A Supplemental Material

A.1 Model and Training Details

For all of the bilingual baselines and multilingual model that we investigate, we use the Transformer (Vaswani et al., 2017) architecture. In particular, we use the Transformer Big model containing 375M parameters in (Chen et al., 2018). For multilingual models, we share all parameters across language pairs including softmax layer in input/output word embeddings.

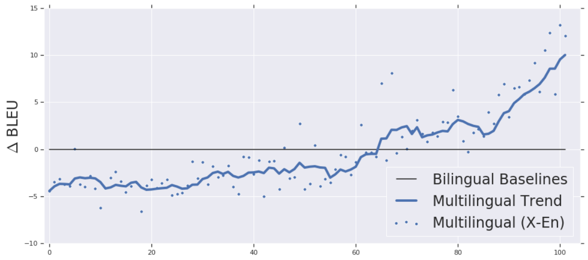
During training, we use a temperature based data sampling strategy, similar to the strategy used to train the multilingual models in (Arivazhagan et al., 2019b). That is, if p_L is the probability that a sentence in the corpus belongs to language pair L, we sample from a distribution where the probability of sampling from L is proportional to ${p_L}^{\frac{1}{T}}$. All the experiments in this paper are performed on a model trained with a sampling temperature $T = 5$.

For the vocabulary, we use a Sentence Piece Model (SPM) (Kudo and Richardson, 2018) with 64k tokens shared on both the encoder and decoder side. To learn a joint SPM model given our imbalanced dataset, we followed the temperature based sampling strategy with a temperature of $T = 5$.

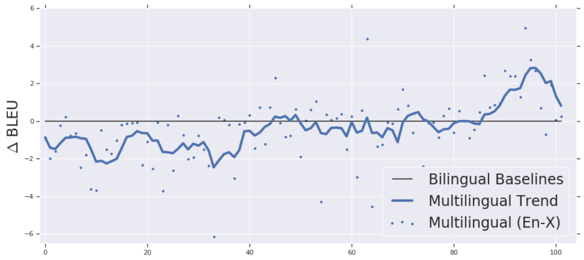
Finally, all models are optimized using Adafactor optimizer (Shazeer and Stern, 2018) with momentum factorization and a per-parameter norm clipping threshold of 1.0. During optimization, we followed a learning rate of a learning rate of 3.0, with 40K warm-up steps for the schedule, which is decayed with the inverse square root of the number of training steps after warm-up. BLEU scores presented in this paper are calculated on true-cased output and references, where we used mteval-v13a.pl script from Moses.

A.2 Baselines and Multilingual BLEU Scores

To assess the quality of our single massively multilingual model trained on 103 languages, we trained bilingual baselines using the same training data, with models that are comparable in their size. For high resource languages, we trained identical architecture models (Transformer-Big) and only for a few low-resource languages we trained smaller models with heavy regularization (Transformer-Base). Results are shared in Figure 7. Note that, x-axes correspond to a different language pairs sorted with respect to the available training data and y-axes correspond to the divergence from the baseline BLEU scores. For each language pair, the BLEU scores are calculated on the test set that is specific for that language pair. From Figure 7 it is clear that our massively multilingual model is dramatically better on low-resource languages (right-most portion of both panels) with some regression on high-resource languages (left-most portion of the panels). We provide the comparison with baselines to ground our analysis of massively multilingual model, which is competitive with bilingual baselines in quality.



(a) Comparison of X-En pairs with baselines.



(b) Comparison of En-X pairs with baselines.

Figure 7: Trendlines depicting translation performance of the massively multilingual model (blue curves) compared to bilingual baselines (solid black lines). From left to right, languages are arranged in decreasing order of available training data (high-resource to low-resource). y-axis depicts the BLEU score relative to the bilingual baseline trained on the corresponding language pair. Top panel for Any-to-English pairs and bottom panel for English-to-Any pairs.

A.3 CCA for Misaligned Sequences

In Section 2.2, we discussed how the mean pooling strategy that we use is more suitable for our problem, where we use SVCCA to compare unaligned sequences. In this section, we attempt to replicate some results in the paper using a token-level CCA strategy, and discuss the differences in our results. In the mean pooling strategy, each datapoint forming the subspace representing a language's encoding for a given layer is the mean of all timestep activations in a single sentence. On the other hand, in what we refer to as a token-level strategy, each data point is the activation of a timestep, with no

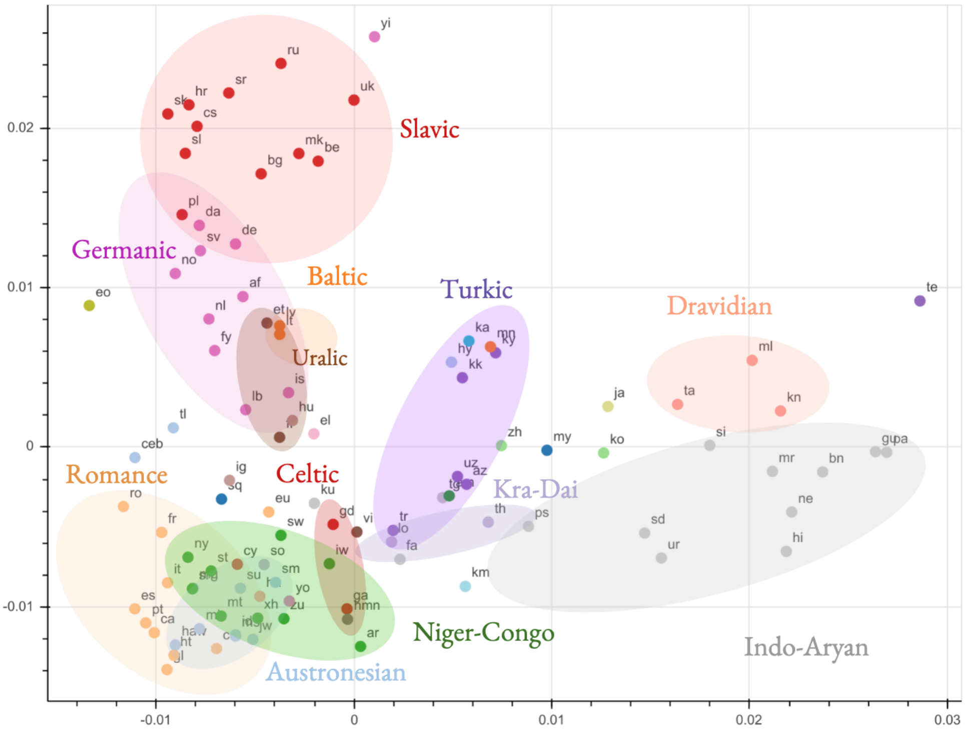


Figure 8: Top layer of the encoder for En-X language pairs using token level SVCCA as a similarity measure.

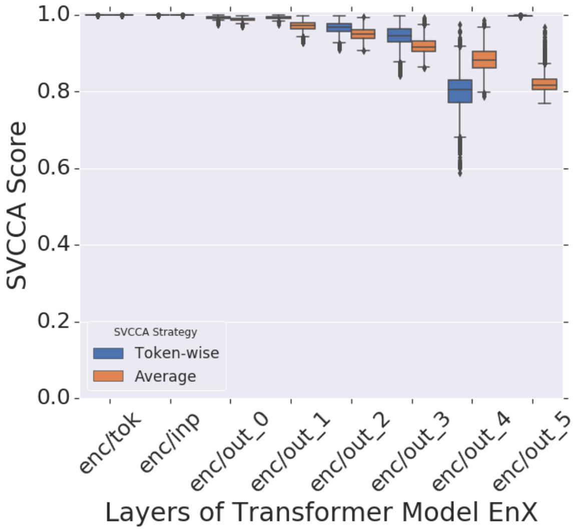


Figure 9: The change in distribution of pairwise SVCCA scores using our pooling strategy and a naive token-wise strategy between English-to-Any language pairs across the encoder layers of a multilingual NMT model. We see that while the encoder representations diverge in both cases, the top layer of the encoder does not seem to show any divergence for the token-wise strategy and is a possible artifact of the strategy.

differentiation between different sentences or positions.

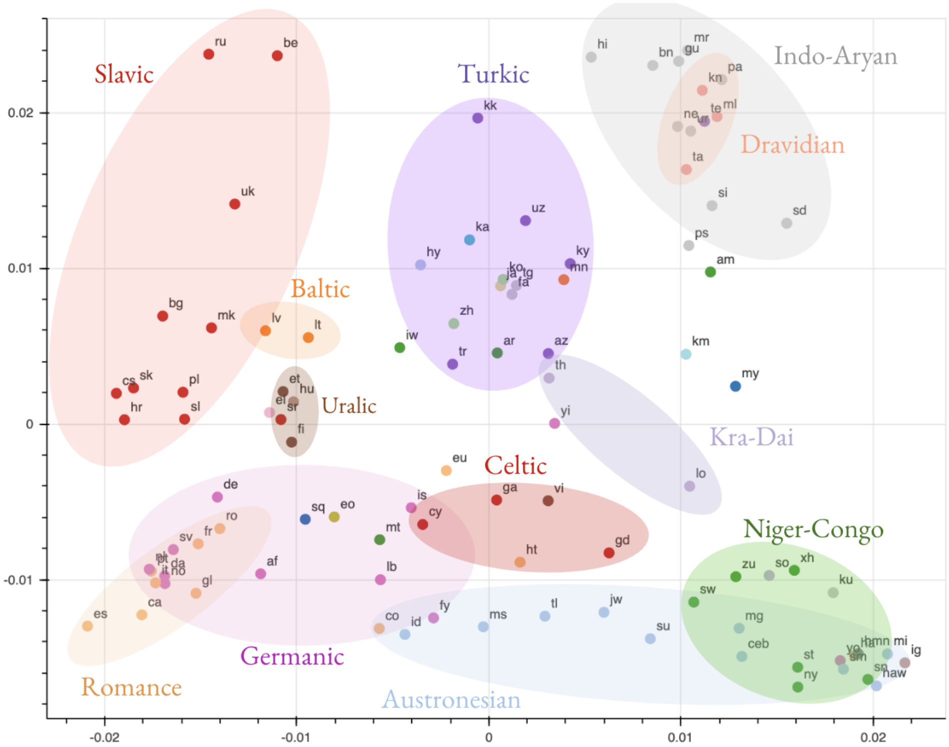
In Figure 8, we plot the cluster formed by the SVCCA scores of English-to-Any language pairs using the method described in Section 3. Our data is unaligned for compared other components of our experiment, so we do not discuss those results. While we do see some amount of clustering according to linguistic similarity, the clusters are less separated than in Figure 10. We also compare the distributions of pairwise SVCCA scores using our pooling strategy and a naive token-wise strategy between English-to-Any language pairs across layers of the encoder. We see similar trends upto the top layer of the encoder - this could possibly be an artifact of the naive strategy.

A.4 Additional Clustering Visualization

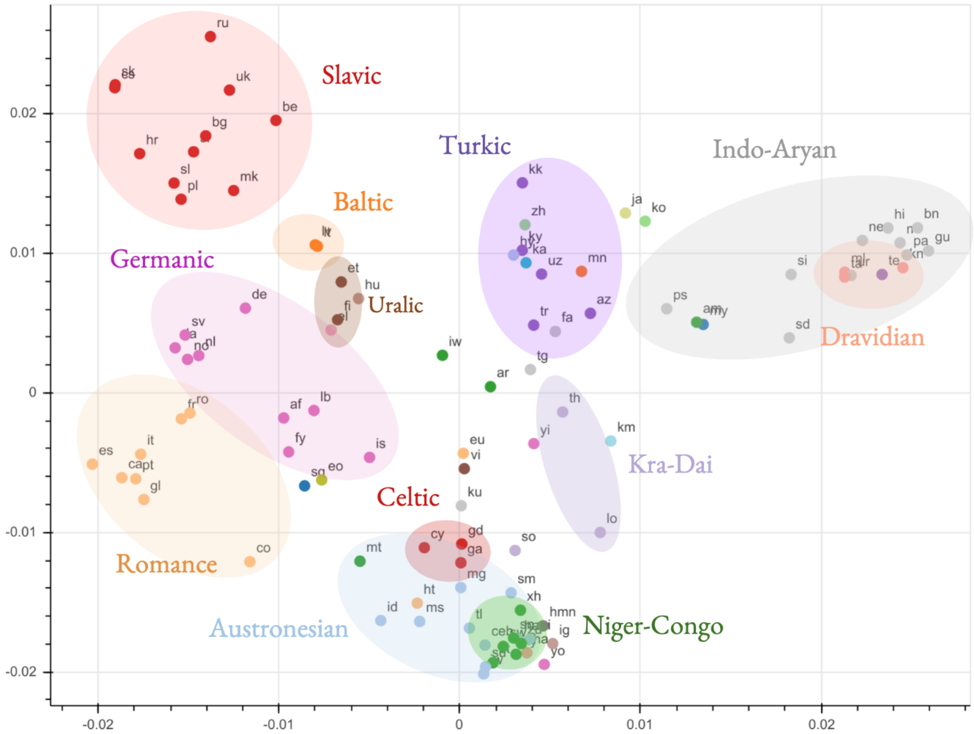
Here, we plot the clusters formed by all languages pairs, color coded by linguistic subfamily for the top layer of both the encoder and decoder. As seen in Figure 11, there is a clear separation between languages of the form En-X and X-En. So, we cluster the En-X and X-En language pairs (for the top layer of the decoder and encoder respectively) separately in Figure 11a and Figure 11b. The activations of the token embedding layers do not separate significantly, so we do not cluster them.

Low-resource, script-diverse language families

In this section we further the analysis from subsection 3.2, with a different set of language fam-



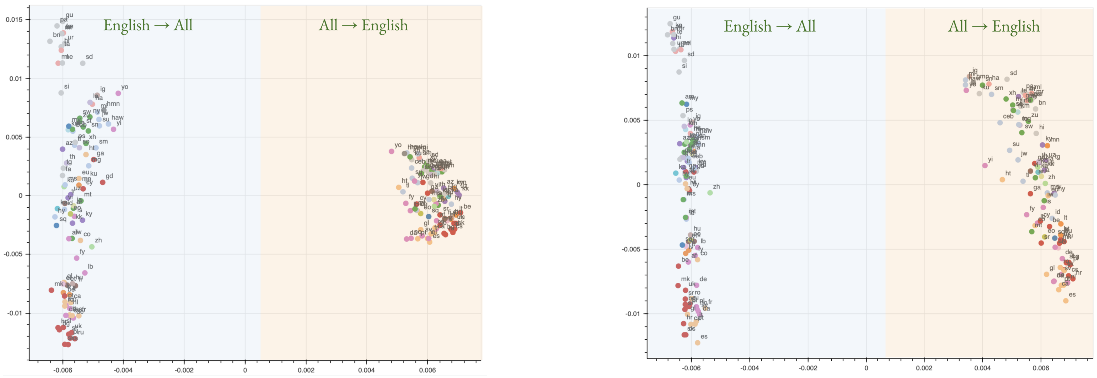
(a) Top layer of the encoder for X-En language pairs.



(b) Top layer of the decoder for En-X language pairs.

Figure 10: Visualization of the top layer of the encoder and decoder. Both the encoder and decoder show clustering according to linguistic similarity.

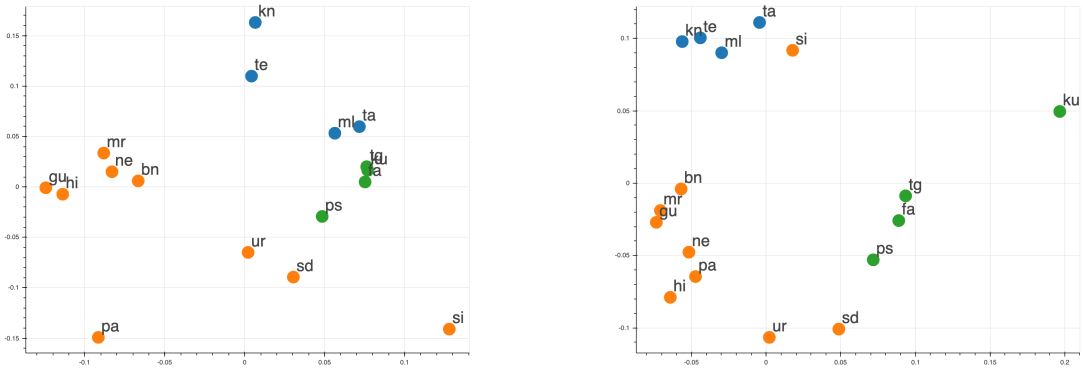
ilies. In Figure 12, we visualize the relationship between representations of the Iranian, Indo-Aryan, and Dravidian languages, and demonstrate that they cluster much more strongly by linguistic similarity than by script or dataset size. We furthermore demonstrate that within macro-clusters corresponding to languages of particular families, there exist micro-clusters corresponding to



(a) Top layer of the encoder.

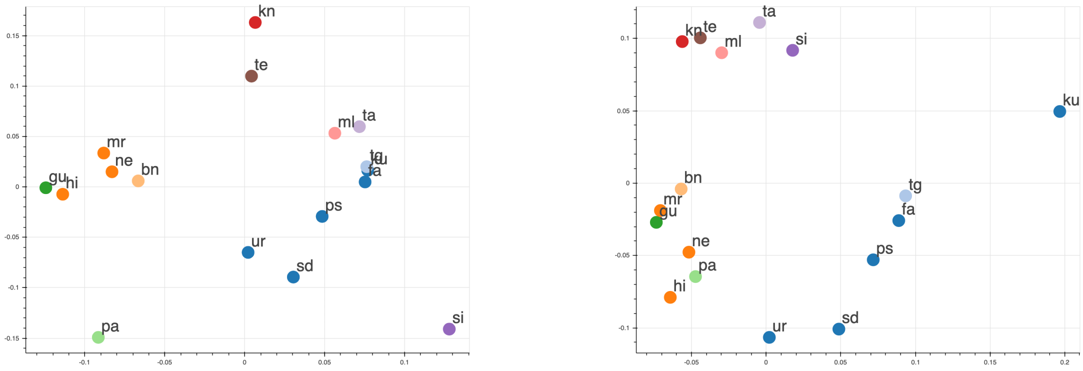
(b) Top layer of the decoder.

Figure 11: All-to-All (X-X) clustering of the encoder and decoder representations of all languages, based on their SVCCA similarity on our multi-way parallel evaluation set.



(a) Embeddings, colored by language group

(b) Encoder layer 5, colored by language group



(c) Embeddings, colored by script

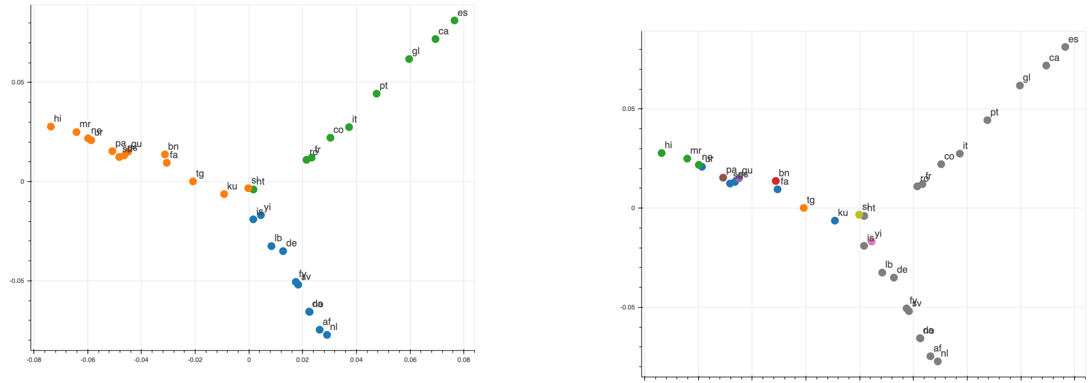
(d) Encoder layer 5, colored by script

Figure 12: Visualization of the Indo-Aryan languages, the Iranian languages, and the Dravidian languages, for the embeddings (left column) and the top layer of the encoder (right column), coloring by different attributes to highlight clusters. These are to-English direction.

branches within those families.

This is a diverse set of mid-to-low resource languages, using a variety of scripts. The Dravidian languages (Kannada, Malayalam, Tamil, and Telugu) are from South India, and each use their own abugida-based writing system. They are agglutinative. The Indo-Aryan languages comprise the North-Indian languages, which are fusional languages written in Devanagari (Hindi, Marathi, Nepali), Arabic script (Sindhi, Urdu), and several language-specific abugidas (Bengali, Gujarati, Punjabi). The Iranian languages include Farsi (Dari), Kurdish (in this case, Kurmanji), and Pashto, written in Arabic script; and Tajik, written in Cyrillic. All languages in these three groups use SOV word order, and lie along swath of land stretching roughly from Sri Lanka to Kurdistan. In our datasets they are all low-resource languages, though Hindi is on the upper end.

The first most striking thing about Figure 12 is



(a) Colored by language group: Indo-Iranian languages in red, Germanic languages in blue, and Romance languages in green. Note that within the Indo-Iranian languages, the Iranian languages are to the right, and the Indic languages to the left.

(b) Colored by script: Roman script in grey, Arabic script in blue, and a variety of other (mostly Indic) scripts.

Figure 13: Visualization of the embedding layer for three branches of the Indo-European language family, coloring by different attributes to highlight clusters. They group most strongly by linguistic group, with weak connection by script.

that **the linguistic group of each language appears to be a much stronger influence on the clustering tendency than the script, even at the level of the embeddings.** The Dravidian languages cluster nicely, even though none shares a subword with the other; as do the Indo-Aryan languages, which similarly are written in a variety of scripts, and the Iranian languages, which are written in two.

It is worth highlighting a few phenomena in this visualization. Firstly, the two dialects of Persian represented here, Tajik (tg) and Farsi (fa), are almost superimposed in the embedding visualization, though they are written in Cyrillic and Arabic scripts, respectively. Note also that in the embeddings, Hindi (hi) clusters closeliest with its fellow Western Indo-Aryan language Gujarati (gu), rather than the two other languages written in the Devanagari script, Marathi (mr) and Nepali (ne). Among the Dravidian languages, we see that Kannada (kn) and Telugu (te) form one pair, whereas Tamil (ta) and Malayalam (ml) form another pair, corresponding correctly with their linguistic similarity (Moorti, 2011), even though none of them share a writing system with any of the others.

Although the language family is important to these clusters, it is important to note the apparent role that writing system also plays in these visualizations. While Urdu (ur) and Sindhi (sd) weakly cluster with the Indo-Aryan languages, they are as close in the visualization to the less related Iranian languages (which also use the Arabic script) than their linguistic nearest neighbors, Hindi (hi) [10] and Punjabi (pa) [11] , respectively.

Mid to high-resource, same-scripted languages Unsurprisingly, in the higher-resource case, and when most languages use a comparable script, the clusters are much cleaner. Figure 13 shows an example with the Romance languages and the the Germanic languages, along with the Indo-Aryan languages for comparison. They form three distinct clusters along what appear to be latent directions encoding language family. The apparent intersection off the three contains a few low-resource languages, and can best be conceptualized as an artifact of the visualization.

A.5 Nearest neighbors based on representation similarity

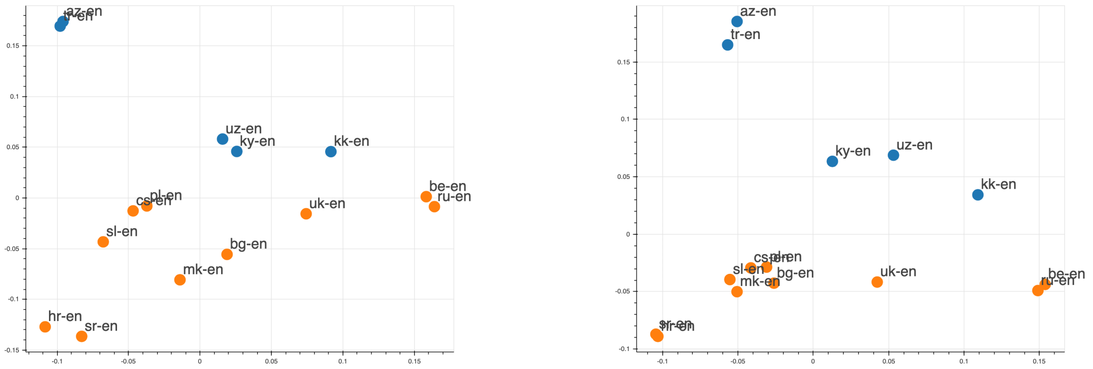
In this section we look at how the nearest neighbors to languages change from their representations in the embeddings to their representations in the top layer of the encoder. Table 1 displays a few representative results.

The nearest neighbors of languages in high-resource language families, like Romance languages and Germanic languages, tend to produce quite accurate nearest neighbor lists that are stable across layers in the encoder. The example of Spanish in Table 1 demonstrates this, producing

---

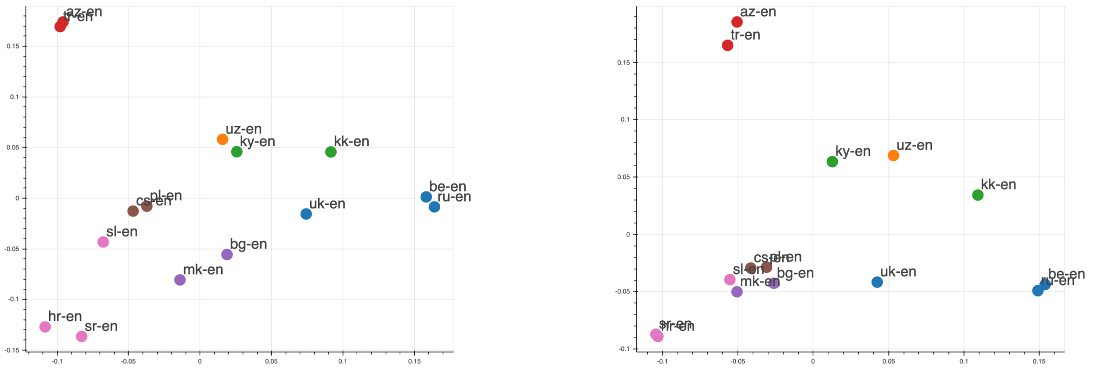
[10] Usually considered to be a register of the same language (Siddiqi, 1994; Hammarström et al., 2017)

[11] In that Sindhi and Punjabi are the two representatives of the Northwestern Indo-Aryan languages in this plot.



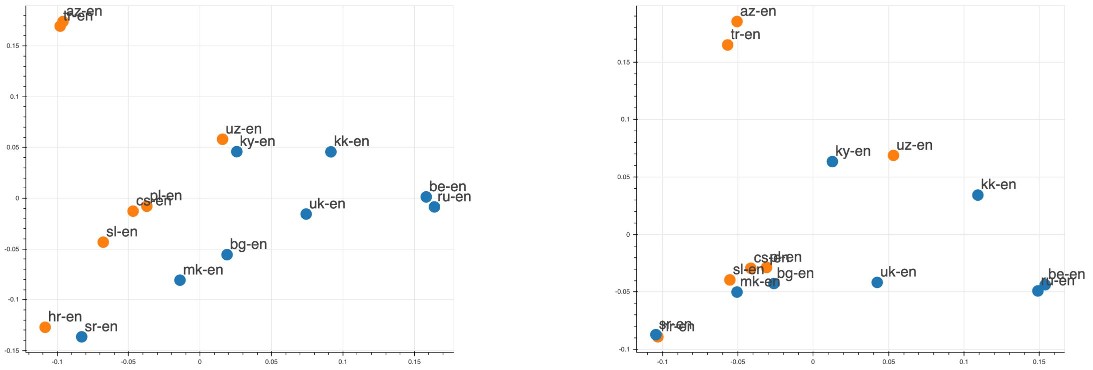
(a) Embeddings, colored by sub-family

(b) Encoder layer 5, colored by sub-family



(c) Embeddings, colored by branch with sub-family

(d) Encoder layer 5, colored by branch within sub-family



(e) Embeddings, colored by script (writing system)

(f) Encoder layer 5, colored by script (writing system)

Figure 14: Visualization of the Slavic and Turkic languages, for the embeddings (left column) and the top layer of the encoder (right column), coloring by different attributes to highlight clusters. These are to-English direction.

(at the encoder top) a remarkable list of the five linguistically closest languages to it in our dataset.

Lower-resource languages, however, tend to produce much noisier representations in the embeddings. The example of Yiddish is given in Table 1. The nearest neighbors in the embedding space are mostly nonsensical, with the exception of German. By the top of the encoder, however, the neighbors are really quite reasonable, and remarkable – given that Yiddish, a Germanic language (Herzog), is written in Hebrew script, whereas the remainder of the Germanic languages are written in Roman script. A similar example is Urdu, where the embeddings seem to be more influenced by less-related languages written in the same (Arabic) script, whereas by the top of the encoder, the neighbor list is a quite high-quality ranking of similar languages in entirely different scripts.

A last amusing example is Basque, a famous language isolate hiding out amidst the Indo-European languages in Europe. As expected from its status as an isolate, the nearest neighbors in the embedding space are a nonsensical mix of languages. However, by the top of the encoder the top four nearest neighbors are those languages geographically closest to Basque country (excepting French), probably reflecting lexical borrowing or areal influences on Basque.

Language	Rank	embeddings	encoder top
Yiddish	1	*Lao*	*German* ⋆⋆
	2	*German* ⋆⋆	*Norwegian* ⋆
	3	*Thai*	*Danish* ⋆
	4	*Hmong*	*Portuguese*
	5	*Korean*	*Macedonian*
Urdu	1	*Punjabi* ⋆	*Hindi* ⋆⋆
	2	Sindhi ⋆	*Punjabi* ⋆
	3	Pashto	*Bengali* ⋆
	4	*Hindi* ⋆⋆	*Gujarati* ⋆
	5	*Gujarati* ⋆	*Marathi* ⋆
Basque	1	Indonesian	Portuguese
	2	Javanese	Spanish
	3	Portuguese	Galician
	4	Frisian	Italian
	5	Norwegian	*Bosnian*
Spanish	1	Catalan ⋆	Catalan ⋆
	2	Galician ⋆	Portuguese ⋆
	3	Portuguese ⋆	Galician ⋆
	4	Italian ⋆	Italian ⋆
	5	Romanian ⋆	French ⋆

Table 1: Nearest Neighbors in SVCCA space for a given source language. Languages marked with a ⋆ are closely related; languages marked with ⋆⋆ are the closest languages in our dataset. Italicized languages are written in a different script. We see that the nearest neighbors are more meaningful in the top of the encoder than in the embeddings, and that the embeddings are more influenced by script.

In this example, the clusters remain about the same throughout the encoder, with the linguistic clusters becoming if anything a little tighter by the top layer of the encoder (Figure 12b, 12d). Sindhi and Urdu remain between the Indo-Aryan and Iranian languages. The one notable difference is that, whereas Sinhala (si) clusters with the Indo-Aryan languages in the embedding layer, it is firmly in the Dravidian cluster in the top of the encoder, with its nearest neighbor being Tamil. This may reflects the status of Sinhala as an Indo-Aryan language which has been lexically and grammatically influenced by sharing the island of Sri Lanka with Tamil over a thousand of years, to the extent that some earlier scholars erroneously believed the language to be Dravidian (Coperahawa, 2007). Alternately it could reflect similar subject matter of text, related to e.g. local politics – further analysis is required.

A.6 Finetuning Experiments

In Table 2 we list the language pairs with which we separately finetune our models and their resource sizes.

Resource Size	Languages
Low ($10^5 - 10^7$ sentences)	mr-en, km-en, uz-en, so-en, ky-en, ny-en, yo-en, ha-en, gd-en, ig-en
High ($10^8 - 10^9$ sentences)	es-en, tr-en,pl-en, ko-en, ru-en, sr-en, uk-en, ca-en

Table 2: Language pairs we finetune our model on. For the purpose of our analysis, low resource languages are language pairs whose training set contained $10^5 - 10^7$ parallel sentences and high resource languages are languages pairs whose training set contained $10^8 - 10^9$ parallel sentences.

Resource Size of Fine-tuning Language Pairs

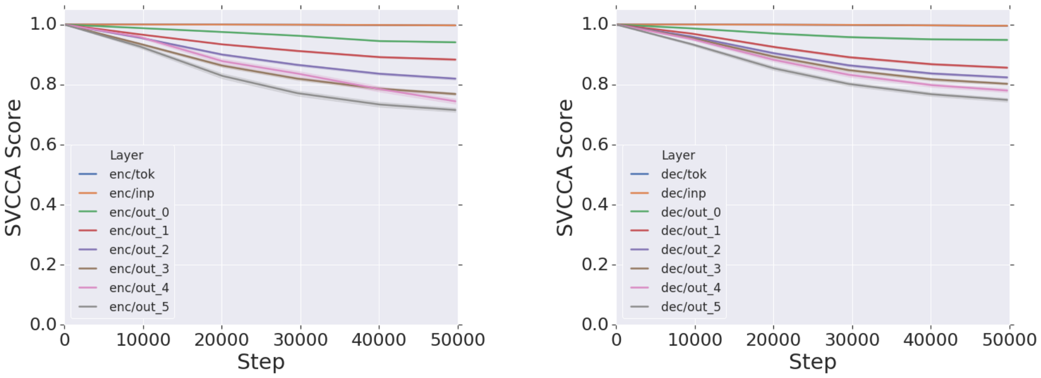
Sensitivity to Fine-tuning Increases Across Layers

In this subsection we plot the extent to which the representation space changes on average across language pairs (ie, decrease in SVCCA score) for different layers in Figure 15 on finetuning with these language pairs: ru-en (Russian), ko-en (Korean), uk-en (Ukrainian), km-en (Khmer). We see that the latter layers change the most across both the encoder and decoder.

Language	Id	Language	Id	Language	Id	Language	Id
Afrikaans	af	Galician	gl	Latvian	lv	Sindhi	sd
Albanian	sq	Georgian	ka	Lithuanian	lt	Sinhalese	si
Amharic	am	German	de	Luxembouish	lb	Slovak	sk
Arabic	ar	Greek	el	Macedonian	mk	Slovenian	sl
Armenian	hy	Gujarati	gu	Malagasy	mg	Somali	so
Azerbaijani	az	Haitian Creole	ht	Malay	ms	Spanish	es
Basque	eu	Hausa	ha	Malayalam	ml	Sundanese	su
Belarusian	be	Hawaiian	haw	Maltese	mt	Swahili	sw
Bengali	bn	Hebrew	iw	Maori	mi	Swedish	sv
Bosnian	bs	Hindi	hi	Marathi	mr	Tajik	tg
Bulgarian	bg	Hmong	hmn	Mongolian	mn	Tamil	ta
Burmese	my	Hungarian	hu	Nepali	ne	Telugu	te
Catalan	ca	Icelandic	is	Norwegian	no	Thai	th
Cebuano	ceb	Igbo	ig	Nyanja	ny	Turkish	tr
Chinese	zh	Indonesian	id	Pashto	ps	Ukrainian	uk
Corsican	co	Irish	ga	Persian	fa	Urdu	ur
Croatian	hr	Italian	it	Polish	pl	Uzbek	uz
Czech	cs	Japanese	ja	Portuguese	pt	Vietnamese	vi
Danish	da	Javanese	jw	Punjabi	pa	Welsh	cy
Dutch	nl	Kannada	kn	Romanian	ro	Xhosa	xh
Esperanto	eo	Kazakh	kk	Russian	ru	Yiddish	yi
Estonian	et	Khmer	km	Samoan	sm	Yoruba	yo
Filipino/Tagalog	tl	Korean	ko	Scots Gaelic	gd	Zulu	zu
Finnish	fi	Kurdish	ku	Serbian	sr		
French	fr	Kyrgyz	ky	Sesotho	st		
Frisian	fy	Lao	lo	Shona	sn		

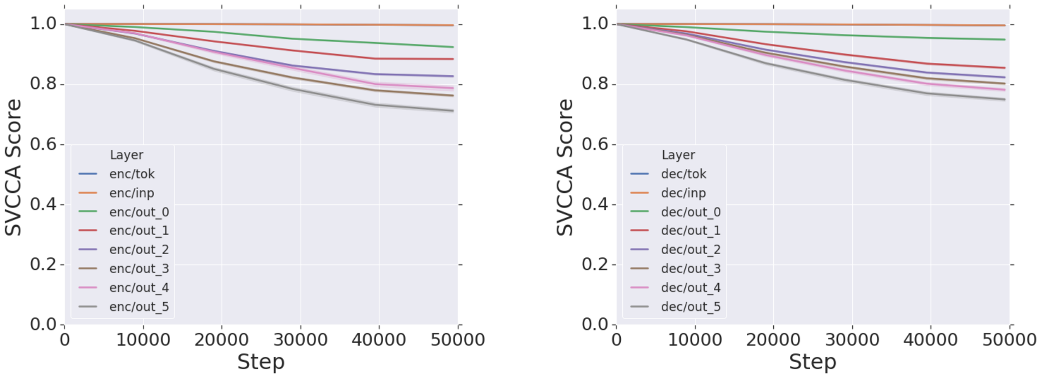
Table 3: List of BCP-47 language codes used throughout this paper (Phillips and Davis, 2009).

.



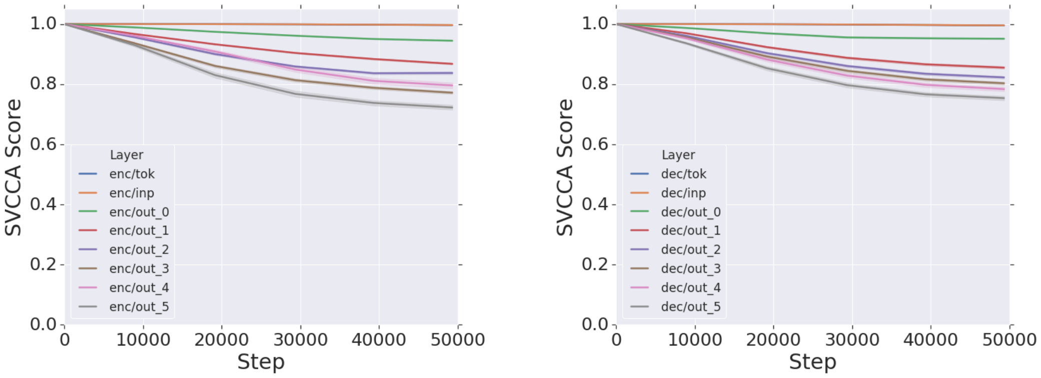
(a) Change in encoder on finetuning with ru-en.

(b) Change in decoder on finetuning with ru-en.



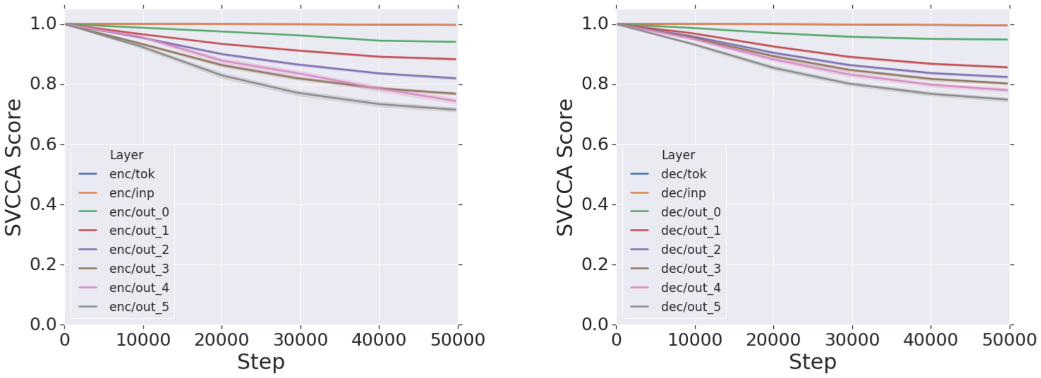
(c) Change in encoder on finetuning with ko-en.

(d) Change in decoder on finetuning with ko-en.



(e) Change in encoder on finetuning with uk-en.

(f) Change in decoder on finetuning with uk-en.



(g) Change in encoder on finetuning with km-en.

(h) Change in decoder on finetuning with km-en.

Figure 15: Comparing average change in representation space over finetuning steps across layers for various language pairs.