)
	Middle French	(frm)
	Galacian	(glg)
	Venetian	(vec)
Turkish	Bashkir	(bak)
	Kazakh	(kaz)
	Turkmen	(tuk)
	Uzbek	(uzb)
Uralic	Estonian	(est)
	Finnish	(fin)
	Komi-Zyrian	(kpv)
	Karelian	(krl)
	Moksha	(mdf)
	Meadow Mari	(mhr)
	Erzya	(myv)
	Livvi	(olo)
	Northern Sami	(sme)
	Udmurt	(udm)
	Veps	(vep)

Table 6: The 36 SIGMORPHON 2020 languages used in Experiment 1.

Family	Seen	Unseen	Difference
Germanic	86.422	79.230	-7.193
Indo-Aryan	97.176	79.724	-17.451
Iranian	99.673	76.447	-23.227
Oto-Manguean	90.339	54.920	-35.419
Romance	99.172	93.600	-5.572
Turkic	95.098	83.532	-11.567
Uralic	92.064	87.013	-5.052
Macro-average	94.278	79.209	-15.068

Table 7: Experiment 1 accuracy by language family, averaged across neural architectures & seeds. This can be compared to Table 2 in Goldman et al. (2022), though the family groupings in their paper are different than those in their data and the SIGMORPHON data. Rather than attempt to replicate this grouping without sufficient information to do so, we instead report grouping by the language families in Goldman et al. (2022)'s data splits.

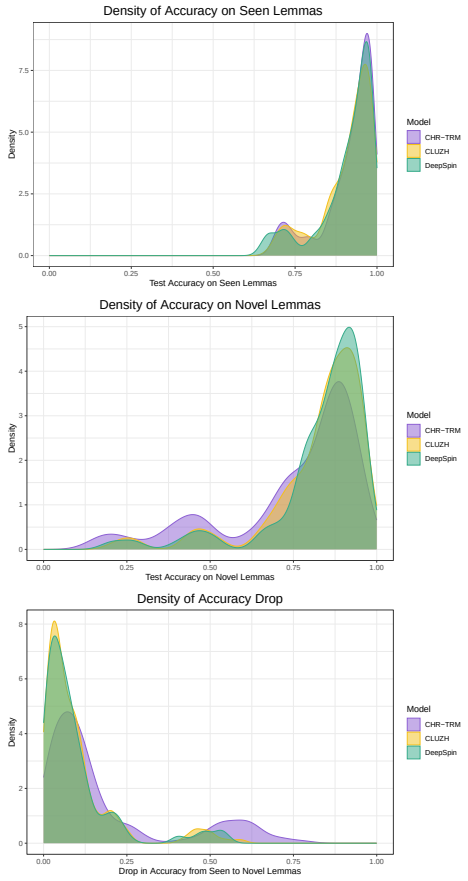

Figure 7: SEEN and NOVEL accuracy and DROP are beta distributed in Experiment 1.

A.1 Beta regression models for SEEN lemmas

The following provides the full R output of the SEEN $\sim$ LEMMAS beta regression models in Experiment 1 (§3.4).

CHR-TRM

```
Call: gam(formula = scaled_seen ~ log(train_lemmas),
    family = betar(link = "logit"),
    data = wu_res_clean)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.7781 -0.7678  0.0000  0.0000  0.7016
```

Lang	Seen			Novel			Drop		
	Chr-Trm	Cluzh	DeepSpin	Chr-Trm	Cluzh	DeepSpin	Chr-Trm	Cluzh	DeepSpin
ang	72.873	72.150	72.631	46.236	49.739	49.582	-26.637	-22.411	-23.049
azg	93.980	93.060	92.480	23.400	45.020	43.100	-70.580	-48.040	-49.380
bak	99.640	99.440	99.600	92.460	93.440	93.380	-7.180	-6.000	-6.220
cat	98.880	98.740	98.500	96.160	96.980	96.440	-2.720	-1.760	-2.060
dan	71.591	71.992	66.814	64.116	66.168	66.646	-7.475	-5.825	-0.168
deu	94.770	95.236	94.516	88.576	89.788	89.635	-6.194	-5.448	-4.881
eng	94.847	95.062	94.868	94.783	94.906	95.364	-0.064	-0.156	0.496
est	93.980	93.040	93.920	75.680	77.760	80.280	-18.300	-15.280	-13.640
fas	99.900	99.340	99.780	50.880	90.020	88.440	-49.020	-9.320	-11.340
fin	97.443	96.238	96.972	92.917	91.703	92.642	-4.527	-4.535	-4.330
frm	99.480	98.920	99.080	89.640	96.460	96.160	-9.840	-2.460	-2.920
glg	99.580	99.120	99.280	88.880	94.820	93.960	-10.700	-4.300	-5.320
hin	100.000	99.960	100.000	43.800	95.440	95.240	-56.200	-4.520	-4.760
isl	91.883	92.576	90.887	81.373	83.005	81.531	-10.511	-9.571	-9.356
kaz	98.060	97.580	98.020	39.400	78.320	79.100	-58.660	-19.260	-18.920
kpv	95.108	93.541	94.138	92.876	91.959	93.169	-2.231	-1.582	-0.969
krl	97.345	93.987	97.185	87.211	84.906	89.064	-10.134	-9.081	-8.121
mdf	90.467	87.596	89.890	88.012	86.236	87.894	-2.455	-1.360	-1.996
mhr	88.850	85.997	86.628	85.631	83.434	84.338	-3.220	-2.564	-2.289
myv	91.041	88.428	90.775	90.909	89.744	91.240	-0.133	1.316	0.465
nld	98.305	97.977	97.958	92.958	92.605	93.174	-5.347	-5.371	-4.783
nno	87.080	87.512	86.896	74.293	74.390	79.770	-12.788	-13.121	-7.126
nob	73.347	75.938	72.768	74.934	74.638	79.460	1.586	-1.300	6.692
olo	89.789	89.986	90.338	88.761	88.098	89.441	-1.028	-1.887	-0.897
ote	98.498	98.097	98.338	83.882	86.339	86.858	-14.615	-11.758	-11.479
otm	97.177	96.997	96.797	71.843	76.480	77.297	-25.335	-20.517	-19.500
pei	74.180	73.720	70.740	17.360	24.200	23.260	-56.820	-49.520	-47.480
san	92.520	92.320	91.280	78.180	83.580	82.820	-14.340	-8.740	-8.460
sme	98.594	97.689	97.931	90.784	88.873	89.073	-7.810	-8.816	-8.858
swe	94.074	94.905	93.951	86.758	87.056	87.718	-7.316	-7.849	-6.233
tuk	85.760	86.340	84.200	73.760	85.600	85.360	-12.000	-0.740	1.160
udm	97.250	95.753	96.656	95.822	94.748	95.685	-1.428	-1.005	-0.971
urd	99.520	99.460	99.520	42.340	97.620	98.500	-57.180	-1.840	-1.020
uzb	97.340	99.180	96.020	89.160	96.780	95.620	-8.180	-2.400	-0.400
vec	99.580	99.380	99.520	84.120	94.660	94.920	-15.460	-4.720	-4.600
vep	80.656	79.728	81.182	70.419	70.456	71.650	-10.237	-9.272	-9.532

Table 8: Full results by language and neural architecture, averaged across our 5 seeds, for Experiment 1.

```
(Dispersion Parameter for Beta regression family
taken to be 0.1221)

    Null Deviance: -56.1663 on 179 degrees of freedom
Residual Deviance: -67.2103 on 178 degrees of freedom
AIC: -440724.1

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                  Df  Sum Sq Mean Sq F value   Pr(>F)
log(train_lemmas)   1  0.8451 0.84509  6.9219 0.009261 **
Residuals         178 21.7319 0.12209
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

      (Intercept) log(train_lemmas)
        3.1273269        -0.1189933
```

Cluzh

```
Call: gam(formula = scaled_seen ~ log(train_lemmas),
    family = betar(link = "logit"),
    data = cluzh_res_clean)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.7758 -0.9112  0.0000  0.0000  0.6348

(Dispersion Parameter for Beta regression family
taken to be 0.116)

    Null Deviance: -42.169 on 179 degrees of freedom
Residual Deviance: -50.6733 on 178 degrees of freedom
AIC: -425049.3

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                  Df  Sum Sq Mean Sq F value   Pr(>F)
log(train_lemmas)   1  1.0235 1.02345  8.8191 0.003392 **
Residuals         178 20.6568 0.11605
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

      (Intercept) log(train_lemmas)
        3.1582321        -0.1301994
```

DeepSpin

```
Call: gam(formula = scaled_seen ~ log(train_lemmas),
    family = betar(link = "logit"),
    data = ds_res_clean)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.7519 -0.7813  0.0000  0.0000  0.6982

(Dispersion Parameter for Beta regression family
taken to be 0.1325)

    Null Deviance: -49.5337 on 179 degrees of freedom
Residual Deviance: -60.7565 on 178 degrees of freedom
AIC: -424201.9

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
```

```
                 Df  Sum Sq Mean Sq F value Pr(>F)
log(train_lemmas)   1  0.6084 0.60839  4.5909 0.0335 *
Residuals         178 23.5891 0.13252
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

      (Intercept) log(train_lemmas)
       2.90407660       -0.09775024
```

A.2 Beta regression models for NOVEL lemmas

The following provides the full R output of the NOVEL ∼ LEMMAS beta regression models in Experiment 1 (§3.4).

CHR-TRM

```
Call: gam(formula = scaled_novel ~ log(train_lemmas),
    family = betar(link = "logit"),
    data = wu_res_clean)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.3688 -0.9631  0.0000  0.0000  0.9252

(Dispersion Parameter for Beta regression
family taken to be 0.2352)

    Null Deviance: 51.5501 on 179 degrees of freedom
Residual Deviance: 79.1182 on 178 degrees of freedom
AIC: -193804.2

Number of Local Scoring Iterations: 5

Anova for Parametric Effects
                 Df Sum Sq Mean Sq F value    Pr(>F)
log(train_lemmas)   1 29.862 29.8618  126.96 < 2.2e-16 ***
Residuals         178 41.867  0.2352
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

      (Intercept) log(train_lemmas)
       -2.5611164         0.5577768
```

CLUZH

```
Call: gam(formula = scaled_novel ~ log(train_lemmas),
    family = betar(link = "logit"),
    data = cluzh_res_clean)
Deviance Residuals:
     Min       1Q   Median       3Q      Max
-1.51904 -0.97321  0.00000  0.09055  0.81805

(Dispersion Parameter for Beta regression family
taken to be 0.284)

    Null Deviance: 30.1758 on 179 degrees of freedom
Residual Deviance: 34.5711 on 178 degrees of freedom
AIC: -261196.3

Number of Local Scoring Iterations: 5

Anova for Parametric Effects
                 Df Sum Sq Mean Sq F value  Pr(>F)
log(train_lemmas)   1  1.888 1.88793  6.6468 0.01074 *
Residuals         178 50.558 0.28404
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

      (Intercept) log(train_lemmas)
        0.5417008         0.1484194
```

DEEPSPIN

```
Call: gam(formula = scaled_novel ~ log(train_lemmas),
    family = betar(link = "logit"),
    data = ds_res_clean)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.5407 -0.9962  0.0000  0.0000  0.8116

(Dispersion Parameter for Beta regression family
taken to be 0.2838)

    Null Deviance: 26.5794 on 179 degrees of freedom
Residual Deviance: 33.2525 on 178 degrees of freedom
AIC: -269455.6

Number of Local Scoring Iterations: 5

Anova for Parametric Effects
                 Df Sum Sq Mean Sq F value   Pr(>F)
log(train_lemmas)   1  3.002 3.00205  10.579 0.001368 **
Residuals         178 50.510 0.28377
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

      (Intercept) log(train_lemmas)
        0.3111430         0.1895828
```

A.3 Beta regression models for DROP

The following provides the full R output of the DROP ∼ LEMMAS beta regression models in Experiment 1 (§3.4).

CHR-TRM

```
all: gam(formula = scaled_diff ~ log(train_lemmas),
    family = betar(link = "logit"),
    data = wu_res_clean)
Deviance Residuals:
     Min       1Q   Median       3Q      Max
-0.96697 -0.83717  0.05881  0.92129  1.11801

(Dispersion Parameter for Beta regression family
taken to be 0.14)

    Null Deviance: 117.7257 on 179 degrees of freedom
Residual Deviance: 128.5483 on 178 degrees of freedom
AIC: 43583.62

Number of Local Scoring Iterations: 4

Anova for Parametric Effects
                 Df Sum Sq Mean Sq F value    Pr(>F)
log(train_lemmas)   1 32.169  32.169  229.69 < 2.2e-16 ***
Residuals         178 24.929   0.140
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

      (Intercept) log(train_lemmas)
        2.9609546        -0.5237525
```

CLUZH

```
Call: gam(formula = scaled_diff ~ log(train_lemmas),
    family = betar(link = "logit"),
    data = cluzh_res_clean)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-0.9184 -0.7647 -0.6405  0.9666  1.1398

(Dispersion Parameter for Beta regression family
taken to be 0.0997)

    Null Deviance: 128.5974 on 179 degrees of freedom
Residual Deviance: 131.0342 on 178 degrees of freedom
AIC: 81136.68

Number of Local Scoring Iterations: 4

Anova for Parametric Effects
                 Df  Sum Sq Mean Sq F value    Pr(>F)
log(train_lemmas)   1  3.5969  3.5969  36.089 1.038e-08 ***
Residuals         178 17.7409  0.0997
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

      (Intercept) log(train_lemmas)
        0.2686925        -0.1737422
```

DEEPSPIN

```
Call: gam(formula = scaled_diff ~ log(train_lemmas),
    family = betar(link = "logit"),
    data = ds_res_clean)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-0.9166 -0.7448 -0.5476  0.9549  1.1635
```

```
(Dispersion Parameter for Beta regression family
taken to be 0.1021)

    Null Deviance: 125.1004 on 179 degrees of freedom
Residual Deviance: 128.2078 on 178 degrees of freedom
AIC: 84275.73

Number of Local Scoring Iterations: 4

Anova for Parametric Effects
                  Df  Sum Sq Mean Sq F value    Pr(>F)
log(train_lemmas)   1  4.3868  4.3868  42.981 5.781e-10 ***
Residuals         178 18.1674  0.1021
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

      (Intercept) log(train_lemmas)
        0.3609071        -0.1935463
```

B Experiment 2

FAMILY	LANGUAGE	
Niger-Congo	Akan	(aka)
	Gã	(gaa)
	Kongo	(kon)
	Lingala	(lin)
	Luganda	(lug)
	Chewa	(nya)
	Shona	(sna)
	Sotho	(sot)
	Swahili	(swa)
	Zulu	(zul)
Romance	Asturian	(ast)
	Catalan	(cat)
	Middle French	(frm)
	Friulian	(fur)
	Galacian	(glg)
	Ladin	(lld)
	Venetian	(vec)
	Anglo-Normal	(xno)
Uralic	Estonian	(est)
	Finnish	(fin)
	Ingrian	(izh)
	Komi-Zyrian	(kpv)
	Karelian	(krl)
	Livonian	(liv)
	Ludic	(lud)
	Moksha	(mdf)
	Meadow Mari	(mhr)
	Erzya	(myv)
	Livvi	(olo)
	Northern Sami	(sme)
	Udmurt	(udm)
	Veps	(vep)
	Votic	(vot)
	Võro	(vro)

Table 9: The 34 languages used to replicate the findings of Goldman et al. (2022) in Experiment 2.

B.1 Beta regression models for SIGMORPHON

The following provides the full R output of the multi-predictor beta regression models trained to predict SIGMORPHON accuracy in Experiment 2 (§4.2).

CHR-TRM

```
Call: gam(formula = SIGMORPHON_scaled_acc ~
    log(SIGMORPHON_train_size) + log(SIGMORPHON_train_lemmas),
    family = betar(link = "logit"), data = CHRTRM)
Deviance Residuals:
```

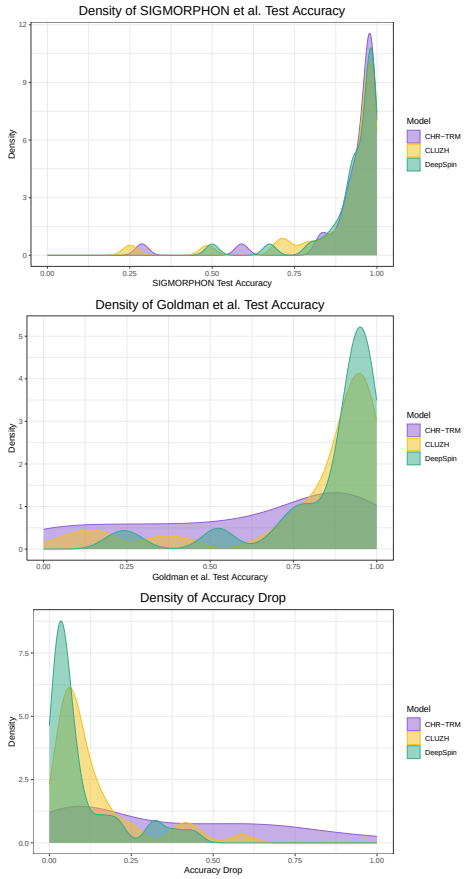

Figure 8: Replicated GOLDMAN and SIGMORPHON accuracy as well as DROP are beta distributed in Experiment 2.

```
    Min      1Q  Median      3Q     Max
-1.9418 -0.7337  0.0000  0.0000  0.0000

(Dispersion Parameter for Beta regression family
taken to be 0.1921)

    Null Deviance: -27.6167 on 33 degrees of freedom
Residual Deviance: -15.6955 on 31 degrees of freedom
AIC: -17287.93

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                              Df Sum Sq Mean Sq F value   Pr(>F)
log(SIGMORPHON_train_size)    1 2.1633 2.16325 11.2586 0.002106 **
log(SIGMORPHON_train_lemmas)  1 1.6542 1.65420  8.6093 0.006245 **
Residuals                    31 5.9564 0.19214
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Intercept)  log(SIGMORPHON_train_size) log(SIGMORPHON_train_lemmas)
-0.5531629                 0.7623518                -0.5661172
```

CLUZH

```
Call: gam(formula = SIGMORPHON_scaled_acc ~
    log(SIGMORPHON_train_size) + log(SIGMORPHON_train_lemmas),
    family = betar(link = "logit"), data = CLUZH)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.9034 -1.0654  0.0000  0.0000  0.4304

(Dispersion Parameter for Beta regression family
taken to be 0.2247)

    Null Deviance: -29.4498 on 33 degrees of freedom
Residual Deviance: -17.2346 on 31 degrees of freedom
```

	GOLDMAN			SIGMORPHON			DIFFERENCE		
Lang	CHR-TRM	CLUZH	DEEPSPIN	CHR-TRM	CLUZH	DEEPSPIN	CHR-TRM	CLUZH	DEEPSPIN
aka	100.000	99.607	100.000	30.038	99.487	94.737	-69.962	-0.120	-5.263
ast	99.314	98.49	99.726	67.645	95.664	99.133	-31.669	-2.826	-0.593
cat	99.778	99.717	99.778	97.507	97.823	97.44	-2.271	-1.894	-2.338
est	95.194	94.317	94.566	79.586	81.069	82.214	-15.608	-13.248	-12.351
fin	99.56	99.746	99.782	94.503	94.147	95.156	-5.057	-5.6	-4.626
frm	99.744	99.488	99.701	94.056	95.99	96.921	-5.688	-3.498	-2.781
fur	99.741	99.353	99.806	70.716	100.000	99.936	-29.025	0.647	0.130
gaa	99.408	100.000	100.000	53.216	100.000	100.000	-46.192	0.000	0.000
glg	99.840	99.680	99.782	95.861	97.037	97.545	-3.98	-2.643	-2.237
izh	88.839	79.018	87.500	26.484	41.553	51.142	-62.355	-37.465	-36.358
kon	98.077	98.718	94.231	88.125	98.750	95.000	-9.952	0.032	0.769
kpv	96.956	96.43	96.805	94.085	93.645	93.798	-2.871	-2.785	-3.007
krl	99.264	99.071	99.475	90.19	88.502	91.257	-9.075	-10.569	-8.218
lin	100.000	100.000	100.000	56.250	100.000	100.000	-43.750	0.000	0.000
liv	96.509	94.638	95.012	50.422	80.141	77.465	-46.086	-14.498	-17.548
lld	99.586	99.034	99.862	60.173	93.75	98.338	-39.413	-5.284	-1.524
lud	28.049	24.39	50.000	1.835	4.128	20.642	-26.214	-20.262	-29.358
lug	90.583	88.434	90.174	18.561	67.153	70.699	-72.022	-21.281	-19.475
mdf	93.391	92.403	93.286	91.742	90.669	91.489	-1.649	-1.734	-1.797
mhr	92.957	93.288	93.511	86.974	85.667	86.18	-5.983	-7.621	-7.331
myv	94.13	93.125	93.758	94.162	93.489	93.974	0.033	0.364	0.216
nya	100.000	99.883	100.000	75.858	100.000	100.000	-24.142	0.117	0.000
olo	94.207	94.726	95.246	91.506	91.074	92.516	-2.701	-3.653	-2.730
sme	99.673	99.537	99.737	94.496	93.053	95.537	-5.177	-6.484	-4.200
sna	100.000	99.781	100.000	26.274	90.784	93.726	-73.726	-8.996	-6.275
sot	97.980	97.980	100.000	0.000	100.000	100.000	-97.980	2.020	0.000
swa	100.000	100.000	100.000	42.959	99.898	100.000	-57.041	-0.102	0.000
udm	98.492	98.52	98.682	96.413	95.596	96.366	-2.079	-2.924	-2.315
vec	99.684	99.512	99.570	91.591	97.310	97.225	-8.093	-2.202	-2.344
vep	83.71	83.056	80.728	77.947	77.197	78.807	-5.764	-5.859	-1.922
vot	85.053	73.31	84.698	4.895	32.517	53.846	-80.158	-40.792	-30.851
vro	59.223	48.544	67.961	0.000	13.542	26.042	-59.223	-35.002	-41.92
xno	96.078	70.588	94.118	0.000	15.686	94.118	-96.078	-54.902	0.000
zul	92.308	87.18	89.744	32.222	75.556	75.556	-60.085	-11.624	-14.188

Table 10: Full results by language and neural architecture for Experiment 2.

```
AIC: -15893.34

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                           Df Sum Sq Mean Sq F value    Pr(>F)
log(SIGMORPHON_train_size)   1 4.0506  4.0506 18.0264 0.0001837 ***
log(SIGMORPHON_train_lemmas)  1 0.5878  0.5878  2.6159 0.1159300
Residuals                    31 6.9659  0.2247
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Intercept)  log(SIGMORPHON_train_size) log(SIGMORPHON_train_lemmas)
-1.1303584                 0.6168127                -0.3049881
```

DEEPSPIN

```
Call: gam(formula = SIGMORPHON_scaled_acc ~
    log(SIGMORPHON_train_size) + log(SIGMORPHON_train_lemmas),
    family = betar(link = "logit"), data = DeepSpin)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.9276 -0.7471  0.0000  0.0000  0.5294

(Dispersion Parameter for Beta regression family
taken to be 0.1431)

    Null Deviance: -24.6582 on 33 degrees of freedom
Residual Deviance: -18.3863 on 31 degrees of freedom
AIC: -17490.71

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                           Df Sum Sq Mean Sq F value   Pr(>F)
log(SIGMORPHON_train_size)   1 1.4196 1.41962  9.9226 0.003603 **
log(SIGMORPHON_train_lemmas)  1 0.9939 0.99394  6.9472 0.012994 *
Residuals                    31 4.4352 0.14307
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Intercept)  log(SIGMORPHON_train_size) log(SIGMORPHON_train_lemmas)
0.08735762                0.59994250               -0.43981367
```

B.2 Beta regression models for GOLDMAN

The following provides the full R output of the multi-predictor beta regression models trained to predict GOLDMAN accuracy in Experiment 2 (§4.2).

CHR-TRM

```
Call: gam(formula = Goldman_scaled_acc ~ log(Goldman_train_lemmas) +
    log(Goldman_train_size), family = betar(link = "logit"),
    data = CHRTRM)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.7181 -0.9309  0.0000  0.0000  1.3871

(Dispersion Parameter for Beta regression family taken to be 0.3214)

    Null Deviance: -1.0739 on 33 degrees of freedom
Residual Deviance: 9.2018 on 31 degrees of freedom
AIC: -3878.832

Number of Local Scoring Iterations: 6

Anova for Parametric Effects
                           Df  Sum Sq Mean Sq F value    Pr(>F)
log(Goldman_train_lemmas)  1 14.2329 14.2329 44.2893 1.932e-07 ***
log(Goldman_train_size)    1  0.7122  0.7122  2.2163    0.1467
```

```
Residuals                 31  9.9622  0.3214
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Intercept) log(Goldman_train_lemmas)   log(Goldman_train_size)
-4.5197092                 0.5323991                 0.2583534
```

CLUZH

```
Call: gam(formula = Goldman_scaled_acc ~ log(Goldman_train_size) +
    log(Goldman_train_lemmas), family = betar(link = "logit"),
    data = CLUZH)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.6752 -0.8953  0.0000  0.0000  0.7093

(Dispersion Parameter for Beta regression family taken to be 0.4671)

    Null Deviance: -22.2953 on 33 degrees of freedom
Residual Deviance: -11.0956 on 31 degrees of freedom
AIC: -10675.54

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                        Df  Sum Sq Mean Sq F value    Pr(>F)
log(Goldman_train_size)   1 7.0203 7.0203 15.0303 0.0005137 ***
log(Goldman_train_lemmas)  1  0.0050  0.0050  0.0107 0.9182216
Residuals                 31 14.4794  0.4671
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Intercept)   log(Goldman_train_size) log(Goldman_train_lemmas)
-1.83118611                0.41435242               -0.02250918
```

DEEPSPIN

```
Call: gam(formula = Goldman_scaled_acc ~ log(Goldman_train_size) +
    log(Goldman_train_lemmas), family = betar(link = "logit"),
    data = DeepSpin)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.6214 -0.9821  0.0000  0.0000  0.7569

(Dispersion Parameter for Beta regression family taken to be 0.3064)

    Null Deviance: -19.4575 on 33 degrees of freedom
Residual Deviance: -12.6952 on 31 degrees of freedom
AIC: -12350.04

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                        Df Sum Sq Mean Sq F value   Pr(>F)
log(Goldman_train_size)    1 3.5682  3.5682 11.6462 0.001809 **
log(Goldman_train_lemmas)  1 0.8896  0.8896  2.9036 0.098388 .
Residuals                 31 9.4980  0.3064
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Intercept)   log(Goldman_train_size) log(Goldman_train_lemmas)
-1.0403182                 0.5326597                -0.3004978
```

B.3 Beta regression models for DROP

The following provides the full R output of the multi-predictor beta regression models trained to predict DROP from SIGMORPHON to GOLDMAN accuracy in Experiment 2 (§4.2).

CHR-TRM

```
Call: gam(formula = scaled_drop ~ log(Goldman_train_lemmas)
    + log(-1 * train_lemma_diff_raw) + log(Goldman_train_size),
    family = betar(link = "logit"), data = CHRTRM)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.0309  0.0000  0.6883  1.1058  2.1611

(Dispersion Parameter for Beta regression family
taken to be 0.1944)

    Null Deviance: -4.2629 on 33 degrees of freedom
Residual Deviance: 20.2587 on 30 degrees of freedom
AIC: 956.8627

Number of Local Scoring Iterations: 6

Anova for Parametric Effects
                             Df  Sum Sq Mean Sq F value    Pr(>F)
log(Goldman_train_lemmas)     1 10.9621 10.9621 56.3870 2.267e-08 ***
log(-1 * train_lemma_diff_raw)  1  0.2024  0.2024  1.0412   0.3157
log(Goldman_train_size)       1  0.1658  0.1658  0.8530   0.3631
Residuals                    30  5.8323  0.1944
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Intercept)    log(Goldman_train_lemmas) log(-1 * train_lemma_diff_raw)
3.7697667                 -1.2338483                  0.1499576
log(Goldman_train_size)
0.1537048
```

CLUZH

```
Call: gam(formula = scaled_drop ~ log(Goldman_train_lemmas)
    + log(-1 * train_lemma_diff_raw) + log(Goldman_train_size),
    family = betar(link = "logit"), data = CLUZH)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-0.7097  0.0000  0.3350  1.0758  1.4194

(Dispersion Parameter for Beta regression family
taken to be 0.1584)

    Null Deviance: -3.114 on 33 degrees of freedom
Residual Deviance: 4.0005 on 30 degrees of freedom
AIC: 5837.224

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                             Df Sum Sq Mean Sq F value    Pr(>F)
log(Goldman_train_lemmas)     1 2.5226 2.52261 15.9285 0.0003913 ***
log(-1 * train_lemma_diff_raw)  1 0.0524 0.05242  0.3310 0.5693799
log(Goldman_train_size)       1 0.0128 0.01279  0.0808 0.7782086
Residuals                    30 4.7511 0.15837
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Intercept)    log(Goldman_train_lemmas) log(-1 * train_lemma_diff_raw)
-0.63896835               -0.08228353                -0.26488998
log(Goldman_train_size)
0.04563225
```

DEEPSPIN

```
Call: gam(formula = scaled_drop ~ log(-1 * train_lemma_diff_raw) +
    log(Goldman_train_size) + log(Goldman_train_lemmas),
    family = betar(link = "logit"), data = DeepSpin)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-0.8752  0.0000  0.0000  0.9393  1.5220

(Dispersion Parameter for Beta regression family taken to be 0.1435)

    Null Deviance: -16.5461 on 33 degrees of freedom
Residual Deviance: -12.7918 on 30 degrees of freedom
AIC: 6982.896

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                             Df Sum Sq Mean Sq F value   Pr(>F)
log(-1 * train_lemma_diff_raw)  1 1.6345 1.63447 11.3933 0.002052 **
log(Goldman_train_size)       1 0.1767 0.17666  1.2314 0.275945
log(Goldman_train_lemmas)     1 0.3227 0.32272  2.2495 0.144106
Residuals                    30 4.3038 0.14346
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Intercept) log(-1 * train_lemma_diff_raw)     log(Goldman_train_size)
-1.0177739               -0.5704532                -0.1261425
log(Goldman_train_lemmas)
0.4467069
```