A LSTM language model learns Hindi-Urdu case-agreement interactions, and has a linear encoding of case

Satoru Ozaki and Rajesh Bhatt and Brian Dillon
University of Massachusetts Amherst
{sozaki,bhatt,bwdillon}@umass.edu

Abstract

Much evaluation work in the literature shows that neural language models seem capable of capturing syntactic dependencies in natural languages, but they usually look at relatively simple syntactic phenomena. We show that a two-layer LSTM language model trained on 250M morphemes of Hindi data can capture the relatively complex interaction between case and agreement in Hindi-Urdu, at an accuracy of 81.17%. Furthermore, we show that this model encodes case-marking linearly, implementing a geometrically intuitive and interpretable syntactic processing mechanism. We also show that this model doesn't calculate agreement extremely eagerly, as case information seems to be persistent over time as a sentence unfolds. This is surprising given LSTMs autoregressive and recurrent nature, which should exert an incremental processing pressure onto our model.

1 Introduction

Neural language models trained for engineering purposes tend to show human-like behavior when evaluated on certain benchmarks constructed to test their understanding of syntactic properties of certain natural languages. These results are quite significant, because they show that neural networks capture syntactic dependencies that target latent hierarchical structures even when they are trained on an objective as simple as next-word prediction, which doesn't provide any explicit signal about hierarchical structure. However, these benchmarks often only target relatively simple grammatical phenomena, such as English subject-verb number agreement. Thus, we don't know if language models really learn the full range of complex phenomena featured in various natural languages. Another problem concerns interpretability: when these language models display human-like behavior, what kind of computation underlies their such performances? Understanding the expressibility and the computation implemented by language models is empirically important for assessing whether they are viable models of grammar and sentence processing. In this paper, we show a LSTM language model (Gulordava et al., 2018) trained on Hindi data predicts the correct agreement form of a participial verb correctly 81.17% of the time, and encodes ergative and accusative case in a subspace of its hidden layer vectors in a way that makes representations for sentences containing each of these case-markers linearly separable from those that don't contain each case-marker. Our results suggest that a LSTM language model is not only capable of learning the relatively complex interaction between case and agreement in Hindu-Urdu, but also encodes case-marking information in a geometrically intuitive and interpretable fashion. We think this work points to a direction for future work in which we can compare language models with different architectures in how they represent and compute with case.

This paper is organized as follows. In Section 2, we describe relevant work. We discuss two groups of methods: those for evaluating language models' ability to learn syntactic properties of natural languages, and those for understanding the representations and computations tacitly implemented by language models.

In Hindu-Urdu, verb agreement targets different arguments depending on their case-marking patterns, making it a relatively complex agreement pattern and a good testing ground for evaluating language models' ability to capture syntactic dependencies. We describe the Hindi-Urdu facts in more detail in Section 3, and the training and evaluation procedures as well as evaluation results in Section 5. Despite the modest model size and training setup, the language model performs reasonably well, predicting the correct gender agreement 81.17% of the time.

In the rest of the paper, we investigate the nature of the computation that underlies our language model's decent performance. This investigation is carried out from two perspectives, which we describe in Section 6. The first one concerns how the language model represents case. We set forth a very specific hypothesis, which is that the model provides a linear encoding of case. If this were true, the model implements a highly interpretable syntactic processing mechanism. The second perspective concerns the memory usage. The computation underlying our language model could be *eager* and *Markovian*, making use of the subject's case information as soon as it is processed, after which this piece of information no longer has any bearing on the predicted gender agreement. Alternatively, it could be *lazy* and *memory-intensive*, storing the subject's case information in its intermediate representations, using it just-in-time as the model predicts a gender agreement marker. In the latter case, subject case information is used long after it has processed the subject.

We carry out the investigation using linear classifier probes and causal intervention techniques. These methods, as well as our results, are described in Section 7. We find positive evidence that the language model provides such a linear encoding for the presence/absence of ergative and accusative case. Our results also align with a lazy characterization of the language model's underlying computation. We conclude in Section 8.

2 Background and related work

There has been much interest in evaluating language models' understanding of grammatical phenomena, a practice sometimes known as *targeted syntactic evaluation* (Marvin and Linzen, 2018). LSTM language models have been evaluated on various syntactic phenomena, including subject-verb agreement (Linzen et al., 2016; Bernardy and Lappin, 2017; Kuncoro et al., 2018; Gulordava et al., 2018), negative polarity item licensing (Jumelet and Hupkes, 2018; Marvin and Linzen, 2018) and filler-gap dependencies (Chowdhury and Zamparelli, 2018; Chaves, 2020; Da Costa and Chaves, 2020; Wilcox et al., 2024). They show various levels of success on each phenomenon.

Much research also seeks to *interpret* language models, i.e., understand their internal mechanisms that grant them their performances. One popular approach in this area is to *probe* language models for representations of certain kinds of grammatical information. Typically, this involves extracting the intermediate representations from a language model produced for certain linguistic expressions, and using them to train and evaluate a shallow classifier that predicts some relevant grammatical information associated with these expressions. For example, Tenney et al. (2019) show that BERT representations can be used to predict syntactic categories of and dependency relations between constituents in English.

A common criticism of probing is that it involves training; thus a positive result can't necessarily be attributed to the language model. There are ways to overcome this problem. For example, probing with weak linear classifiers allows one to conclude that the relevant grammatical information is encoded by the language model as a subspace, allowing a geometrically intuitive interpretation of the language model's inner workings. Further, by *counterfactually intervening* the language model's representations using the classifier probe's weights and checking if the intervention affects the language model's inference process, one can check if the language model is actually using the grammatical information the way it is encoded as suggested by the classifier probe. A recent line of work incorporates both of these aspects; for example, Hao and Linzen (2023) find a linear encoding of number in a subspace of BERT's contextualized representations for English, and show that causal intervention in this subspace affects BERT's performance on subject-verb number agreement tasks.

Agreement is a classic example of a syntactic dependency that targets hierarchical structure; a lot of interpretability work has focused on LSTM language models' learning of agreement. Linzen et al.'s (2016) pioneering work shows that LSTMs are capable of predicting English number agreement as a classification task, on which they are trained with explicit supervision. Gulordava et al. (2018) show that LSTM language models naturally learn to predict number agreement correctly in Italian, English, Hebrew and Russian. Lakretz et al. (2019) argue that two units in Gulordava et al.'s (2018) language model track number, which means LSTM language models implement genuine syntactic processing mechanisms.

3 Case and agreement in Hindi-Urdu

In Hindi-Urdu, the participial main verb and any auxiliary agree with the structurally most prominent argument of the verb that is not case-marked overtly (Bhatt, 2005). The subject is more structurally prominent than the object. The overt case marker for subjects is *-ne*, which we will call *ergative case*. The overt case marker for objects is *-ko*, which we will call *accusative case*. For example, when the subject is not marked ergative, the verb and auxiliary agree with the subject no matter whether the object is marked accusative or not (1). This agreement is coded on an aspectual morpheme that immediately follows the verb stem.

(1) Rahul kitaab(-ko) paṛh-taa
Rahul[M] book[F](-ACC) read-HAB;MSG
thaa
be[PST;MSG]
'Rahul used to read a/the book.'

When the subject is marked ergative, agreement targets the object if the object is not marked accusative (2).

(2) Rahul-ne kitaab paṛh-ii thii
Rahul[M]-ERG book[F] read-PFV;F be[PST;FSG]
'Rahul had read a book.'

When both arguments are overtly case-marked, agreement targets neither argument. The result is default masculine agreement, shown in (3), where there are no masculine arguments.

(3) Sita-ne kitaab-ko paṛh-aa
Sita[F]-ERG book[F]-ACC read-PFV;MSG
thaa
be[PST;MSG]
'Sita had read the book.'

While case controls agreement in Hindi-Urdu, case itself is controlled by independent factors. The subject receives ergative case iff its verb is transitive and in the perfective aspect. The object receives accusative case iff it is specific or definite.

4 Current study

As described in the previous section, Hindi-Urdu features a more complex verbal agreement system than subject-verb agreement systems found in languages like English, making it an interesting challenge for language models to learn. In the rest of this paper, we train a LSTM language model on Hindi data, and address the following two research questions concerning this model. First, how well does the model learn the case-agreement interaction in Hindi-Urdu (Section 5)? Second, if learning is successful, how does the model compute agreement using case information (Sections 6–7)? In particular, we employ causal intervention techniques to answer the second question.

5 Training and evaluation

5.1 Training

The training data for our language model comes from the Hindi Wikipedia (Foundation) and the Hindi data from the CC-100 corpus (Conneau et al., 2020; Wenzek et al., 2020), both taken from the Hugging Face website. The data mostly consists of unromanized Devanagari. We perform unsupervised morphological segmentation with Morfessor 2.0 (Smit et al., 2014), which reduced our vocabulary size from 2.4M to 146K. We then discarded all sentences longer than 80 morphemes and converted all morphemes except the most frequent 30000 to a designated UNK(nown) token, giving us about 246M non-UNK tokens. We follow a train:dev:test split of 7:1:2.

We train Gulordava et al.'s (2018) LSTM language model. Due to the limited size of our training data, we decided to train a LSTM language model rather than a Transformer. Gulordava et al. show that their LSTM language models predict Italian number agreement across long-distance dependencies at near-human performance. The architecture of the model is a two-layer LSTM with an embedding size and hidden layer size of 650. We follow the set of hyperparameters that gave Gulordava et al. their best validation set perplexity, which we detail in Appendix A. Our test set perplexity is 47.17, comparable to Gulordava et al.'s results.

5.2 Evaluation

We artificially generate an evaluation dataset intended to test our language model's ability to predict gender agreement correctly. Each data point is a pair $\langle s, \gamma \rangle$ where s is a sentence prefix and γ is a gender label. The sentence prefix s consists of a subject, an object and a verb stem, and should be continued with an aspectual morpheme that shows gender agreement. The correct gender is encoded by the label γ. The data points are manipulated by three conditions: whether or not the subject is marked ergative, whether or not the object is marked accusative, and the genders of the subject and object, which are always different. Table 1 illustrates the kinds of data points generated for each combination of conditions. We combina-

torially generate 320K data points. Most sentence prefixes in the data set are semantically nonsensical, an intended effect; we want the model to rely only on structural properties of the data, not semantic ones.

Evaluation proceeds as follows. For each data point with sentence prefix s and correct gender γ, we compare the conditional probability of the masculine and feminine singular forms of the following four aspectual morphemes given the context s:

(4) a. HAB: habitual (M: ता, F: ती)
b. INF: infinitival (M: ना, F: नी)
c. PFVC: perfective morpheme that begins with the consonant य (M: या, F: यी)
d. PFVV: perfective morpheme that doesn't begin with a consonant (M: ◌ा, F: ◌ी)

Within each aspect, the form corresponding to gender γ should be higher than the form for the incorrect gender. Accuracy is aggregated over the dataset for each aspect. Incorporating results from multiple aspectual forms gives us a more comprehensive evaluation with more generalizable results, unlike previous evaluation work on English subject-verb number agreement that only focuses on one auxiliary pair, e.g. *is*/*are*.

However, it can be misleading to compare accuracy across items or conditions within each aspect, because certain aspectual morphemes are incompatible with certain items and conditions. For example, whether a verb takes the PFVC or the PFVV morpheme in the perfective is lexically specified; a verb takes PFVC iff its stem ends in a vowel (e.g. सजा *sajā*, but not भेज *bhej*). Ergative marking results only in the perfective. A language model with adequate knowledge of Hindi-Urdu may reasonably assign equally low probabilities to the masculine and feminine PFVC forms of the verb भेज *bhej*, and to the masculine and feminine PFVC forms of the verb सजा *sajā* when the subject is not ergative, because all of these forms are ungrammatical. This would result in a low accuracy for PFVC forms.

To address this, we also calculate a form of accuracy that incorporates all aspects. Specifically, for each data point, we compare the probability summed over the masculine forms of all four aspects with the probability summed over the feminine forms of all four aspects. Intuitively, the summation represents marginalization over aspect, allowing us to compare the probability of the two genders directly. We call the accuracy aggregated over the dataset this way *general accuracy*. Table 2 reports the by-aspect and general accuracy for our language model, broken down by subject and object case-marking as well as the correct gender to show agreement for, i.e., the gender label γ.

Additionally, in Table 2, we report the sensitivity index d' for all three case patterns that doesn't result in default masculine agreement. We calculate d' as $z(\text{hits}) - z(\text{FA})$, where z is R's qnorm function, hits is the proportion of true masculine examples correctly predicted masculine, and FA (false alarm) is the proportion of true feminine examples incorrectly predicted masculine. Thus, d' quantifies the language model's sensitivity to the agreement contrast after factoring out any general biases towards masculine or feminine morphemes the model may have.

Among the four aspects, the habitual aspect (HAB) gives the best results, with a high accuracy of 82.69 and a sensitivity index d' of 1.84. In comparison, the other aspects have a slightly above-chance performance. Recall that general accuracy and d' are calculated by comparing the marginal probabilities of the masculine vs. feminine forms, where marginalization is summation over aspects. General accuracy is 81.18 and d' is 1.73, a decent performance. For comparison, Gulordava et al. (2018) train models with the same architecture on Italian, English, Hebrew and Russian data, and evaluate their models using two subject-verb agreement tasks. They report accuracies in the range 67.5–95.2. The results suggest that our language model has reasonably understood the case-agreement interaction in Hindi-Urdu.

6 Characterizing the language model's underlying computation

We see that our language model has learned the case-agreement interaction in Hindi-Urdu to some extent. What kind of computation could our language model be performing in order to determine agreement?

To frame this question more specifically, let's consider what forms this computation can take. A correct Hindi-Urdu agreement computation can be thought of generally as a process that takes case information as input and returns the agreement target as output. For example, it can be modelled as the simulation of a finite-state machine illustrated in Figure 1, where case determines the transitions and the accepting states determine which argument the agreement should target. The simulation keeps

Genders	Cases	Data point $\langle s, \gamma \rangle$	Glossed example for s
masc. subject, fem. object	∅,∅	$\langle NP_{\text{M}}\ NP_{\text{F}}^{\neg A}\ V, \text{M} \rangle$	कुमार एक माता छोड़ Kumar[M] one mother[F] leave
	∅,ACC	$\langle NP_{\text{M}}\ NP_{\text{F}}^{A}\ acc\ V, \text{M} \rangle$	कुमार एक माता को छोड़ Kumar[M] one mother[F] ACC leave
	ERG,∅	$\langle NP_{\text{M}}\ erg\ NP_{\text{F}}^{\neg A}\ V, \text{F} \rangle$	कुमार ने एक माता छोड़ Kumar[M] ERG one mother[F] leave
	ERG,ACC	$\langle NP_{\text{M}}\ erg\ NP_{\text{F}}^{A}\ acc\ V, \text{M} \rangle$	कुमार ने एक माता को छोड़ Kumar[M] ERG one mother[F] ACC leave
fem. subject, masc. object	∅,∅	$\langle NP_{\text{F}}\ NP_{\text{M}}^{\neg A}\ V, \text{F} \rangle$	सीता एक पिता छोड़ Sita[F] one father[M] leave
	∅,ACC	$\langle NP_{\text{F}}\ NP_{\text{M}}^{A}\ acc\ V, \text{F} \rangle$	सीता एक पिता को छोड़ Sita[F] one father[M] ACC leave
	ERG,∅	$\langle NP_{\text{F}}\ erg\ NP_{\text{M}}^{\neg A}\ V, \text{M} \rangle$	सीता ने एक पिता छोड़ Sita[F] ERG one father[M] leave
	ERG,ACC	$\langle NP_{\text{F}}\ erg\ NP_{\text{M}}^{A}\ acc\ V, \text{M} \rangle$	सीता ने एक पिता को छोड़ Sita[F] ERG one father[M] ACC leave

Table 1: Data point templates for each combination of conditions, with examples. In the **Cases** column, ∅ means no overt case-marking; e.g., ∅,ACC means non-overtly marked subject, ACCusative-marked object. In the **Data point** column, each sentence prefix s is described as the right-hand side of a rewrite rule. Uppercase variables are non-terminals: NP_γ stands for a singular noun phrase with gender γ, NP_γ^A specifically stands for one that may be ACC-marked, i.e., specific or definite, $NP_\gamma^{\neg A}$ specifically stands for one that may not be ACC-marked, i.e., not specific or definite. V stands for a verb stem.

ERG?	ACC?	Correct	HAB		INF		PFVC		PFVV		General	
			Acc	d'	Acc	d'	Acc	d'	Acc	d'	Acc	d'
−	−	M	74.18	1.18	57.86	0.33	32.11	0.32	93.76	0.28	78.00	1.04
−	−	F	70.35		55.38		78.36		10.52		60.56	
−	+	M	98.96	3.17	88.84	1.12	72.08	0.97	99.97	2.03	99.53	3.16
−	+	F	80.57		46.09		64.97		8.24		71.22	
+	−	M	95.25	2.55	59.52	1.14	56.10	1.41	97.05	1.62	84.42	1.96
+	−	F	81.02		81.44		89.63		39.49		82.82	
+	+	M	83.30		68.38		79.68		99.88		91.60	
Average			82.69	1.84	65.27	0.77	68.00	1.08	66.80	1.17	81.18	1.73

Table 2: Accuracy and d' for our language model evaluated on the case-agreement dataset.

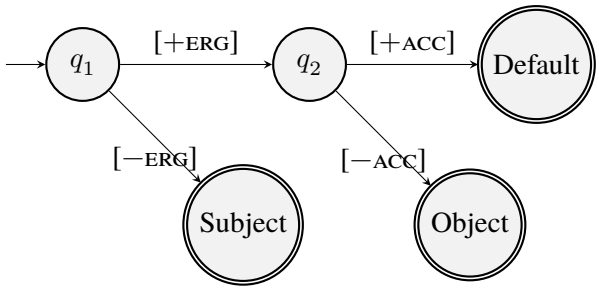

Figure 1: Agreement computation as a finite-state machine.

track of the current state, and follows transitions depending on subject and object case.

In order for the language model to implement such a simulation, it needs to represent case information somehow. But exactly how does it represent case? We take a very specific hypothesis to this question: our language model linearly encodes case in a subspace of its hidden layer vectors. That is, there is a subspace for ergative case, such that the hidden layer vectors for sentences with an ergative-marked subject are linearly separable from those for sentences with an non-ergative-marked subject when both sets of vectors are projected onto this subspace. In other words, we can use an ensemble of linear binary classifiers to predict the presence of ergative marking in a sentence from its hidden layer vector representation. The same applies to accusative case. Under this hypothesis, our language model implements a highly interpretable syntactic processing mechanism.

Aside from representations of the input, we can also consider other aspects of this computation. One dimension along which we can characterize alternative forms of computation is memory usage. This places an *eager* and *Markovian* computation on one end of a spectrum, and a *lazy* and *memory-intensive* computation on the other end. These two computations differ in how soon they advance the simulation as they process linguistic input. As soon as an eager and Markovian computation processes case information, it advances the simulation by following the corresponding transition. A lazy and memory-intensive computation would store the subject and case information, and performs the entire simulation in one fell swoop when it reaches the verb stem, just in time before it needs to compute agreement. Where is our language model's underlying computation located along this eager/lazy spectrum?

In the next section, we use linear classifier probes and causal intervention techniques to investigate whether our language model encodes case linearly, and how eager/lazy it is at advancing the simulation.

7 Investigating the language model's underlying computation

For our first investigation, we first explore the hypothesis that the language model linearly encodes the presence/absence of each case-marking as a subspace in its hidden layers. To do this, we first use a method known as *iterative nullspace projection* (INLP) to find three sets of orthonomal basis vectors that identify a potential case subspace; two for ergative, and one for accusative. We then rerun the evaluation described in Section 5.2, but intervening on the subject and object representations, reflecting them onto the "opposite side" of the case subspaces, effectively making the representation of a case-marked argument not case-marked, and that of a non-case-marked argument case-marked. We check how effective the intervention is by measuring how intervention affects the language model's performance. An effective intervention suggests the subspace identified by INLP really is how the language model encoding case.[1]

7.1 Method: intervention

Intervention is a process that takes three things as input: a vector $x \in \mathbb{R}^d$, which is a representation produced by our language model, a set of orthonormal basis vectors $\mathbb{B} = b_1, \cdots, b_k \in \mathbb{R}^d$, which identifies a subspace that encodes case, and an *intensity parameter* $\alpha \geq 1$. First, for each $j = 1, \cdots, k$, calculate λ_j, the scalar projection of x onto b_j with $\lambda_j = x^\top b_j$. Then, return the intervened vector $x' \in \mathbb{R}^d$, calculated as $x' = x - \alpha \sum_{j=1}^{k} \lambda_j b_j$. The interpretation of x' depends on α. When $\alpha = 1$, x' is the projection of x onto the nullspace of the case subspace; x' then represents x but with all case information removed. When $\alpha = 2$, x' is the reflection of x onto the opposite side of the case subspace; x' then inverts the case information of x. For example, if $\mathbb{B}$ represents the ergative subspace, and x represents an ergative-marked argument, then x' represents the same argument as x except it's non-ergative-marked. Any $\alpha > 2$ pushes x' further in the opposite case direction, intensifying the effect of the intervention.

[1] Our description of intervention and INLP largely follows Hao and Linzen's (2023) presentation.

7.2 Method: iterative nullspace projection

To perform intervention with respect to a case subspace, we first need a set of orthonormal basis for that subspace. Iterative nullspace projection (INLP) (Dufter and Schütze, 2019; Ravfogel et al., 2020) is a supervised method to help us find the bases for a subspace of interest. We describe INLP for identifying the ergative subspace; the same process works for the accusative subspace. First, we designate a training split of the evaluation dataset, and run the language model on each sentence prefix $s^{(i)}$ of the training split to obtain a hidden layer vector $h^{(i)}$ at some position of interest. Each $h^{(i)}$ is paired with a binary label $c^{(i)}$ representing the presence/absence of ergative case in that sentence prefix. Then, we train a linear classifier to predict $c^{(i)}$ from $h^{(i)}$. The normalized weights of the classifier, a vector in $\mathbb{R}^d$, is taken to be the first basis b_1. For each additional jth basis we'd like to find, we train another linear classifier the same way, except we preprocess the input $h^{(i)}$ by intervening it with the first $j-1$ bases and intensity $\alpha = 1$, removing the ergative case information captured by the first $j-1$ bases. We train each classifier using gradient descent, which guarantees that the new classifier weight b_j is a weighted sum of the preprocessed inputs $h^{(i)}$. Since the preprocessing projects each $h^{(i)}$ onto the nullspaces of the first $j-1$ bases, b_j is guaranteed to be orthogonal to all of $b_1, \cdots b_{j-1}$.

7.3 Evaluation with causal intervention: is case encoded linearly?

We perform a 50-fold cross validation on the evaluation dataset, with a training split of 6.4K data points in each fold. For the ergative subspace, we run INLP on hidden layer vectors obtained from two positions: one set after processing the subject, and another after processing the object. For the accusative subspace, we run INLP on hidden layer vectors obtained after processing the object. This gives us three sets of bases: one for the post-subject ergative subspace, one for the post-object ergative subspace, and one for the post-object accusative subspace.

The remaining 313.6K data points in each fold is used for evaluation. We re-run the evaluation described in Section 5.2, while performing causal intervention with respect to each one of the three case subspaces at the appropriate location. For example, for the post-object ergative subspace, we feed each sentence prefix into our language model, and pause once the model processes the object. We intervene the hidden layer vectors with respect to the post-object ergative subspace using some intensity α, and resume model inference using the intervened hidden layer vectors, effectively flipping the presence/absence of ergative marking. We compare the agreement performance of the language model before and after the intervention to see how successful the intervention was. We use sensitivity index (d') to quantify model performance. The results are shown in Figure 2 for ergative intervention and Figure 3 for accusative intervention. We present the results for $\alpha = 5$ just as Hao and Linzen (2023) did, noting that lower values for α doesn't change our results qualitatively.

Let's first consider ergative intervention. We believe ergative case information should be the most recoverable at the post-subject position; hence in this section, we only look at the results of the post-subject ergative intervention. In the [-ERG,-ACC] condition, agreement should target the subject. A successful ergative intervention should assimilate this to the [+ERG,-ACC] condition, where agreement should target the object. Indeed, we see that the agreement performance flips to the opposite prediction, as d' drops below zero. In the [-ERG,+ACC] condition, agreement should target the subject. A successful ergative intervention assimilates this to the [+ERG,+ACC] condition, which requires default agreement. This should be reflected as chance performance, which is exactly what we see in our results. Finally, in the [+ERG,-ACC] condition, agreement should target the object. A successful intervention assimilates this to the [-ERG,+ACC] condition, where agreement should target the subject. However, our ergative intervention only drives the agreement performance to near-chance level, not exactly reversing the agreement predictions.

Let's turn to accusative intervention. In the two [-ERG] conditions, a successful accusative intervention shouldn't affect agreement computations, because agreement should always target the subject if it isn't ergative-marked. Indeed, our intervention doesn't change the agreement predictions qualitatively, as it remains above chance in both conditions. In the [+ERG,-ACC] condition, agreement targets the object. A successful accusative intervention should cause agreement to fall back to default masculine. However, our intervention keeps the agreement above chance, which means agreement is still targetting the object.

Thus, we have found positive evidence that our

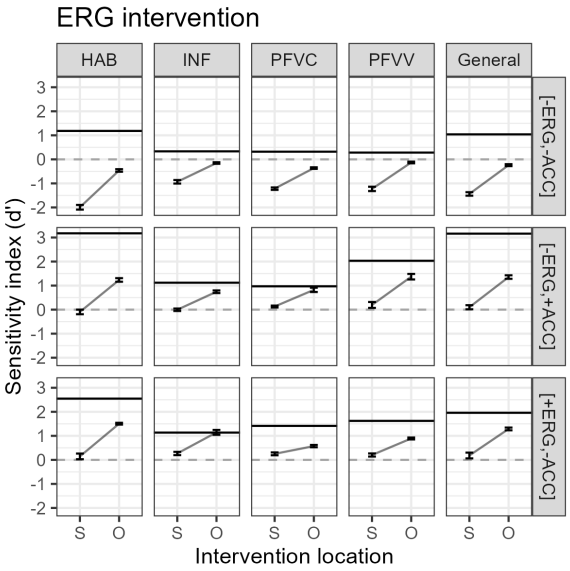

Figure 2: Ergative intervention results. The two intervention locations are post-**S**ubject and post-**O**bject. The light dashed line is drawn at $d' = 0$, indicating chance performance. The dark solid line indicates the original performance of the language model before intervention. Error bars indicate one standard error average across cross validation.

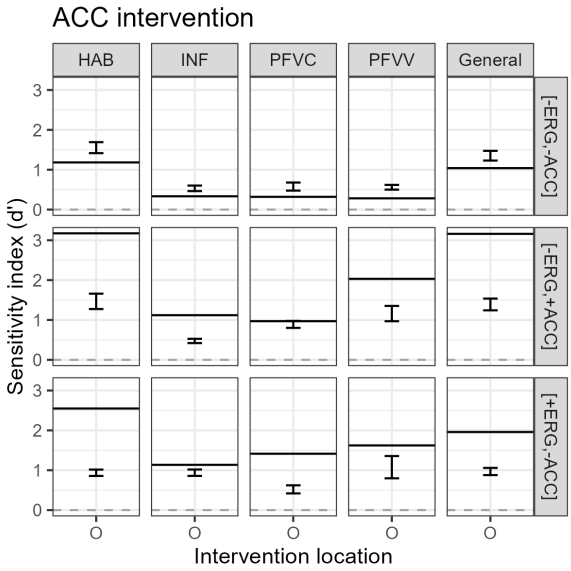

Figure 3: Accusative intervention results. The intervention location is post-**O**bject.

language model uses a linear encoding of ergative and accusative case marking, and uses this encoding to calculate agreement.

7.4 Is agreement computation eager or lazy?

For our second investigation, we check whether our language model aligns more with an eager or a lazy characterization of agreement computation. We suggest that looking at the effectiveness of the post-object ergative intervention may give us a clue, because it should only be effective in a lazy, but not an eager, computation. An eager computation would use the ergative case information to advance the simulation as soon as it processes the subject, discarding that information, while a lazy computation would store the ergative case information until it sees the verb. Looking at Figure 2 again, we observe that post-object ergative intervention is still effective, although the magnitude of the intervention effect is smaller than post-subject intervention. This suggests our language model isn't computing agreement in a purely eager way.

Although this by itself is a very weak conclusion, we think that the general method of causal interventions with respect to linear encodings we pursue here can be extended in interesting ways to help us better understand the underlying computation of language models. For example, we plan to perform the same analysis we describe in this paper to Transformers. We think the autoregressive and recurrent nature of the LSTM architecture create an incremental processing pressure that encourages performing computations on the fly, while Transformers aren't subject to this pressure. Thus, we expect Transformers to show signs of a lazier computation than our LSTM language model.

8 Conclusion

In this paper, we train a LSTM language model on Hindi data and show that it has learned case-agreement interactions in Hindi-Urdu, predicting correct gender agreement 81.17% of the time. We further show that our language model has learned to encode case information in a low-dimensional subspace of its hidden layer vectors, where case-marked arguments are linearly separable from non-case-marked arguments. In addition, our model uses case information encoded this way as part of its agreement computation. Preliminary evidence also suggests that our language model doesn't calculate agreement extremely eagerly, as our causal intervention methods reveal that case information seems to be persistent over time as the language model processes a sentence. The general method described in this paper can be adopted to study interesting phenomena concerning case and agreement in other languages.

Limitations

We see two limitations in our work, which both concern interpreting our causal intervention results. The first limitation is that we don't know how much of the effectiveness of our case interventions is meaningful. For example, Figure 2 shows that post-subject ergative intervention in the [−ERG,−ACC] condition decreases general d' by about 2.5. Can all of this 2.5 point decrease be attributed to successful ergative intervention? For example, if we had performed multiple post-subject interventions, each time with respect to a set of randomly generated orthonormal basis vectors, and observed a d' decrease in the range 1.5 to 3, then our ergative intervention result wouldn't be meaningful, since just any intervention would affect d' in a similar way. We plan to add a comparison between our current results and intervention with respect to random bases in a future version of this paper.

The second limitation is that we presently offer no way of quantifying how lazy or eager our language model's underlying computation is. This would be possible if we know how effective we would expect post-object ergative intervention to be under a fully lazy and a fully eager computation. While a fully lazy computation should result in equal effectiveness between post-object and post-subject ergative intervention, we don't know how effective a fully eager computation should be. In the future, we hope to consider alternative ways of quantifying the eagerness/laziness of our language model's underlying computation.

References

Jean-Phillipe Bernardy and Shalom Lappin. 2017. Using deep neural networks to learn syntactic agreement. *Linguistic Issues in Language Technology*, 15(2).

Rajesh Bhatt. 2005. Long distance agreement in Hindi-Urdu. *Natural Language & Linguistic Theory*, 23:757–807.

Susana Béjar and Milan Rezac. 2009. Cyclic Agree. *Linguistic Inquiry*, 40(1):35–73.

Rui Chaves. 2020. What don't RNN language models learn about filler-gap dependencies? In *Proceedings of the Society for Computation in Linguistics 2020*, pages 1–11, New York, New York. Association for Computational Linguistics.

Shammur Absar Chowdhury and Roberto Zamparelli. 2018. RNN simulations of grammaticality judgments on long-distance dependencies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 133–144, Santa Fe, New Mexico, USA. 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.

Jillian Da Costa and Rui Chaves. 2020. Assessing the ability of transformer-based neural models to represent structurally unbounded dependencies. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 12–21, New York, New York. Association for Computational Linguistics.

Philipp Dufter and Hinrich Schütze. 2019. Analytical methods for interpretable ultradense word embeddings. 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 1185–1191, Hong Kong, China. Association for Computational Linguistics.

Wikimedia Foundation. Wikimedia downloads.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Sophie Hao and Tal Linzen. 2023. Verb conjugation in transformers is determined by linear encodings of subject number. *Preprint*, arXiv:2310.15151.

Jaap Jumelet and Dieuwke Hupkes. 2018. Do language models understand anything? on the ability of LSTMs to understand negative polarity items. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 222–231, Brussels, Belgium. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. 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 11–20, Minneapolis, Minnesota. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online. Association for Computational Linguistics.

Shauli Ravfogel, Yoav Goldberg, and Tal Linzen. 2019. Studying the inductive biases of RNNs with synthetic variations of natural languages. 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 3532–3542, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24, Gothenburg, Sweden. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *Preprint*, arXiv:1905.06316.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Ethan Gotlieb Wilcox, Richard Futrell, and Roger Levy. 2024. Using computational models to test syntactic learnability. *Linguistic Inquiry*, 55(4):805–848.

Annie Zaenen, Joan Maling, and Höskuldur Thráinsson. 1985. Case and grammatical functions: the Icelandic passive. *Natural Language & Linguistic Theory*, 3(4):441–483.

A Hyperparameters

Our LSTM had a hidden layer size of 650. We trained with an initial learning of 20, gradient clipping with a maximum L2 norm of 0.25, truncated backpropagation through time with window size 35, a dropout rate of 0.2 for 40 epochs. Whenever the validation set perplexity doesn't improve in a new epoch, the learning rate is scaled by a factor of 0.25.