BERTologiCoMix*
How does Code-Mixing interact with Multilingual BERT?

Sebastin Santy† **Anirudh Srinivasan**† **Monojit Choudhury**

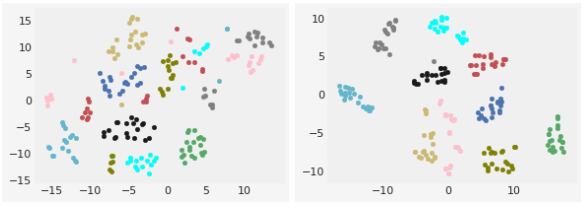
Microsoft Research, India
{t-sesan, t-ansrin, monojitc}@microsoft.com

Abstract

Models such as mBERT and XLMR have shown success in solving Code-Mixed NLP tasks even though they were not exposed to such text during pretraining. Code-Mixed NLP models have relied on using synthetically generated data along with naturally occurring data to improve their performance. Finetuning[1] mBERT on such data improves it's code-mixed performance, but the benefits of using the different types of Code-Mixed data aren't clear. In this paper, we study the impact of finetuning with different types of code-mixed data and outline the changes that occur to the model during such finetuning. Our findings suggest that using naturally occurring code-mixed data brings in the best performance improvement after finetuning and that finetuning with any type of code-mixed text improves the responsivity of it's attention heads to code-mixed text inputs.

1 Introduction

Massive multilingual models such as mBERT (Devlin et al., 2019) and XLMR (Conneau et al., 2020) have recently become very popular as they cover over 100 languages and are capable of zero-shot transfer of performance in downstream tasks across languages. As these models serve as good multilingual representations of sentences (Pires et al., 2019), there have been attempts at using these representations for encoding code-mixed sentences (Srinivasan, 2020; Aguilar et al., 2020; Khanuja et al., 2020). **Code-Mixing (CM)** is the mixing of words belonging two or more languages within a single sentence and is a commonly observed phenomenon in societies with multiple spoken languages. These multilingual models have shown promise for solving CM tasks having surpassed the previously achieved performances (Khanuja et al., 2020; Aguilar et al., 2020). This is an impressive feat considering that these models have never been exposed to any form of code-mixing during their pre-training stage.



(a) Vanilla mBERT (without fine-tuning) (b) mBERT fine-tuned on En-Hi CM

Figure 1: t-SNE representations of En-Es CM sentences on the respective models. Each color represents CM sentences with the same meaning but with different amounts of mixing generated based on the g-CM method in Sec 3.1. The tight clusters in (b) shows that CM sentence representations align better in mBERT fine-tuned on any CM data regardless of the language of mixing.

Traditionally, CM has been a spoken phenomenon though it is slowly penetrating into written form of communication (Tay, 1989). However, they mostly occur in an informal setting and hence such CM data is not publicly available in large quantities. Such scarcity of data would mean that building independent CM models can be unfeasible. With the onset of pre-trained multilingual models, further training with CM data can help in adapting these models for CM processing. However, even for further training, there is a requirement for a significant amount of data albeit lesser than starting from scratch. The amount of data available even for their monolingual counterparts is very less (Joshi et al., 2020) let alone the amount of real-world CM data. This can prove to be a bottleneck. Rightly so,

* The word BERTologiCoMix is a portmanteau of BERTology and Code-Mixing, and is inspired from the title of the graphic novel: *Logicomix: An Epic Search for Truth* by Apostolos Doxiadis and Christos Papadimitriou (2009).

† The authors contributed equally to the work.

[1] In this paper, unless specifically stated, finetuning refers to MLM finetuning/continued pretraining and not downstream task finetuning

there have been previous works exploring synthesis of CM data for the purpose of data augmentation (Bhat et al., 2016; Pratapa et al., 2018a). Synthesis of CM mostly rely on certain linguistic theories (Poplack, 2000) to construct grammatically plausible sentences. These works have shown that using the synthetic and real CM data in a curriculum setting while fine-tuning can help with achieving better performances on the downstream CM tasks. Though this is analogous to adapting models to new domains, CM differs in that the adaptation is not purely at vocabulary or style level but rather at a grammatical level. Although it is known such adaptation techniques can bring an improvement, it is not well understood how exactly fine-tuning helps in the CM domain.

Through this paper, we seek to answer these lingering questions which exist in area of CM processing. We first study the impact of finetuning multilingual models with different forms of CM data on downstream task performance. For this purpose, we rely on three forms of CM varying in their complexity of mixing, naturalness and obtainability - (i) randomly ordered code-mixing (l-CM), (ii) grammatically appropriate code-mixing (g-CM) both of which are synthetically generated and (iii) real-world code-mixing (r-CM). We perform this comparative analysis in a controlled setting where we finetune models with the same quantity of CM text belonging to different forms and then evaluate these finetuned models on 11 downstream tasks. We find that on average the r-CM performs better on all tasks, whereas the synthetic forms of CM (l-CM, g-CM) tend to diminish the performance as compared to the stock/non-finetuned models. However, these synthetic forms of data can be used in conjuction to r-CM in a curriculum setting which allows to alleviate the data scarcity issue. In order to understand the difference in the behavior of these models, we analyze their self-attention heads using a novel visualization technique and show how finetuning with CM causes the model to respond more effectively to CM texts. We notice that using r-CM for finetuning makes the model more robust and the representations more distributed leading to better and stable overall performances on the downstream tasks.

The rest of the paper is organized as follows. Section 2 surveys prior work done in domain adaptation of transformer-based LMs, code-mixing and interpretability and analysis techniques. Section 3 introduces the different types of code-mixing and the models that we build with them. Section 4 and 5 respectively presents the task-based and attention-head based probing experiments along with the findings. Section 6 concludes the paper by summarizing the work and laying out future directions.

2 Related Work

2.1 Domain Adaptation of BERT

Pre-trained Language Models trained on generic data such as BERT and RoBERTa are often adapted to the domain where it is required to be used. Domain adaptation benefits BERT in two ways (i) it gives exposure to text in the domain specific contexts and (ii) adds domain specific terms to the vocabulary. BERT has been adapted to several domains especially once which have its own complex jargon of communication such as the biomedical domain (Lee et al., 2020; Peng et al., 2019; Alsentzer et al., 2019), scientific texts or publications (Beltagy et al., 2019), legal domain (Chalkidis et al., 2020) and financial document processing (Yang et al., 2020b). Most of these works employ sophisticated techniques for mining large quantities of domain specific text from the internet and thus prefer to train the BERT model from scratch rather than fine-tuning the available BERT checkpoints. This is because they don't have to accommodate existing vocabulary along with the domain specific vocabulary which can lead to further fragmentation (Gu et al., 2020). While most works have looked at domain adaptation by plainly continuing the training using MLM objectives, some works have explored on different techniques to improve downstream task performance. Ma et al. (2019) uses curriculum learning and domain-discriminative data selection for domain adaptation. Adversarial techniques have been used for enforce domain-invariant learning and thus improve on generalization (Naik and Rose, 2020; Wang et al., 2019; Zhang et al., 2020). Ye et al. (2020) explores adapting BERT across languages. However, domain adaptation is not always effective and can lead to worse performances. This depends on several factors such as how different the domains are (Kashyap et al., 2020) or how much data is available (Zhang et al., 2020).

2.2 Code-Mixing

Traditionally, Code-Mixing has been used in informal contexts and can be difficult to obtain in large quantities (Rijhwani et al., 2017). This scarcity of data has been previously tackled by generation of synthetic CM data to augment the real CM data. Bhat et al. (2016); Pratapa et al. (2018a) demonstrate a technique to generate code-mixed sentences using parallel sentences and show that using these synthetic sentences can improve language model perplexity. A similar method is also proposed by Samanta et al. (2019) which uses parse trees to generate synthetic sentences. Yang et al. (2020a) generates CM sentences by using phrase tables to align and mix parts of a parallel sentence. Winata et al. (2019) proposes a technique to generate code-mixed sentences using pointer generator networks. The efficacy of synthetic CM data is evident from these works where they have been used in a curriculum setting for CM language modelling (Pratapa et al., 2018a), cross-lingual training of multilingual transformer models (Yang et al., 2020a) as well as to develop CM embeddings as a better alternative to standard cross-lingual embeddings for CM tasks (Pratapa et al., 2018b). In this work, we use grammatical theories to generate synthetic CM data from parallel sentences analogous to the aforementioned techniques.

2.3 BERT Attention based probing

Given the complex black-box nature of the BERT model, there have been a large number of works that propose experiments to probe and understand the working of different components of the BERT model. A large portion of these methods have focused on the attention mechanism of the transformer model. Clark et al. (2019); Htut et al. (2019) find that certain attention heads encode linguistic dependencies between words of the sentence. Kovaleva et al. (2019) report on the patterns in the attention heads of BERT and find that a large number of heads just attend to the `[CLS]` or `[SEP]` tokens and do not encode any relation between the words of the sentence. Michel et al. (2019); Prasanna et al. (2020) also show that many of BERT's attention heads are redundant and pruning heads does not affect downstream task performance. In this paper, we borrow ideas from these works and propose a technique for visualizing the attention heads and how their behaviour changes during finetuning.

3 Models

In this section, we describe the mBERT models, the modifications we make to them, and the types of CM data that we use for training.

3.1 Types of Code-Mixing

For the purpose of this study, we characterize CM data across two dimensions: *linguistic complexity* and *languages involved*. Here, we experiment with CM for two different language pairs: English-Spanish ($enes$) and English-Hindi ($enhi$). While Spanish has similar word order and a sizeable shared vocabulary with English, Hindi has a different word order and no shared vocabulary by virtue of using a different script. Thus, investigating through these two diverse pairs is expected to help us understand the representational variance.

The linguistic complexity of code-mixing can be categorized into the following three *types*:

Lexical Code-Mixing (l-CM): The simplest form of code-mixing is to substitute lexical units within a monolingual sentence with its counterpart from the other language. This can be achieved by using parallel sentences, and aligning the words with an aligner (Dyer et al., 2013).

Grammatical Code-Mixing (g-CM): There are grammatical constraints (Joshi, 1982; Poplack, 2000; Belazi et al., 1994) on word-order changes and lexical substitution during code-mixing that the l-CM does not take into account. Pratapa et al. (2018a) propose a technique to generate all grammatically valid CM sentences from a pair of parallel sentences. Here, we use this generated dataset as our g-CM[2].

Parse trees are generated for parallel sentences (between two languages L_1 and L_2), and common nodes between these parse trees are then replaced based on certain conditions specified by Equivalence Constraint (EC) theory (Poplack, 2000; Sankoff, 1998), thereby producing a grammatically sound code-mixing. Fine-tuning with this form of CM should ideally impart the knowledge of grammatical boundaries for CM and would let us know whether a grammatically correct CM sentence is required to improve the performance.

Real Code-Mixing (r-CM): While g-CM considers purely the syntactic structure of CM, real-world

[2]It is important to note that Pratapa et al. (2018a) uses GCM to denote "Generated CM" data, and not for "grammatical" as is used here.

	SENT		NER		POS			LID		QA	NLI
model	*enes*	*enhi*	*enes*	*ehi*	*enes*	*enhi*	*enhi*	*enes*	*enhi*	*enhi*	*enhi*
$m_{\langle\rangle}$	67.81$_{\pm 2.5}$	**58.42**$_{\pm 1.1}$	59.50$_{\pm 0.9}$	75.55$_{\pm 0.6}$	93.35$_{\pm 0.2}$	87.49$_{\pm 0.1}$	63.40$_{\pm 0.5}$	95.99$_{\pm 0.0}$	**95.80**$_{\pm 0.4}$	71.95$_{\pm 0.8}$	**63.25**$_{\pm 1.9}$
$m_{\langle l,\odot\rangle}$	68.07$_{\pm 1.5}$	58.08$_{\pm 0.8}$	59.39$_{\pm 1.0}$	76.53$_{\pm 1.0}$	**93.84**$_{\pm 0.1}$	88.00$_{\pm 0.2}$	**64.09**$_{\pm 0.2}$	96.09$_{\pm 0.1}$	95.32$_{\pm 0.9}$	70.53$_{\pm 3.5}$	62.94$_{\pm 2.7}$
$m_{\langle g,\odot\rangle}$	68.64$_{\pm 1.5}$	57.90$_{\pm 1.1}$	59.88$_{\pm 0.7}$	76.86$_{\pm 0.6}$	93.74$_{\pm 0.1}$	87.79$_{\pm 0.2}$	63.79$_{\pm 0.2}$	96.06$_{\pm 0.0}$	95.41$_{\pm 0.8}$	70.11$_{\pm 1.8}$	55.19$_{\pm 6.5}$
$m_{\langle r,\odot\rangle}$	**68.51**$_{\pm 0.7}$	58.25$_{\pm 0.8}$	**60.46**$_{\pm 0.6}$	**76.86**$_{\pm 0.5}$	93.68$_{\pm 0.1}$	**88.00**$_{\pm 0.0}$	63.38$_{\pm 0.0}$	**96.12**$_{\pm 0.0}$	94.60$_{\pm 0.2}$	**73.54**$_{\pm 3.9}$	60.00$_{\pm 5.7}$

Table 1: Performance of the models for different tasks along with their standard deviations. The trained model language corresponds to the language the model is tested on, and is denoted by $\odot$. ***r***-CM trained models almost always perform better than models trained on other types of CM data.

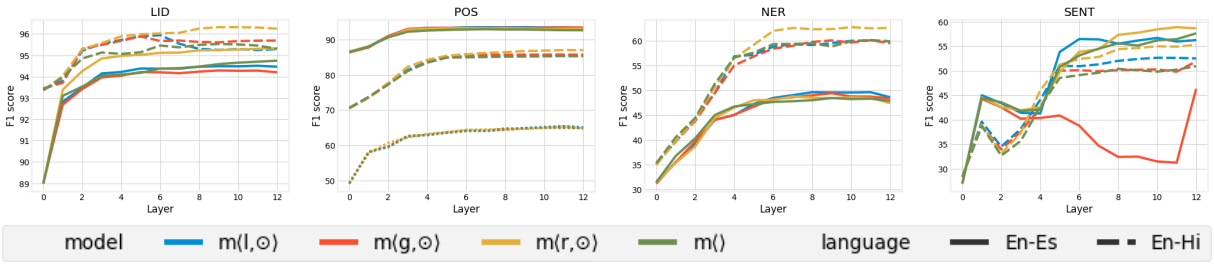


Figure 2: Layer-wise F1 scores for LID, POS, NER and SENT respectively across different layers. The dashed lines represent the *enhi* versions and solid lines represent the *enes* versions of different tasks.

code-mixing is influenced by many more factors such as cultural/social and/or language-specific norms which comes in the semantic and pragmatics space of language understanding. Though ***r***-CM is a subset of ***g***-CM, there does not exist any method which can sample realistic CM from such synthetic data, hence we rely on real-world CM datasets. Fine-tuning with this form should let the model become aware of certain nuances of real-world code-mixing which are still not completely known.

3.2 Training Procedure

There are 3 [types] $\times$ 2 [language-pairs] = 6 combinations of data which can be obtained based on the previous specifications. For ***l***-CM and ***g***-CM, we use the same set of parallel sentences: en-es from Rijhwani et al. (2017) and en-hi from Kunchukuttan et al. (2018). As CM is prominently used in informal contexts, it is difficult to procure textual ***r***-CM data. We use twitter data from Rijhwani et al. (2017) for en-es; for en-hi, we use data from online forums and Twitter respectively from Chandu et al. (2018) and Patro et al. (2017). For each of the 6 combinations, we randomly sample 100,000 sentences which is then used to further train mBERT with the masked language modelling objective. We use layer-wise scaled learning rate while finetuning the models. Sun et al. (2019)

Model Notation: Let $m_{\langle\rangle}$ be the vanilla mBERT, then $m_{\langle p,q\rangle}$ are the mBERTs further trained on $\langle p, q\rangle$ data, where $p \in \{l, g, r\}$ is the complexity of mixing and $q \in \{enes, enhi\}$ is the language of mixing. For example, a model trained on English-Hindi lexical code-mixed data will be represented as $m_{\langle l,enhi\rangle}$. $\odot$ means that the model used depends on the configuration of the corresponding data. For example, $m_{\langle l,\odot\rangle}$ with *enes* data would mean that the model used is $m_{\langle l,enes\rangle}$ while with *enhi* data would mean that the model used is $m_{\langle l,enhi\rangle}$.

4 Task-based Probing

In this section, we describe layer-wise task-based probing of the different models.

4.1 Tasks

Recently, two benchmarks for code-mixing were released: GLUECoS (Khanuja et al., 2020) and LINCE (Aguilar et al., 2020). For this study, we probe with the following tasks from GLUECoS: Language Identification (LID), Part-of-Speech (POS) Tagging, Named Entity Recognition (NER) and Sentiment Analysis (SENT) for both *enes* and *enhi*, and Question Answering (QA) and Natural Language Inference (NLI) for only *enhi*.

4.2 Method

We first measure the performance of these models on the aforementioned tasks. For each task, we fine-tune the models further after attaching a task specific classification layer. We report the average performances and standard deviations of each model run for 5 seeds in Table 1. [3]

In addition to getting absolute performances, we want to get an insight of how much each layer of the different models contribute to the performance of a particular task. Following Tenney et al. (2019), we measure the *solvability* of a task by finding out the *expected layer* at which the model is able to correctly solve the task. Here the mBERT weights are kept frozen and a weighted sum of representations from each layer are passed to the task specific layer. Figure 2 shows the layer-wise F1 scores for the tasks for different models and language pairs. We additionally calculate scalar mixing weights which lets us know the contribution of each layer by calculating the attention paid to each layer for the task.

4.3 Observations

From Table 1 it is clear that for almost all the tasks, $m_{\langle r,\odot\rangle}$ models perform better than the other fine-tuned models. In Particular, fine-tuning with $\boldsymbol{r}$-CM data helps with SENT, NER, LID $enes$ as well as QA tasks. While for POS, the performance remains almost same regardless of which data the model is fine-tuned with. [4]

These differences are also reflected in the layer-wise performance of these models as shown in Figure 2. The tasks are considered solved at the knee point where the performances start plateauing. The performances of different models start at the same note, and after a certain point $m_{\langle r,\odot\rangle}$ diverges to plateau at a higher performance than others. This can be attributed to final layers adapting the most during MLM fine-tuning. (Liu et al., 2019; Kovaleva et al., 2019). LID gets solved around 2$^{\text{nd}}$ layer. $enhi$ LID gives a relatively high performance at the 0$^{\text{th}}$ layer indicating that it only needs the token+positional embeddings. This is because $enhi$ LID task has en and hi words in different scripts, which means it can be solved even with a simple unicode classification rule. POS gets solved at around 4$^{\text{th}}$ layer. The indifference to fine-tuning observed in case of POS is reflected here as well, as all the models are performing equally at all the layers for both the languages.

NER gets solved around the 5$^{\text{th}}$ layer. Here, $\boldsymbol{r}$-CM training seems to help for $enhi$, perhaps due to exposure to more world knowledge which is required for NER. SENT shows an interesting shift in patterns. We can see that $m_{\langle l,enes\rangle}$ solves the task at 6$^{\text{th}}$ layer whereas the other models solve it at around 8$^{\text{th}}$ layer. Thus, the general trend observed is that easier tasks like LID, POS are solved in the earlier layers and as the complexity of the tasks increase, the effective layer moves deeper - which shows a neat pattern of how BERT "re-discovers" the NLP pipeline (Tenney et al., 2019), or rather the CM pipeline in our case.

5 Structural Probing

As observed earlier, exposing mBERT to $\boldsymbol{r}$-CM help boost its overall and layer-wise performance on CM tasks. In this section, we describe three structural probing experiments, through which we will try to visualize the structural changes in the network, if any, induced by continued pre-training with CM data that are responsible for performance gains. We will first look at whether there are any changes in the behaviour of attention heads at a global level by checking the inter-head distances within a model. Further, we want to localize and identify the heads whose behaviours have changed. Finally, we take a look at how the attention heads respond to code-mixed stimulus.

5.1 Probes

The probes for conducting the experiments consist of CM and Monolingual sentences. We take a sample of 1000 sentences for each type of CM as well as monolingual sentences for each language.

Probe Notation: To denote these probes, we use $d_{\langle p,q\rangle}$ such that $p \in \{l, g, r\}$ is the complexity of mixing and $q \in \{enes, enhi, en, hi, es\}$ are the languages. For example, English-Spanish lexical CM data is represented as $d_{\langle l,enhi\rangle}$ and (real) Spanish monolingual data is represented as $d_{\langle -,es\rangle}$.

[3] As we use just 100k sentences as opposed to 3M sentences, we do not get the same performance jump reported by Khanuja et al. (2020).

[4] We also carried out training in a curriculum fashion where synthetic CM data was first introduced followed by real CM data in different ratios similar to Pratapa et al. (2018a). However, we do not include these numbers as we could not derive any meaningful insights from them. This can most probably be due to a fixed constraint of 100k sentences that we use.

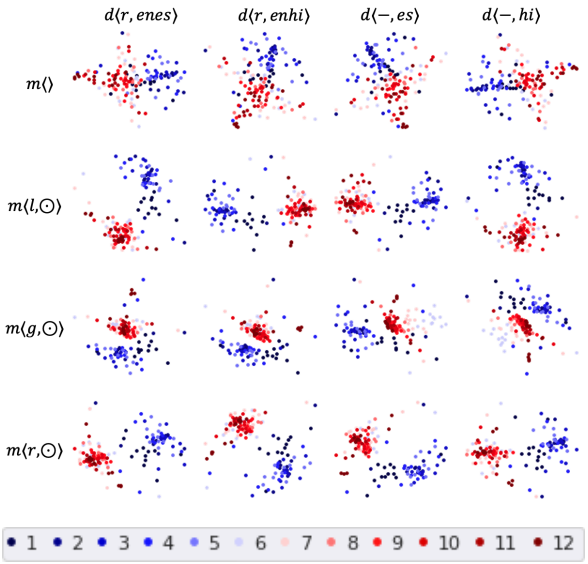


Figure 3: Intra-head distances $d(\mathbf{H}_i, \mathbf{H}_j)$ for models. The points are colored layer-wise and follows dark blue $\rightarrow$ light blue/red $\rightarrow$ dark red scheme. The rows are the models which are used and columns are the different set of probes used. $\odot$ indicates that model trained in the same language (code-mixed) as probe is used.

5.2 Global Patterns of Change

Has anything changed within the models due to pre-training with CM datasets? In order to answer this question, we look at the global patterns of relative distances between the attention heads within a model.

Method: Clark et al. (2019) describes an inter-head similarity measure which allows for visualizing distances between each attention head with another within a model. The distance d between two heads $\mathbf{H}_i$ and $\mathbf{H}_j$ is calculated as,

$$d(\mathbf{H}_i, \mathbf{H}_j) = \sum_{\text{token} \in \text{ sentence}} JS(\mathbf{H}_{i(\text{token})}, \mathbf{H}_{j(\text{token})}) \quad (1)$$

where JS is the Jensen-Shannon Divergence between attention distributions. We average these distances obtained across 1000 sentences ($d_{\langle\odot,\odot\rangle}$). Further, in order to visualize these head distances, we use multidimensional scaling (Kruskal, 1964) which preserves the relative distance better than other scaling methods such as T-SNE or PCA (Van Der Maaten et al., 2009).

Observation: Figure 3 shows the two-dimensional projections of the heads labeled by the layers. There are clear differences between the patterns in $m_{\langle\rangle}$ and the other models, though the same cannot be said for the probes. $m_{\langle\rangle}$ shows a rather distributed representation of heads across layers; in particular, $\boldsymbol{g}$-CM models have a tightly packed representation especially for the later layers.

5.3 Local Patterns of Change

Attention patterns of which heads have changed? We observe that there is a change in the overall internal representations based on the type of data which the models are exposed to. It would be interesting to know which specific attention heads, or layers are most affected by the exposure to CM data.

Method: In order to contrast the attention patterns of specific heads between $m_{\langle\odot,\odot\rangle}$ and the base model - $m_{\langle\rangle}$, we calculate the distance between their corresponding heads as follows:

$$\Delta_m = JS(\mathbf{H}^{m_{\langle\odot,\odot\rangle}}_{i,j\,(\text{token})}, \mathbf{H}^{m_{\langle\rangle}}_{i,j\,(\text{token})}) \quad (2)$$

where JS is the Jensen-Shannon Divergence, i and j are the layers and their respective heads, $m_{\langle\odot,\odot\rangle}$ is any model in the set of fine-tuned models and $m_{\langle\rangle}$ is the vanilla model. We visualize these distances in form of heatmaps (Δ_m maps). For the sake of clarity, only top 15 attention heads is plotted for each Δ_m map. The darker the head, the more the head has changed between a particular trained model and the vanilla model. Visual triangulation can let us understand if there are common heads between sets of models and probes.

Observation: Figure 4 depicts the different combinations of Δ_m maps. It can be seen how there are common heads between different configurations of trained models as well as the inference data which is used. Here, even the difference between different languages and forms of code-mixing stand out compared to the previous analysis. We also look at cross-interaction of languages: fine-tuned in one language and probed on another. Through visual examination, we highlight some of the common heads which are present among the different Δ_m plots.

5.4 Responsivity to Code-Mixing

How do attention heads respond to code-mixed probes? The common patterns in the way heads are functioning between different models and probes are easily observable from the set of Δ_m maps. These do point us to certain heads getting more activated while encoding a particular type of CM sentence. In this section, we want to understand

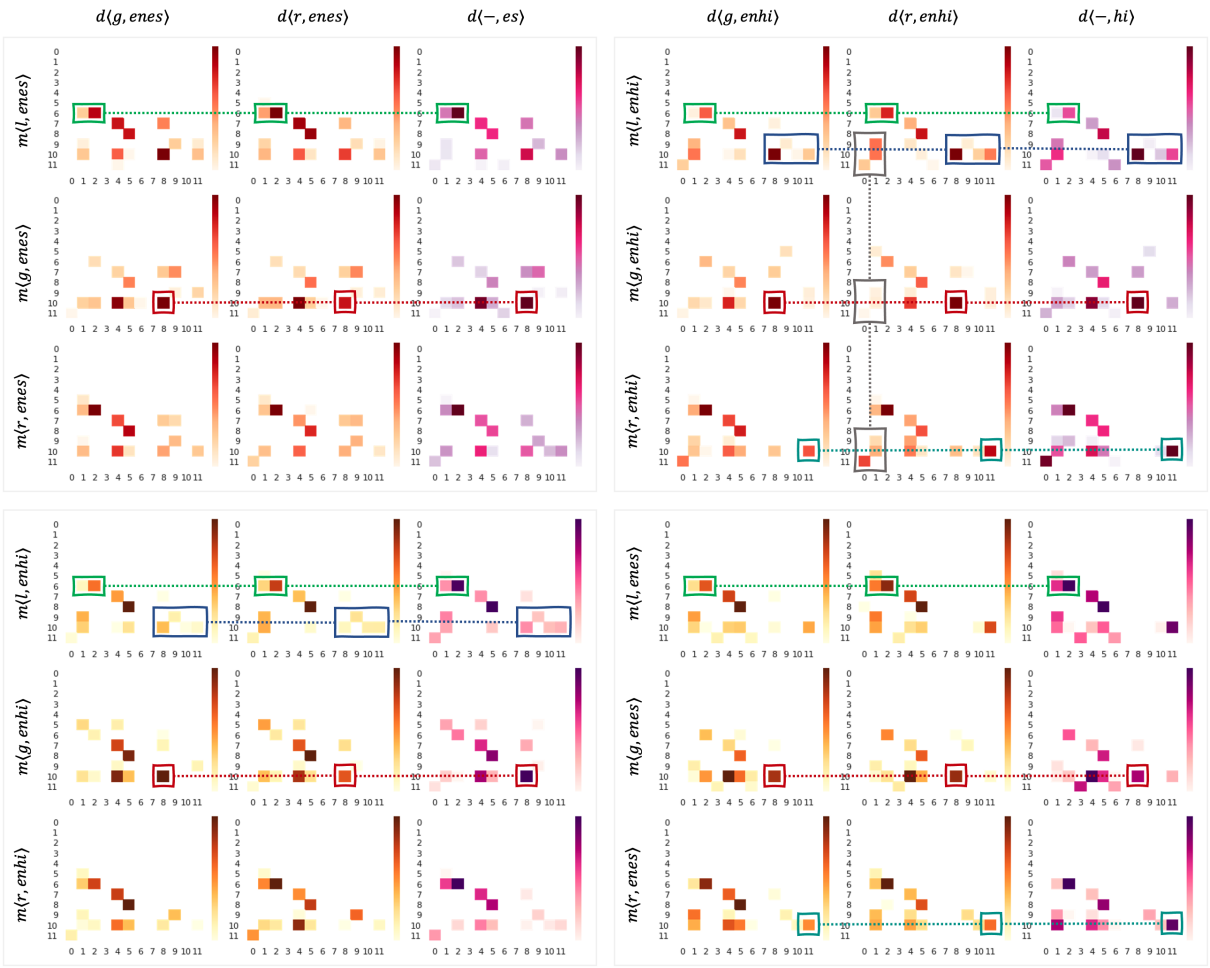


Figure 4: Δ_m maps for different configurations of trained models and probes. The first row of maps depict head interactions within same language whereas the second row of maps depict cross-language interaction i.e. trained and probed on different languages. Some of the common heads that can be observed have been marked to show the patterns which differentiate between complexity and language of models and probes.

how these heads respond to input probes. We borrow the term *responsivity*, $\mathcal{R}$, from the field of neuroscience which is used to summarize the change in the neural response per unit signal (stimulus) strength. In this context, we want to understand the change in attention head response of different models when exposed to CM data which act as the stimulus.

Method: Our aim is to understand the excitement of different heads when they see code-mixed data as a stimulus. To this end, we design a classification experiment to quantify the excitement of each head (/ neuron) while distinguishing between monolingual and CM classes. For the CM class, we randomly sample 2000 sentences from $\boldsymbol{r}$-CM in the same way as we did for probes. Similarly, for monolingual class, we sample 1000 sentences each from en and es or hi. Each probe is then passed through the different models to obtain the attentions. To summarize the net attention for each head, we average the attentions over all the tokens after removing `[CLS]` & `[SEP]` tokens present in that head. (`[CLS]` & `[SEP]`) tokens are removed as they act as a sink to non-attended tokens.

These average attention heads are then used as features (x) ($12 \times 12 = 144$ features) with the monolingual and CM classes being the predictor variable (y). To capture the relative excitement of different heads to y, we define responsivity ($\mathcal{R}$) as the gain of information of each feature (or heads) in context of the prediction variable (y). This is analogous to Information Gain used in determining feature importance. Hence, Responsivity of a head x for class y can be written as:

$$\mathcal{R}_{x,y} = H(x) - H(x|y) \tag{3}$$

where, $H(x)$ is the entropy of class distribution for x and $H(x|y)$ is the conditional entropy for x

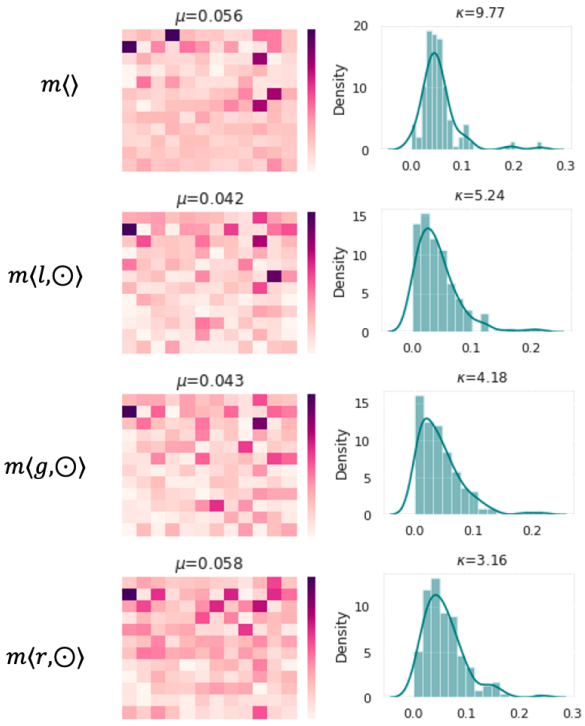


Figure 5: $\mathcal{R}$ of different models when classifying Monolingual vs. Code-mixing sentence

given y.

Observation: As shown in Figure 5, we plot the responsivity of different attention heads to CM in the form of 12×12 $\mathcal{R}$ heatmaps. We also plot the distribution of these values. We report two values, mean responsivity (μ) of a model to code-mixing and kurtosis (κ) to measure the skewness or the tailedness of the distribution compared to a normal distribution.

It can be observed from the heatmaps that there are certain common heads such as (1, 0), (2, 9) which are highly responsive to CM. As we pump in different types of CM data, we can observe that responsivity of some heads [(5, 10), (6,9)] are reducing while for other heads [(1, 7), (4, 8)] it is spiking up. A distinctive pattern that can be noticed from the heatmaps is that as CM data is fed to the models in the order of their linguistic complexity, more and more heads are responding towards the CM stimulus. Even the distribution density curve widens as confirmed by decreasing Kurtosis.

As described earlier, there is no single point in the network which responds to CM data. Previous studies (Elazar et al., 2020) involving probing of specific regions to understand their independent contributions to solving any task has been somewhat futile. It has been observed that heads collectively work towards solving tasks, and such specific regions cannot be demarcated - which means that information pertaining to task-solving is represented in a distributed fashion. In line with this, it has been shown that these models can be significantly pruned during inference with minimal drop in performance (Michel et al., 2019; Kovaleva et al., 2019). Our study confirms these observations for code-mixing as well, through a different visualization approach.

6 Conclusion & Future Work

In this work, we develop different methods of fine-tuning BERT-like models for CM processing. We then compare the downstream task performances of these models through absolute performance, their stability as well as the layer-wise *solvability* of certain tasks. To further understand the varied performances between the three types of CM, we perform structural probing. We adopted an existing approach and introduced a couple of new techniques for the visualization of the attention heads as a response to probes.

The most important finding from these probing experiments is that there are discernable changes introduced in the models due to exposure to CM data, of which a particularly interesting observation is that this exposure increases the overall *responsivity* of the attention heads to CM. As of now, these experiments are purely analytical in nature where we observed how the attention heads behave on a CM stimuli. One future direction is to expand the analysis to a wider range of domains and fine-tuning experiments to understand how generalizable are our findings of distributed information in BERT-like models. We use a fairly simple and easily replicable method for testing this through the *responsivity* metric that we propose. This method can be further improved to rigorously verify our observations.

7 Acknowledgements

We thank the anonymous reviewers for their many insightful comments and suggestions on our paper. We also thank Tanuja Ganu and Amit Deshpande for their valuable feedback on some of our experiments.

References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. LinCE: A centralized benchmark for linguistic code-switching evaluation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1803–1813, Marseille, France. European Language Resources Association.

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Hedi M Belazi, Edward J Rubin, and Almeida Jacqueline Toribio. 1994. Code switching and x-bar theory: The functional head constraint. *Linguistic inquiry*, pages 221–237.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Gayatri Bhat, Monojit Choudhury, and Kalika Bali. 2016. Grammatical constraints on intra-sentential code-switching: From theories to working models. *arXiv preprint arXiv:1612.04538*.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Khyathi Chandu, Thomas Manzini, Sumeet Singh, and Alan W. Black. 2018. Language informed modeling of code-switched text. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 92–97, Melbourne, Australia. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2020. When bert forgets how to pos: Amnesic probing of linguistic properties and mlm predictions. *arXiv preprint arXiv:2006.00995*.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. 2019. Do attention heads in bert track syntactic dependencies?

Aravind Joshi. 1982. Processing of sentences with intra-sentential code-switching. In *Coling 1982: Proceedings of the Ninth International Conference on Computational Linguistics*.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Abhinav Ramesh Kashyap, Devamanyu Hazarika, Min-Yen Kan, and Roger Zimmermann. 2020. Domain divergences: a survey and empirical analysis. *arXiv preprint arXiv:2010.12198*.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. GLUECoS: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585, Online. Association for Computational Linguistics.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Joseph B Kruskal. 1964. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Domain adaptation with BERT-based domain classification and data selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 76–83, Hong Kong, China. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Aakanksha Naik and Carolyn Rose. 2020. Towards open domain event trigger identification using adversarial domain adaptation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7618–7624, Online. Association for Computational Linguistics.

Jasabanta Patro, Bidisha Samanta, Saurabh Singh, Abhipsa Basu, Prithwish Mukherjee, Monojit Choudhury, and Animesh Mukherjee. 2017. All that is English may be Hindi: Enhancing language identification through automatic ranking of the likeliness of word borrowing in social media. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2264–2274, Copenhagen, Denmark. Association for Computational Linguistics.

Yifan Peng, Shankai Yan, and Zhiyong Lu. 2019. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 58–65, Florence, Italy. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Shana Poplack. 2000. Sometimes i'll start a sentence in spanish y termino en español: Toward a typology of code-switching. *The bilingualism reader*, 18(2):221–256.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT Plays the Lottery, All Tickets Are Winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3208–3229, Online. Association for Computational Linguistics.

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018a. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1543–1553, Melbourne, Australia. Association for Computational Linguistics.

Adithya Pratapa, Monojit Choudhury, and Sunayana Sitaram. 2018b. Word embeddings for code-mixed language processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3067–3072, Brussels, Belgium. Association for Computational Linguistics.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. 2017. Estimating code-switching on Twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1971–1982, Vancouver, Canada. Association for Computational Linguistics.

Bidisha Samanta, Niloy Ganguly, and Soumen Chakrabarti. 2019. Improved sentiment detection via label transfer from monolingual to synthetic code-switched text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3528–3537, Florence, Italy. Association for Computational Linguistics.

David Sankoff. 1998. The production of code-mixed discourse. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics,*

Volume 1, pages 8–21, Montreal, Quebec, Canada. Association for Computational Linguistics.

Anirudh Srinivasan. 2020. MSR India at SemEval-2020 task 9: Multilingual models can do code-mixing too. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 951–956, Barcelona (online). International Committee for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Mary WJ Tay. 1989. Code switching and code mixing as a communicative strategy in multilingual discourse. *World Englishes*, 8(3):407–417.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. 2009. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13.

Huazheng Wang, Zhe Gan, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, and Hongning Wang. 2019. Adversarial domain adaptation for machine reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2510–2520, Hong Kong, China. Association for Computational Linguistics.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2019. Code-switched language models using neural based synthetic data from parallel sentences. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 271–280, Hong Kong, China. Association for Computational Linguistics.

Jian Yang, Shuming Ma, Dongdong Zhang, ShuangZhi Wu, Zhoujun Li, and Ming Zhou. 2020a. Alternating language modeling for cross-lingual pre-training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9386–9393.

Yi Yang, Mark Christopher Siy UY, and Allen Huang. 2020b. Finbert: A pretrained language model for financial communications.

Hai Ye, Qingyu Tan, Ruidan He, Juntao Li, Hwee Tou Ng, and Lidong Bing. 2020. Feature adaptation of pre-trained language models across languages and domains with robust self-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7386–7399, Online. Association for Computational Linguistics.

Xinyu Zhang, Andrew Yates, and Jimmy Lin. 2020. A little bit is worse than none: Ranking with limited training data. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 107–112, Online. Association for Computational Linguistics.