VQ-VAEs & Gumbel-Softmax for Language-Aware POS Induction

Anjani Ganapathy

Word count: 9655

Abstract

We investigate unsupervised part-of-speech (POS) tagging for Hindi using Vector-Quantized Variational Auto-Encoders (VQ-VAEs) with Gumbel-Softmax. Our approach addresses Hindi's morphological richness and flexible word order through three experiments: testing Hidden Markov Model assumptions, incorporating character-level embeddings, and adding character reconstruction. Using the Universal Dependencies Hindi corpus, we achieve 47.41% Many-to-One accuracy. Results show that linguistic assumptions tailored to Hindi's free word order improve performance more than morphological features. This work provides the first neural unsupervised POS induction baseline for Hindi and demonstrates that language-specific architectural decisions can enhance unsupervised syntactic learning in morphologically rich languages with flexible syntax.

1: Introduction

A crucial milestone in natural language acquisition is learning syntactic groups. This feature is a fundamental part of what makes natural language infinitely productive. Humans are able to learn these without explicit instruction and later learn explicit labels like “noun” for these groups. In machine learning, this kind of learning is called unsupervised learning. Unsupervised learning requires a model to categorize input data without knowing what each cluster should represent.

This is a difficult task that comes with its own challenges. First, unsupervised models are very sensitive to the data they are trained on, with noisy or unclean data causing issues. Additionally, the instantiation of these models can determine how they converge, which means that results can differ between training instances. Evaluating unsupervised models is also difficult, especially if using an unlabeled dataset for training and testing.

In this paper, we investigate the application of unsupervised learning to part of speech (POS) tagging, or POS induction. POS tagging is not a new area of work; rather, it is a common application of multiclass classification in NLP where each class corresponds to a POS tag to be predicted. This is an example of a structured prediction problem in machine learning. Structured prediction problems involve predicting sequences of discrete, inter-connected labels. POS tags are quite interdependent in sequences, and rules governing where tags can appear differ between languages. Computational approaches to POS tagging are broad, but every approach attempts to efficiently solve this version of the structured prediction problem.

For this task, we created a variational auto-encoder that utilizes vector quantization to compress input data into a learned codebook that represents each predicted tag. VQ-VAEs are a powerful way to learn discrete representations, for example, of classes like POS tags, without supervision (Van Den Oord et al**.,** 2017). These have been successfully applied to other generative applications such as speech and video generation. On top of the VQ-VAE, we implement the Gumbel-Softmax procedure that enables us to sample from categorical distributions produced by the encoder component of our architecture, while creating a continuous layer that does not interfere with backpropagation (Jang et al., 2016; Maddison et al., 2016).

We chose to apply this methodology to accomplish unsupervised POS tagging for Hindi, one of the most highly spoken languages in the world. Hindi has relatively flexible word order and high morphological richness, and we are curious to understand how these linguistic features impact learning. Prior work in POS induction tended to prioritize creating methods that are less linguistically-driven but that are effective across several languages (Zhou et al., 2022; Gupta et al., 2022; Stratos, 2019). The languages that have been tested in these works tend to skew towards European languages that are prominent in research. We aim to diversify this set of languages incrementally with this work.

Our prediction is that providing a model with information and assumptions tailored to Hindi's linguistic context will improve its ability to learn syntactic groupings even in an unsupervised setting. We test this by first creating a baseline model that is similar to others used in recent POS induction work and incrementally adding components to it over three experiments. The first tests the effect of predicting original words only based on their latent tag rather than the latent tags of the entire sentence. The second tests the impact of providing the model with character-level information on its latent tag assignments. The last experiment tests the impact of training the model to reconstruct characters when character-level information is provided to it.

The results of these experiments will enable us to answer the following research questions.

1. How effective are current POS induction methods for Hindi? What factors contribute to this efficacy?
2. What kinds of linguistic features are helpful to POS induction?

These answers will provide insight into the utility of including linguistically-relevant features for unsupervised POS tagging, which could extend to other NLP tasks.

2: Literature Review

2.1: Supervised, Unsupervised, and Semi-supervised POS Tagging

Supervised models are the most straightforward method of POS tagging. Learning from labeled data is an easier task, the POS tags to be learned are defined, and it is easier to compare results to prior work due to its prevalence. However, supervised training for POS tagging relies on access to high-quality, POS-annotated corpora. These exist for many languages through Universal Dependencies treebanks, but these treebanks can be small - often in the range of a few thousand training sequences. One example of a supervised POS tagger is Stanza (Qi et al., 2020), an open-source Python language-processing toolkit that supports 60+ languages. Stanza achieved high levels of accuracy in several languages by training on Universal Dependencies (UD) treebanks.

Because unsupervised models have to learn without answers, they more closely mimic the human cognitive process of acquiring syntactic groups in natural language. Training an unsupervised model to infer meaningful groups from unlabeled data could reveal something about what kinds of patterns the model recognizes. Additionally, because the training data does not need to be annotated to be used in an unsupervised model, this opens up more possibilities to the kinds of corpora that could be used to train the model. An issue with POS induction is that the model does not have a pre-defined set of tags to learn. Instead, it must be given a target number of groups to learn during training, and in evaluation, the researcher has to associate these with real linguistic syntactic groups. How this should be done is not standard; Christodoulopoulos et al. (2010) reviewed evaluation methods for POS taggers and found that approaches varied widely.

Semi-supervised models combine both supervised and unsupervised models in one integrated network to reap the benefits of both training approaches. This means that the model can take advantage of both labeled and unlabeled data, which is helpful in situations where diverse and representative annotated data is difficult to find. The work of Stratos & Collins (2015) tested varying levels of supervision by training the model with labeled data and unlabeled data in incrementally-updated amounts. They found that training on as few as 400 labeled words resulted in over 97% accuracy, extending to German and Spanish, and concluded that a

simple classifier can accurately predict POS tags when provided with the most informative training data.

2.2: Computational Techniques for POS Induction

POS induction has typically been approached with the goal of inferring latent tags from training data. The latent tags are intended to represent real linguistic parts of speech and must be optimized somehow to resemble their likeness to them. This task has been accomplished by using a range of computational methods.

2.2.1: Hidden Markov Models (HMMs)

Prior approaches to POS induction have prioritized making stronger independence assumptions and used dynamic programming methods (Christodoulopoulos et al., 2010). Examples of this work have used HMMs as well as Conditional Random Field (CRF) autoencoders and clustering techniques. HMMs in particular are a helpful way to model linguistic structure due to the assumptions that they make, which will henceforth be referred to as HMM assumptions *a* and *b* respectively.

1. Hidden Markov Model Assumptions for POS tagging
    a. $P(t_i|t_{i-1})$ - the probability of a POS tag at position *i* is dependent only on the POS tag at position *i-1*.
    b. $P(w_i|t_i)$ - the probability of a word at position *i* in a sequence is dependent only on the tag of that word.

These assumptions make HMMs a computationally-tractable method to accomplish POS tagging while still accounting for the syntactic interdependence of linguistic categories. Many variations of using HMMs to achieve POS induction exist, each with their advantages. The work of Merialdo (1994) used maximum likelihood estimates (MLE) to optimize a HMM; results showed that this approach is effective but relies on substantial quantities of training data. Goldwater & Griffiths (2007) used a fully Bayesian approach to select optimal model parameters and found that this performs better than the MLE approach.

A limitation of HMMs is that they encode information in one direction; however, useful syntactic information can and does exist in both directions of a target word. Additionally, the assumptions that they make are not necessarily reflective of language.

2.2.2: Deep learning

In more recent work in POS tagging, bidirectional LSTMs (Bi-LSTMs) have been utilized for their ability to capture sequential dependencies when parsing a sequence in both directions. Stanza (Qi et al., 2020) was unprecedented as the first toolkit of its kind that was made fully neural and used language-agnostic architecture, using a Bi-LSTM along with pre-trained word and character embeddings (Qi et al., 2018). Plank et al. (2016) tested Bi-LSTMs' abilities in POS tagging when aspects like language, input representations, and dataset size were changed. The results of that work showed that character-level representations were clearly helpful for these models, and when combined with word representations resulted in the best performance.

Methods like LSTMs and Transformers use the context around a target for making predictions. This does not align with the assumptions that HMMs make, which limit the context used to make predictions. However, Bi-LSTMs have so far shown promising results in POS induction for the languages they have been tested on, which seems to indicate that the contextual information is helpful in learning.

2.2.3: Vector-Quantized Variational Auto-Encoders (VQ-VAEs)

A recently-proposed method to accomplish the discrete categorization that POS induction requires is the VQ-VAE, introduced by Van Den Oord et al**.** (2017). VQ-VAEs provide a way of encoding information into discrete representations that combine variational auto-encoders with vector quantization to force an encoder to compress data into a finite set of groupings.

In order to understand VQ-VAEs, it is important to first understand Variational auto-encoders (VAEs). VAEs have featured frequently in research since they were originally proposed in Kingma & Welling (2013)**.** VAEs are built on top of basic Encoder-Decoder architecture, shown in Figure A. The motivation behind building these models is to take in some input

data, learn (or “encode”) a representation of it, and then output (or “decode”) it in a different form.

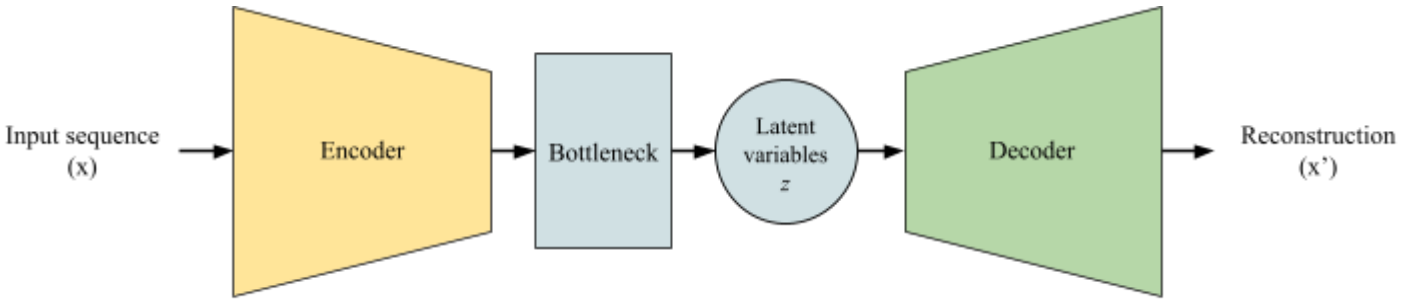


Figure 1: Basic Auto-encoder Architecture

An auto-encoder is a type of Encoder-Decoder network in which the Encoder compresses some input x into a latent space and the Decoder attempts to reconstruct the input from the latent space. We use the accuracy of this reconstruction to evaluate how well the Encoder compresses, or represents, the input data in the latent space. A Variational Auto-Encoder (VAE) is one type of auto-encoder that adds probabilistic properties to the latent random variables it generates. In VAEs, the encoder outputs are used to learn a posterior distribution $Q(z|x)$ that represents the probability $P(z|x)$, which is the expected probability that input x is part of class z. The posterior distribution is usually Gaussian and can be represented as $\mathcal{N}(\mu,\sigma)$. The model tries to match a “prior” distribution, $P(z)$, during training, which is the expected probability distribution of the latent space. The decoder in a VAE then tries to reconstruct the original input by learning a probability distribution for $P(x|z)$, the probability that the output is word x for a given label z.

VQ-VAEs modify VAEs by discretizing the latent space to better categorize input. This approach works well for cases in which it is more useful to categorize information rather than represent it with a continuous distribution. This includes modeling of language or image description.

A distinguishing feature of VQ VAEs is the presence of a codebook. This is a set of vectors in the latent space corresponding to the specified dimension of that space. In our research, the number of latent tags specified for prediction corresponds to the number of vectors in this codebook. For each vector outputted from the encoder, the closest vector in the codebook is found by calculating the L2 norm between them. These codebook vectors are then fed into

the decoder. Although a useful technique, the discretization of the latent space in this way poses challenges for backpropagation, as discrete layers are non-differentiable. This issue can be resolved using a version of the reparameterization trick (Kingma et al., 2015), which introduces a parameter that can be used to express the discrete sample in a more continuous way.

Applications of VQ-VAEs to POS induction are uncommon; the most recent examples of this work that inspired our work are those of Gupta et al. (2022) and Zhou et al. (2022). The former applied VQ-VAEs, which they called "deep clustering", for English POS induction in an effort to create a better probing method for large language models. They used pre-trained embeddings from BERT and multilingual BERT (mBERT) to create latent tags. Their model was able to achieve competitive performance in POS induction when trained and evaluated on treebanks for 10 different languages (McDonald et al., 2013). Similarly, Zhou et al**.** (2022) performed masked language modeling for unsupervised POS tagging. Their architecture implements a masked Bi-LSTM to capture longer range, bidirectional dependencies. Their model architecture adopted VQ-VAE infrastructure and was trained on data from both Universal Dependencies treebanks and the Penn WSJ Treebank. The results from this work were mixed, finding that the usage of long-range dependencies was helpful for some languages, like English and German, but not in others like Korean and Indonesian. A takeaway was that this kind of contextual information is helpful for languages with richer morphological agreement, but creates noise when used for languages with more isolating morphology or less long-range dependence between words.

2.2.4: Gumbel-Softmax

The discrete latent representations that VQ-VAEs enable are very useful for categorical learning, but they create a non-differentiable latent space that is difficult to backpropagate through. One way of handling this is to use "straight-through" backpropagation, in which gradients from the Decoder are simply copied to the Encoder. An alternative method was proposed by Jang et al. (2016) and Maddison et al. (2016) and is called Gumbel-Softmax. This procedure serves to replace these categorical distributions with a differentiable, continuous approximation during training. This is a type of reparameterization trick that specifically deals with categorical distributions using discrete latent random variables.

The Gumbel-Softmax procedure evolved from the Gumbel-Max trick, which is a clever method to sample from a categorical distribution. It does this by first sampling from the standard Gumbel distribution (Gumbel, 1954) to get Gumbel noise g for every category. This process is shown in Equation 1.

$$g = -log\,(-log\,(u))\ where\ u \sim Uniform\,(0,\ 1) \tag{1}$$

This noise is added to the log-probabilities associated with each category. Lastly, it applies the argmax function over every resulting value in order to pick the category with the highest value as the sample. This trick can be summarized by the following equation, where π_i represents the probability of class *i*, g_i is Gumbel noise for class *i*, and *z* is the resulting sample drawn from a categorical distribution.

$$z = one\_hot(argmax[g_i + \log \pi_i]) \tag{2}$$

Gumbel-Softmax adapts this trick by applying the softmax function instead of the combined argmax and one-hot functions. This creates a more continuous distribution over the various categories, rather than the discrete distribution created by argmax. This function has the effect of generating sample vectors *y* for each class *i*, and can be described by Equation 3.

$$y_i = \frac{\exp\left((\log\left(\pi_i\right) + g_i)/\tau\right)}{\sum_{j=1}^{k} \exp\left((\log\left(\pi_j\right) + g_j)/\tau\right)} \tag{3}$$

An important parameter for Gumbel-Softmax is the temperature parameter, τ. This parameter controls the extent of discreteness versus continuity of the samples produced. A very small τ (close to 0) will result in a more discrete distribution of samples, while a higher value of τ results in a more continuous distribution. A very high value of τ will result in distributions that approach the uniform distribution. In the case of POS induction, simulated annealing for τ could be helpful. Setting a higher value to for τ in earlier epochs and decreasing it over time could allow the model to prioritize exploration of all latent classes initially, and then of certain classes once the model can more meaningfully identify them.

2.3: POS Tagging in the Context of Hindi

Hindi is a morphologically rich language, with a comprehensive set of affixes for tense, gender, and number inflection (Butt, 2001). On top of these inflections exist a variety of case markers and adpositions that convey further information. Hindi also uses a small set of auxiliary verbs to communicate aspect and mood. Affixes to represent verb tenses can be found on verbs and auxiliaries. Case markers are adpositions and typically are found after nouns, proper nouns, and pronouns. Inflection for gender and number agreement can be found in nouns and adjectives. The highly inflectional nature of Hindi can be an advantage for POS tagging.

Hindi also has relatively free word order, like other languages of this category. All permutations of subject, verb, and object are acceptable in simple sentences, though tone and pragmatic meaning can vary. This could pose a challenge to context-dependent POS taggers, as the context around a particular word is not constant. For example, a simple sentence of the structure subject-object-verb (SOV) can mean the same thing as a sentence of the structure subject-verb-object (SVO), or even one that solely consists of V (Butt, 2001). Hindi does have the canonical order of SOV, and it is likely that a training dataset would mostly contain sentences with that order, but even a small amount of variance within the training data could make it more difficult for a model to learn contextual dependencies.

In comparison to the amount of work that has been done in POS induction, not much has focused on achieving this in Hindi to our best knowledge. One of the most recent works in this space was conducted by Singh et al. (2006), who investigated whether the lack of large training corpora impacted POS tagging in Hindi. This work found that the morphological richness of Hindi can offset a lack of annotated training data.

2.4: Gaps in POS Induction Research

Though POS induction is well-researched, our survey of the work reveals the following gaps. We do not currently know how current methods would work on Hindi POS induction. We also do not have an understanding of how linguistic features should affect unsupervised modeling decisions. Additionally, we do not have a great comparison between Bi-LSTMs and simpler approaches like feedforward networks (FFNs), which vary in context usage, for

different languages. Lastly, the question of how different kinds of embeddings affect learning and whether these are helpful in particular for morphologically-rich languages has not yet been answered.

3: Setup for Experiments

3.1: Experiments

Our work aims to answer questions not yet addressed by current POS induction research. We investigate the impact that information and assumptions given to the model during training have on induction. In particular, we test the efficacy of HMM assumption *b* along with character-level embeddings and reconstruction. We predict that providing the model with these additional components will improve the model's performance. To test this hypothesis, we conducted three experiments; one for each tested component. We chose the order of these experiments to first choose the best base architecture and then enhance the inputs and feedback used by the model in training.

3.1.1: Experiment 1 - Testing HMM Assumptions

This experiment tests HMM assumption *b* by comparing the baseline Bi-LSTM decoder to a FFN decoder for word reconstruction. Bi-LSTMs appear to be the most used neural method for decoding based on our survey of related work; for example, Zhou et al**.** (2022), because they use contextual information in both directions of a sentence to inform predictions.
We test this assumption because of Hindi's relatively free word order, which brings up the question of whether providing the model with sequential context is helpful in reconstruction. Using a FFN instead of Bi-LSTM decoder allows us to incorporate HMM assumption *b* because it attempts to reconstruct a word at position *i* (w_i) with only that word's latent tag information (t_i). In contrast, a Bi-LSTM decoder reconstructs w_i using the latent tags of the entire sequence ($t_{i...W}$). This experiment essentially compares using $P(w_i \mid t_{i...W})$ to $P(w_i|t_i)$. We do not test the other HMM assumption.

3.1.2: Experiment 2 - Testing Character Embeddings

Experiment 2 tests the value of using character embeddings to capture morphological information for Hindi. For example, the case marker *ka*/*ki* follows nouns to mark the genitive case (Butt, 2001). Using character embeddings could make adpositions like this easier for the encoder to identify and group. The pre-trained BERT that we use, MuRIL (Khanuja et al.**,**

2021), unfortunately does not tokenize Hindi words in a morphologically-driven way, as they use BPE tokenization. We hypothesized that learned character embeddings could help the encoder learn representations of affixes, postpositions, and case markers. By combining character embeddings with MuRIL embeddings and projecting these to a word-level, we hoped to create more informative word representations with which latent tags could be predicted. Both Qi et al. (2020) and Zhou et al**.** (2022) used character embeddings in their work to a positive effect.

3.1.3: Experiment 3 - Testing Character-level Reconstruction

This tests the efficacy of a decoder that reconstructs sequences at a character level as well as at a word level. For our research question, it is less important that a model reconstructs the word exactly and more important that it can reconstruct something close to it or in the same category. Character level reconstruction could emphasize creating words with similar morphological features. Our character-level decoder is a language model that uses a unidirectional LSTM to reconstruct each input word by character. We decode each word in a batch of sequences in parallel. We use this decoder in addition to the word level decoder to evaluate the meaningfulness of the encoder-generated latent space. This method of decoding retains HMM assumption *b* because the decoding of each word in a batch is done simultaneously. This experiment also tests whether decoding at a character level works well in tandem with character embeddings.

3.2: Model Architecture

To accomplish our intended research, we created a neural method of POS induction for Hindi. For this, we utilize a VAE with BERT contextual embeddings, vector quantization, and Gumbel-softmax. The following sections outline our model components.

3.2.1: BERT-Informed Encoder

We use Google's MuRIL ((Multilingual Representations for Indian Languages) to generate contextual embeddings for Hindi (Khanuja et al., 2021). This model was created in an effort to improve multilingual systems' sub-optimal performance in Indian languages. MuRIL uses a BERT base architecture and was trained on 16 South Asian languages as well as English.

With MuRIL, we generate token-level contextual embeddings that are passed through a linear layer to compress contextual information into a bottleneck, generating latent random variables that represent a predicted tag for each word. The goal of this component is to learn a posterior distribution $P(t|w)$, the probability that a word w is of tag t. Because we use contextual embeddings for this, it follows that the probability of a tag at position i in a sequence of length W can be represented as $P(t_i|w_{i...W})$.

3.2.2: Compression Using Gumbel-Softmax and Vector Quantization

Gumbel-Softmax is applied to the output of the encoder in order to generate a random, but representative, sample from the categorical distribution that the encoder generates. The encoder's output distribution is categorical due to the nature of classification and thus creates issues for backpropagation. Gumbel-Softmax softens the output of the encoder to generate continuous distributions from which samples can be drawn in our model's forward passes, and that are differentiable for backpropagation. These samples are associated with vectors in the latent space through our application of vector quantization. Vector quantization discretizes the latent space, which is a natural fit for classification. For each word in a sequence, we encode it to create a probability distribution over a number of tags K that we define. These tags correspond to a learned codebook with K Q-dimensional vectors.

3.2.3: Reconstruction Decoder

The predicted label vectors are then passed through a decoder that attempts to reconstruct the original input from the information in the latent space. As mentioned above, the type of neural network used in the decoder and what is being reconstructed varies based on the experiment. We calculate loss based on the accuracy of reconstruction and this informs the model's adjustments of weights and biases in its next steps.

3.3: Dataset Used

To train and evaluate our model, we used the Universal Dependencies (UD) HDTB Hindi annotated corpus containing a training set of 13k sentences, a validation set of 1.6k sentences, and a test set of 1.6k sentences. The UD dataset was created by Bhat et al. (2017) and primarily sources from Hindi newswire treebanks. Because speech in the news domain tends

to be of a formal register, it is likely that the sequences in the dataset tend to utilize canonical syntactic structures.

Though we use unsupervised methods, having annotated data allows us to compare predicted tags to gold tags and create confusion matrices for evaluation. The Universal Dependencies corpus has 17 universal POS tags in the Hindi corpus; we use 14 of these for training and evaluation. The symbol tag does not exist in the corpus at all and the interjection tag does not exist in the test corpus. We also clean the dataset of punctuation for simplification; this removes noise in the form of words that have punctuation in them. Lastly, we do not predict the unknown tag at all.

The proportion of each POS tag in the training and test datasets are visualized in Figure 2. The distribution of tags is skewed towards the categories of Noun, Adposition, Proper Noun, and Verb. More than 60% of the words in both of these datasets can be categorized into these 4 classes, with the remaining 11 classes accounted for by less than 40% of the data.

The distribution of sentence lengths in the training dataset is visualized in Figure 3. The majority of sentences are between 9 and 32 words in length and it is uncommon to find any sentences of length greater than 40 words or less than 8 words in this dataset. We truncated and padded training sentences to 32 words during data preprocessing, as this would capture the most data while limiting padding.

Figure 2: POS tag Distribution in Training and Test Datasets

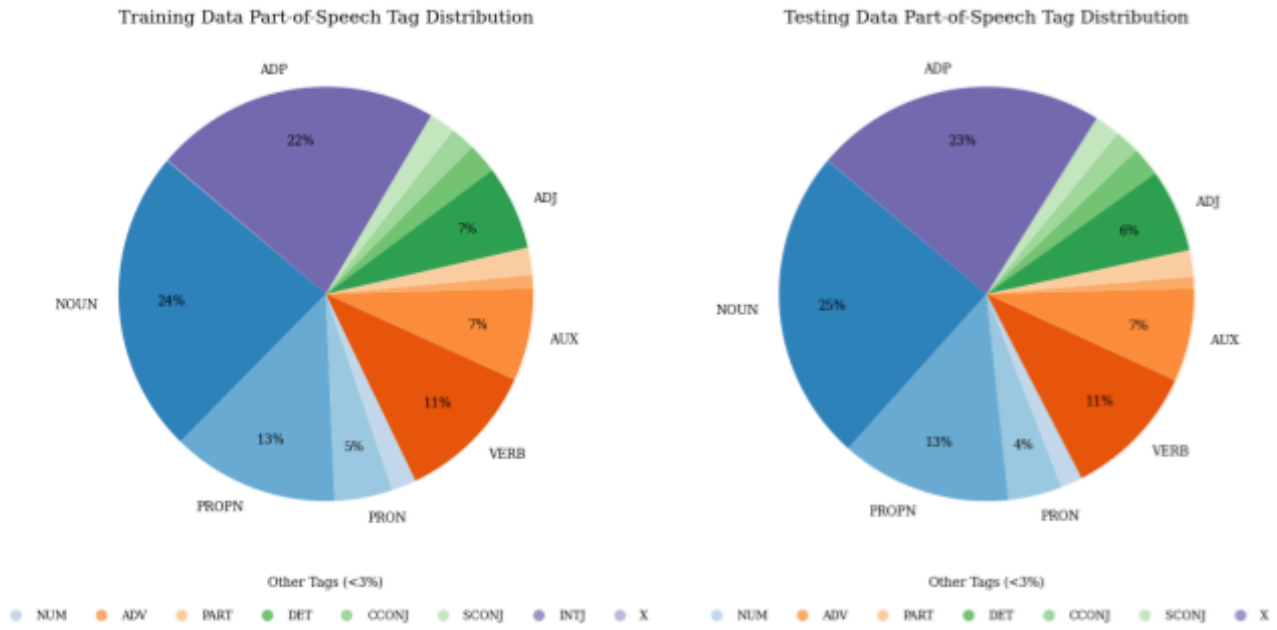
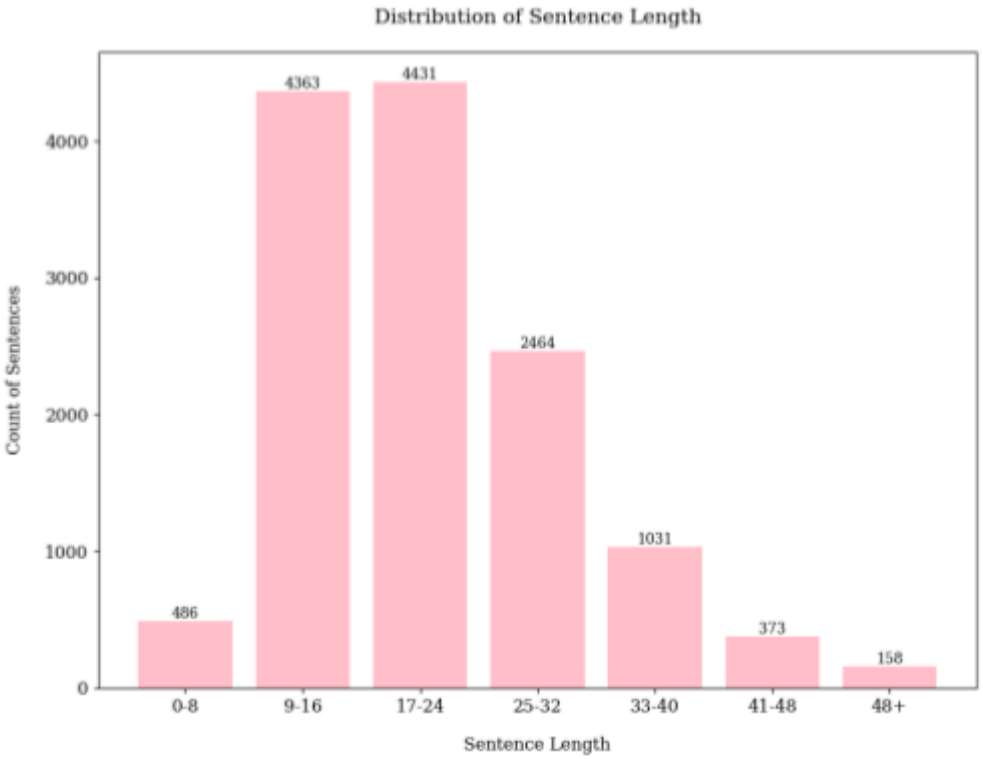


Figure 3: Training Dataset Breakdown by Sentence length (in words)



3.4: Loss Used in Training

Our model's training objective was to minimize a loss that was derived from up to 3 different smaller loss terms over our experiments.

3.4.1: Vocabulary Reconstruction Loss

The first is vocabulary reconstruction loss, which measures the ability of the decoder to reconstruct the input sequence originally passed into the encoder. It is calculated using Pytorch's Cross entropy loss function and compares the decoder's predicted words to the original input. The function measures the performance of the output of the classification model, which will be a vocabulary logits for every word in a given training sentence, higher values meaning more likely predictions. Cross entropy loss applies the softmax function over every value in the distribution, which has the effect of normalizing each value to be probabilities. If the label with the highest probability is not the gold label, loss will increase.

Equation 4 describes how reconstruction loss is calculated. This function iterates over every position *i* in a sequence of length *W* words and multiplies a mask, m_i, by the log probability that this word is the correct input word x. The HMM assumption implemented from Experiment 1 onwards means that we represent this probability as $p_\theta(x_i|z_i)$, whereas in the baseline model this would be $p_\theta(x_i|z_{i\ldots W})$.

$$loss_{recon} = -\sum_{i=1}^{W} m_i \log p_\theta(x_i|z_i) \tag{4}$$

3.4.2: Character Reconstruction Loss

In Experiment 3, we added character reconstruction loss. This loss is calculated in the same way as vocabulary reconstruction loss, using the Cross Entropy Loss function. The main difference between these losses is that character reconstruction loss compares the character-level tokenized input sequences to predicted character-level sequences.

3.4.3: Diversity Loss

The last loss term used across all experiments is the diversity loss. During initial training attempts, we noticed that our unsupervised model was under-using the codebook in its latent space, resulting in every word being assigned to very few latent tags. This phenomenon is called codebook collapse and vastly decreases the generative power of our model. To combat this, we introduced a loss term penalizing the model for low codebook diversity inspired by Bridle et al. (1991). This loss was calculated by measuring how far the actual usage distribution of codebook entries deviates from a uniform distribution. To calculate diversity loss and integrate it into total loss, we go through the following steps.

First, we calculate the entropy of the averaged (or unconditional) probability distribution over all codebook entries for each batch, where K is the number of codebook entries.

$$Entropy_{\text{actual}} = -\sum_{k=1}^{K} \bar{p}(z_k) \log \bar{p}(z_k) \tag{5}$$

Next, we calculate the maximum possible entropy in a sequence:

$$Entropy_{\max} = \log K$$

(6)

Last, we subtract the actual entropy from the maximum entropy and attempt to minimize the diversity value.

$$loss_{diversity} = Entropy_{max} - Entropy_{actual} = \log K + \sum_{k=1}^{K} \bar{p}(z_k) \log \bar{p}(z_k) \quad (7)$$

If the distribution across latent tags is more uniform, the value of $Entropy_{actual}$ will approach $-\ logK$, resulting in a small value for $loss_{diversity}$.

3.4.4: Total Loss

We integrate these three loss terms into our total loss as per Equation 8.

$$loss_{total} = \beta(loss_{VocabRecon}) + (1 - \beta)(loss_{CharRecon}) + \lambda(loss_{diversity}) \quad (8)$$

β and λ are weights used to adjust the importance of the reconstruction and diversity losses, respectively, to the total loss.

3.5 Evaluation Metrics

To evaluate each experiment, we used the following quantitative methods.

1. A graph showing total training and validation loss, along with vocabulary reconstruction and diversity losses for the training dataset.
2. A confusion matrix[1] comparing latent predicted tags to gold tags.
3. A confusion matrix comparing Many-to-One (M-1) predicted tags to gold tags.
4. An overall M-1 accuracy score

Additionally, we qualitatively evaluated the composition of latent tags by creating a table sampling three latent tags and displaying up to 10 most frequent words found in each[2].

[1] To aid interpretation of confusion matrices, we order gold and M-1 part of speech tags on their respective axes based on categorical similarity. For example, nouns, proper nouns, and pronouns are next to each other.
[2]Space permitting. Padding or unknown words found in latent tags are not displayed as they are not informative.

3.5.1: Many-to-One Accuracy

Most works in POS induction use M-1 accuracy for evaluation. We do the same in our work to ensure comparability. M-1 accuracy is calculated by mapping each of the latent tags predicted to its most highly correlated gold POS tag (Johnson, 2007). For T gold tags and Z latent tags, we calculate the M-1 tag of each latent tag, t^*_z, using the following equation, where $n_{z,t}$ is the number of words of gold tag t that appear in latent tag z. This associates each latent tag z with the gold tag t that it has the most words of.

$$t_z^* = \boldsymbol{argmax}(n_{z,t}) \tag{9}$$

The M-1 tag t_z^* is assigned to all members of latent tag z and accuracy is calculated by comparing M-1 tags to gold tags. This is done by dividing the total number of correctly predicted tags by the total number of predictions. The numerator in this case will be the same as the sum of n_{z,t_z^*} for all $z \in Z$, as only the words in latent tag z with the gold tag t that equals t_z^* will count towards the total correct predictions. This is reflected in Equation 10.

$$\boldsymbol{accuracy}_{M-1} = \frac{1}{N}\sum_{z \epsilon Z}(n_{z,t_z^*}) \tag{10}$$

Overall M-1 accuracy will decrease when used to evaluate tags that mix items of different gold tags. Consider a latent tag group that consists of the following gold tag distribution: 40% tag A, 50% tag B, and 10% tag C. The M-1 tag for this group would be gold tag B and all members of the group would assume tag B when calculating accuracy. Thus, the accuracy of gold tags A and C will decrease.

A downside of this metric is that the set of chosen M-1 tags may not include every gold tag. This can happen if some gold tags do not comprise a majority of at least one latent tag. Accuracy for these missing tags will be calculated as 0% as they will never be considered to be predicted.

The value of this metric is that it helps to approximate a lower bound for accuracy in unsupervised classification settings where predictions do not necessarily correspond to actual

labels. Creating confusion matrices to compare M-1 predicted tags to gold tags can provide further insight into the uniformity and makeup of these tags.

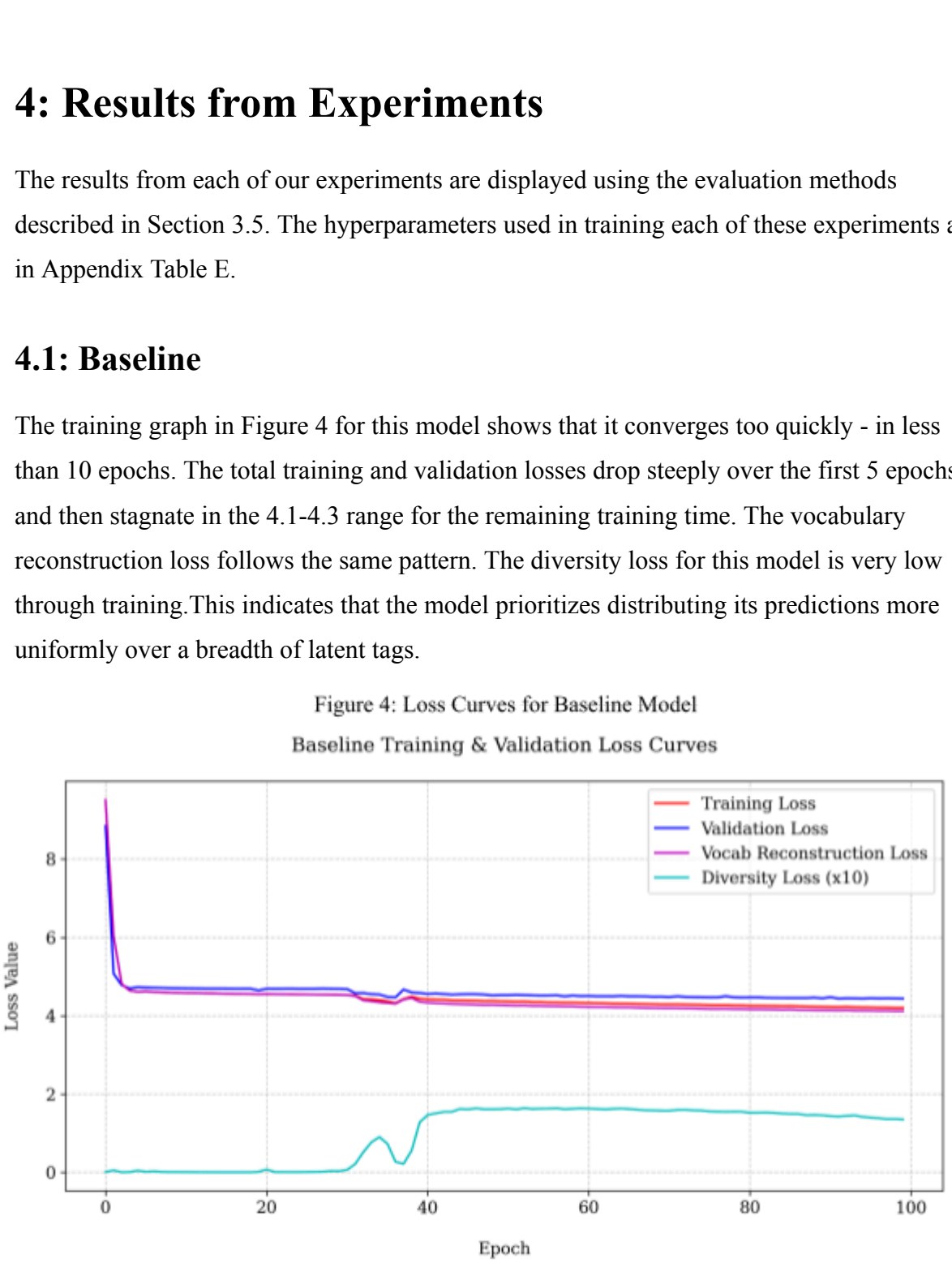
4: Results from Experiments

The results from each of our experiments are displayed using the evaluation methods described in Section 3.5. The hyperparameters used in training each of these experiments are in Appendix Table E.

4.1: Baseline

The training graph in Figure 4 for this model shows that it converges too quickly - in less than 10 epochs. The total training and validation losses drop steeply over the first 5 epochs and then stagnate in the 4.1-4.3 range for the remaining training time. The vocabulary reconstruction loss follows the same pattern. The diversity loss for this model is very low through training.This indicates that the model prioritizes distributing its predictions more uniformly over a breadth of latent tags.

Figure 4: Loss Curves for Baseline Model



The confusion matrix in Figure 5 shows that this model includes nouns and adpositions at a high rate in almost every latent tag. The model does this to a lesser extent for proper nouns, verbs, and auxiliaries. Adverbs seem to be the least distributed gold POS tag. Interestingly, it also shows that no gold POS tag occupies more than half of any latent tag. It does not seem that the model is clustering similar parts of speech in latent tags based on the distributions seen here.

Figure 5:
Baseline Confusion Matrix:
Latent Tags and Gold Labels

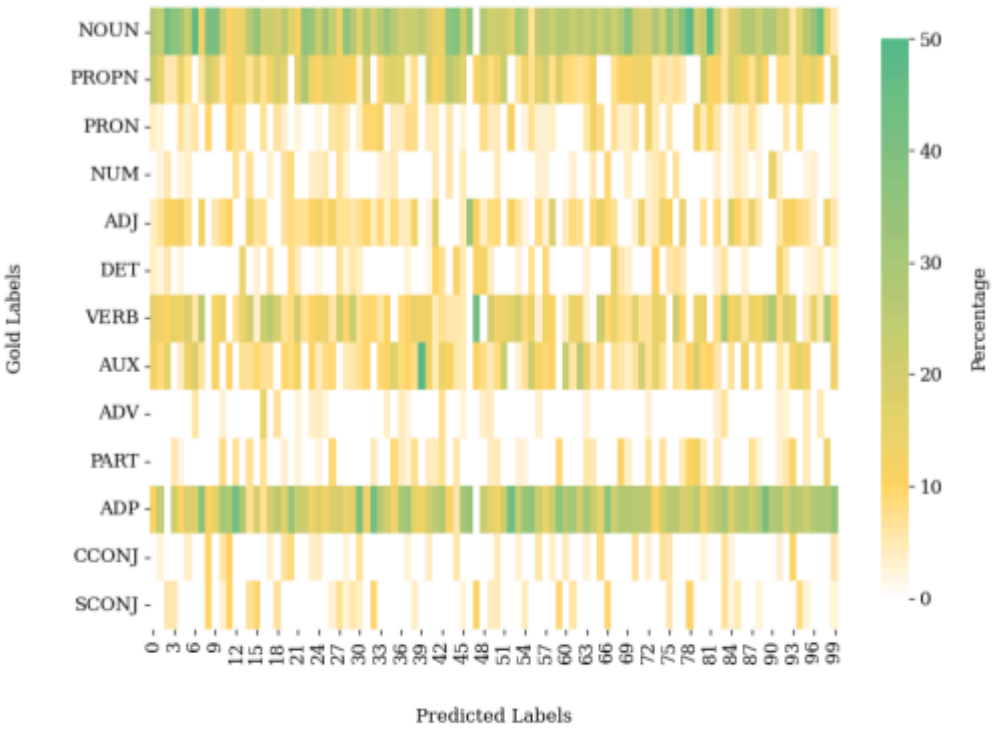
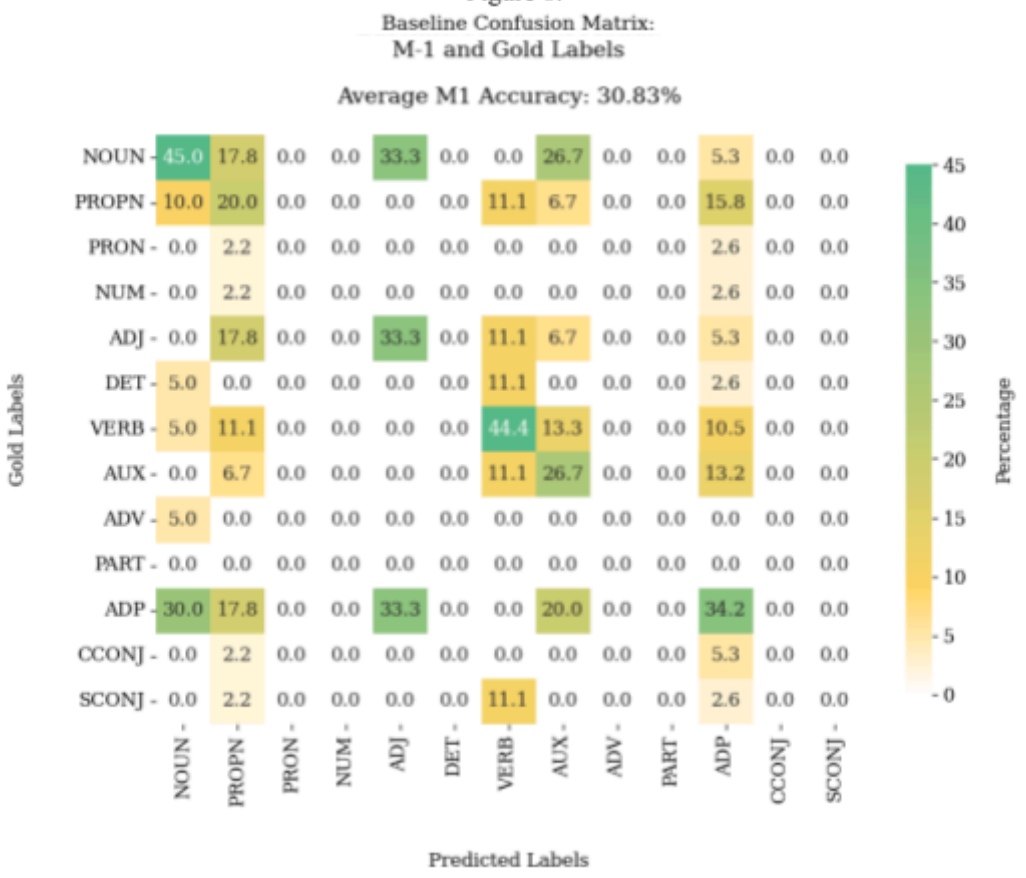


Figure 6 shows that this model chose only 6 of the 13 available gold tags as an M-1 tag for any of the latent tags. 5 of the 9 missing tags are functional classes like coordinating conjunctions, pronouns, and determiners. This model achieves the highest categorical accuracy with nouns and verbs at about 45% each. Other categories like adpositions and adjectives are predicted correctly around 33% of the time. The M-1 tags that this model creates are quite mixed, each containing words that represent at least 3 gold tags. For example, the M-1 adjective group is equally split between nouns, adjectives, and adpositions. The most mixed M-1 group is adpositions, containing words that represent all but 2 of the gold tags. This model's low overall accuracy can be explained by the combination of mixed latent groups and missing M-1 tags.

Figure 6:



The two confusion matrices indicate that latent tags for this model may not effectively group words of similar parts of speech. Table A shows the top unique words assigned to a small sample of tags. This data shows adpositions to be the highest-occurring parts of speech in latent tags even when their M-1 tag is not adposition. The adpositions that occur most do not appear to be grouped by a particular way; in fact, certain groups of adpositions appear in almost every latent tag. For example, the genitive case markers *ke/ki/ko* appear in all three of the below groups, and this pattern is reflected across almost all latent tags. This is in line with the observed mixing of adpositions with almost all gold tags from Figure 6.

Table A: Top Words per Latent Tag for Baseline Model

Tag	M-1 Tag	Top Words								
53	Adposition	HI:	के	की	है	में	को	से	कि	पर	करने
			ke	*ki*	*hai*	*mein*	*ko*	*se*	*ki*	*par*	*karne*
		EN:	of	of	is	in	to	from	that	but	to do
66	Noun	HI:	की	के	को	कहा	है	में	से	और	सभी
			ki	*ke*	*ko*	*kaha*	*hai*	*mein*	*se*	*aur*	*sabhi*
		EN:	of	of	to	said	is	in	from	and	all
85	Proper Noun	HI:	के	में	की	से	है	ने	भी	को	
			ke	*mein*	*ki*	*se*	*hai*	*ne*	*bhi*	*ko*	
		EN:	of	in	of	from	is	by	also	to	

4.2: Experiment 1

Figure 7 shows losses over 100 epochs of training. The loss trends here are quite different from those observed during the baseline model's training, showing a clear exponential decay pattern but with some key differences. This model spends longer initially for learning, taking most of 60 epochs to reach a lower training and validation loss, which starts to plateau around a value of 3. The diversity loss increases steadily over the first 25 epochs and then stabilizes at a value of about 1.6. The model approaches convergence towards the end of training, though it may be worthwhile to observe an additional 25-50 epochs to ensure it.

Figure 7: Loss Curves for Model in Experiment 1

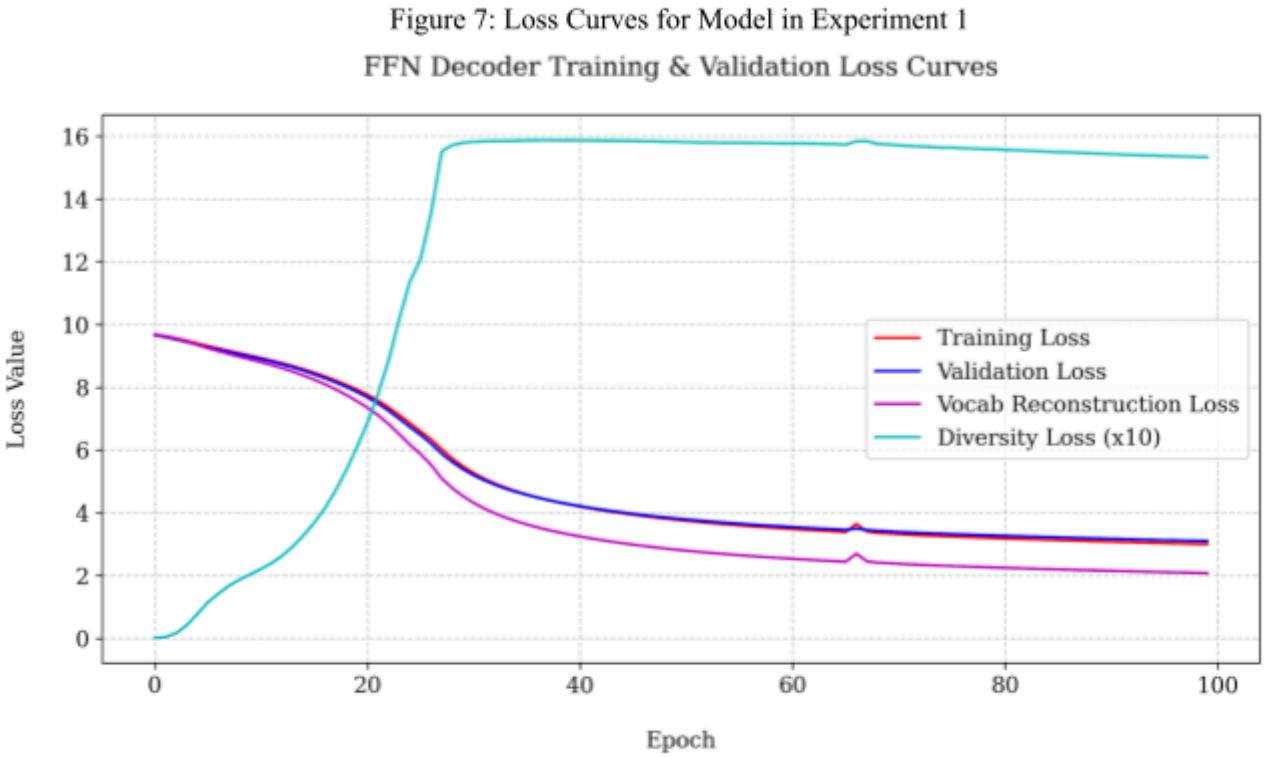


Figure 8 displays the confusion matrix comparing predicted latent tags to gold tags. When compared to that of the baseline model, this matrix is less noisy and the scale shows a higher range of percentages. Nouns tend to comprise more than 50% of most of the latent tags that they occupy. These and proper nouns are distributed among the most latent tags. We observe far less distribution among almost all other classes, regardless of lexical or functional status. For example, determiners occupy about 10 tags here versus close to 40 in the baseline. Auxiliaries, which were distributed among almost every latent tag in the baseline, only appear in about 20 tags in this experiment.

Figure 8:
FFN Decoder Confusion Matrix:
Latent Tags and Gold Labels

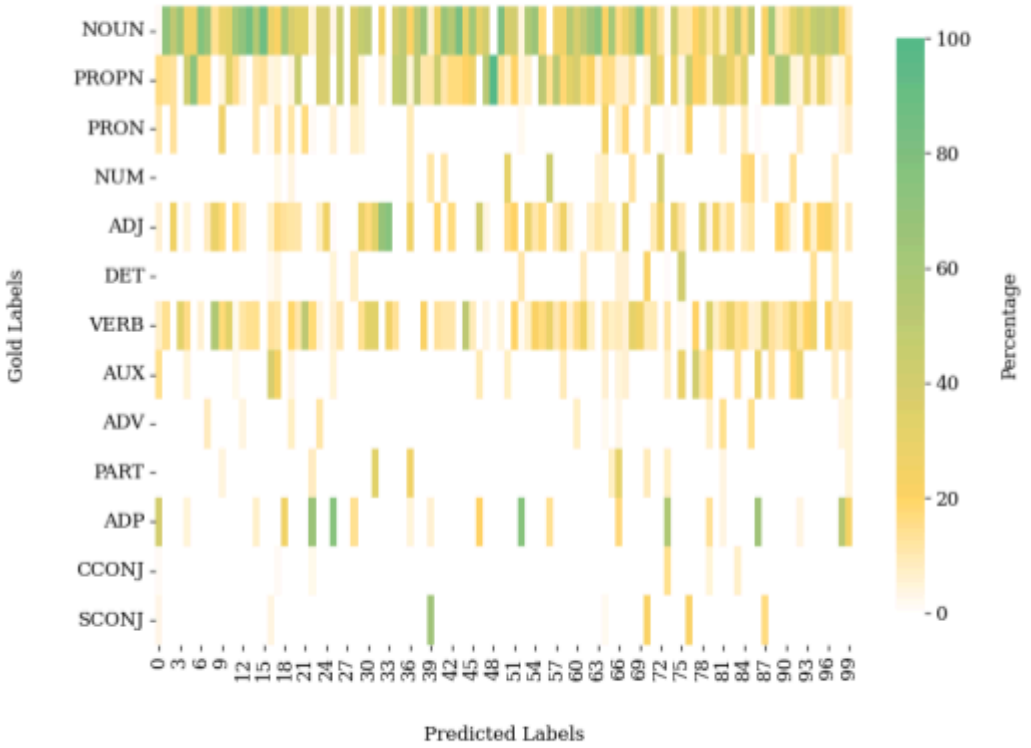


Figure 9 shows stronger clustering patterns and more predicted M-1 tags in comparison to the baseline model. This model utilizes 11 out of the 13 possible M-1 tags, adding determiners, particles, pronouns, numbers, and subordinating conjunctions. For most M-1 tags, gold tags are predicted correctly around 40% of the time or more. The most accurately identified parts of speech are subordinating conjunctions, proper nouns, and nouns, at 64.7%, 57.1%, and 52.9% respectively. Pronouns and adpositions fall short, at less than 25% accuracy for each. Like the baseline model, the adposition group is also the most mixed M-1 group here. It and the particle group contain words that represent all but 4 of the gold tags. Other groups tend to be more clustered when compared to the baseline model.

Table B displays three sampled latent groups and their highest-frequency words. In comparison to the baseline, adpositions no longer dominate these tags. There are at least a few words of the M-1 tag represented in the top 10. These tags still mix words of different tags, but to different extents. Tag 15 is associated with the gold tag Noun by M-1; 8 out of the 9 words in it are nouns and the last word is a pronoun. In tag 71, which is assigned the M-1 tag Determiner, only 2 out of the 8 words shown are determiners. The remainder are distributed among verbs, adverbs, particles, pronouns, and nouns.

Figure 9:

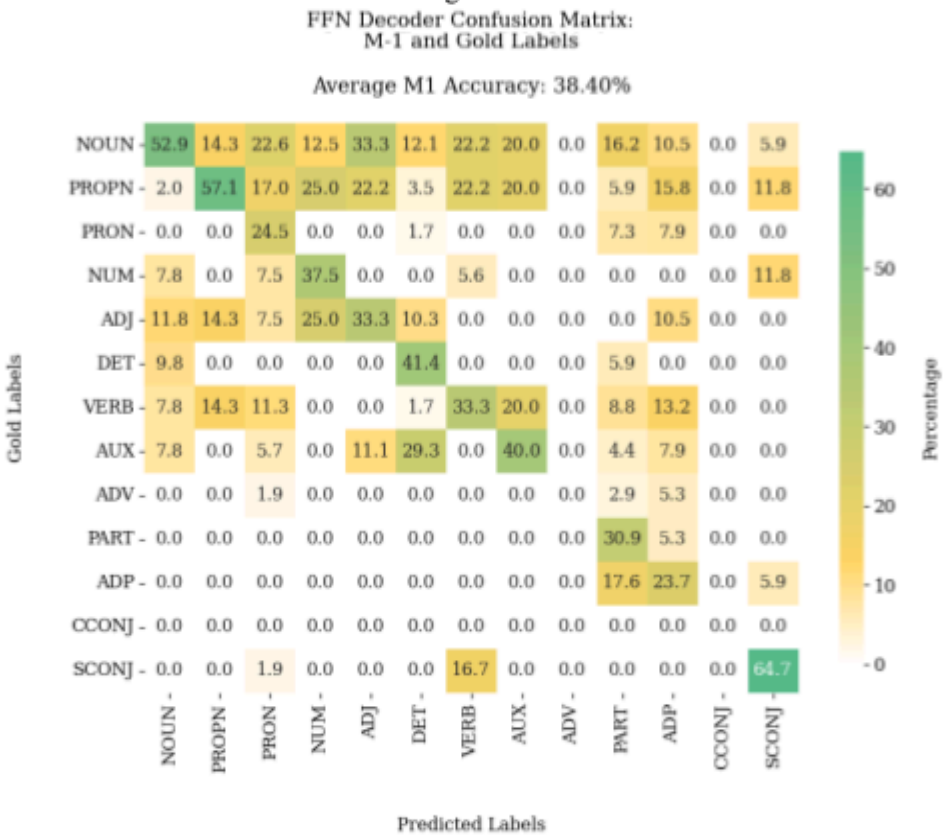


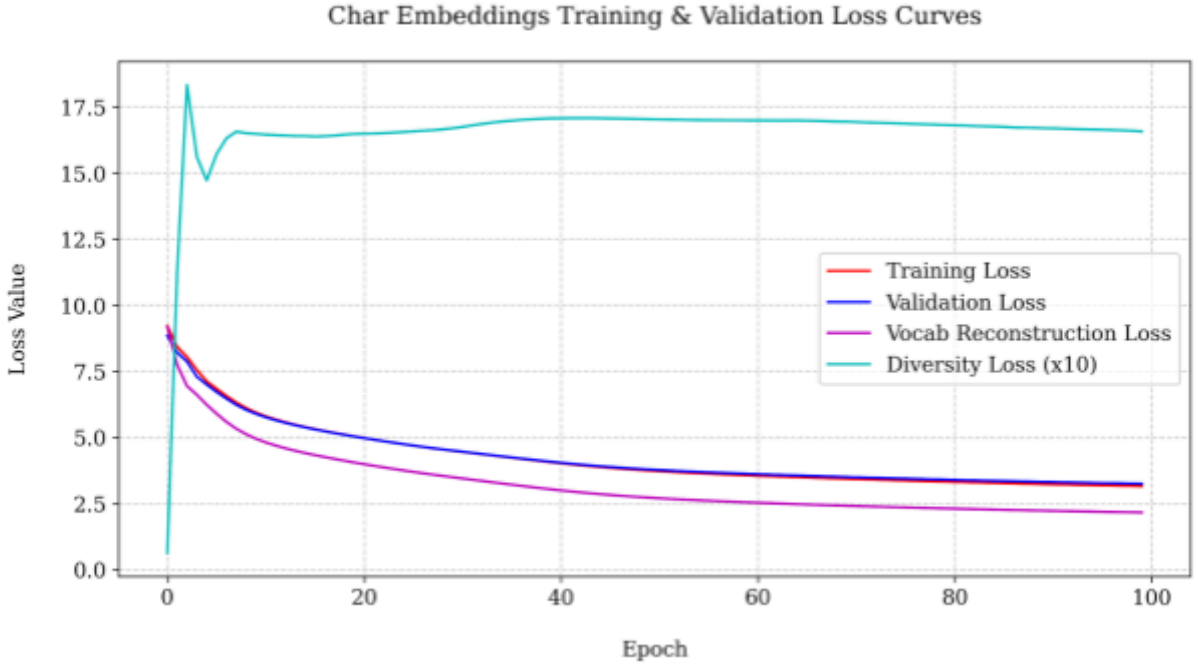
Table B: Top Words per Latent Tag Generated by FFN Decoder Model

Tag	M-1 Tag	Top Words								
15	Noun	HI:	राज्य	इसके	समय	रूप	क्षेत्र	संभावना	समर्थन	वाम	माह
			raajya	*uske*	*same*	*roop*	*kshetr*	*sambhaavna*	*samarthhan*	*vaam*	*ma*
		EN:	state	its	time	form	area	possibility	support left	month	
71	Determiner	HI:	यह	तो	किसी	हुआ	गिरफ्तार	मंगलवार	सहयोग	जल्द	
			yah	*to*	*kisi*	*hua*	*giraftaar*	*mangalvaar*	*sahayog*	*jald*	
		EN:	this	so	any	happened	arrested	Tuesday	collaboration	soon	
87	Adposition	HI:	में	है	ने	किया	इसके	राज्य	समय	पुलिस	
			mein	*hai*	*ne*	*kiya*	*iske*	*raajya*	*samay*	*polis*	
		EN:	in	is	by	did	its	state	time	police	

4.3: Experiment 2

The training loss graph for this model, shown in Figure 10, looks similar to that of the HMM assumption model (Figure 7), showing a similar smooth exponential decay curve. Both the training and validation losses decrease consistently for the first 30 epochs and smooth out over the rest of training. In contrast to the model in Experiment 1, the diversity loss for this model increases sharply at the beginning of training and stays relatively stable at a value of around 1.7 for the remainder of training. This model converges by the end of training at a loss of about 2.5, slightly lower than that of the HMM assumption model.

Figure 10: Loss Curves for Model in Experiment 2



The confusion matrix (Figure 11) comparing predicted latent tags to gold POS tags looks similar to that of Experiment 1. Adjectives, nouns, proper nouns, and verbs are similarly distributed over most latent tags. Nouns continue to compose at least half of most of the latent tags they are found in. Most other parts of speech are clustered similarly, although there is noticeably more clustering of pronouns.

Figure 11:
Char Embeddings Confusion Matrix:
Latent Tags and Gold Labels

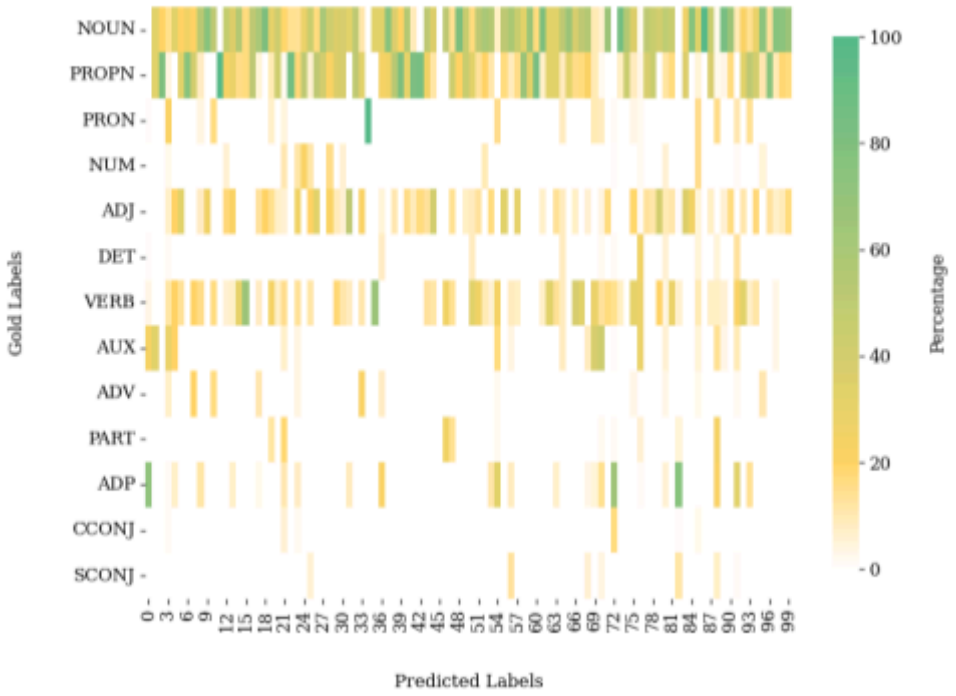


Figure 12 shows that 8 out of 13 possible M-1 predicted tags are used, with the removal of number, subordinating conjunction, and determiner groups. The decrease in POS tags used likely contributes to its lower overall M-1 accuracy compared to Experiment 1. This matrix shows more confidence in clustering, with many M-1 categories only including a few gold POS tags. Particles, auxiliaries, and adpositions continue to be mixed in their gold tag compositions. Notably, accuracy for pronouns jumped from about 25% in the previous model to 100% in this model. This model still categorizes adjectives, nouns, proper nouns, and verbs across the most latent tags.

The top words for a selection of this model's latent tags can be found in Table C. Like the data seen in Experiment 1, these are better than the baseline, but remain mixed. In tag 8, 3 of the top 7 words are proper nouns and match their assigned M-1 tag. 2 more are nouns, and the remaining two are verbs. Tag 32 contains 4 adjectives out of the top 6 words contained. The remaining 2 are one noun and one verb.

Figure 12:

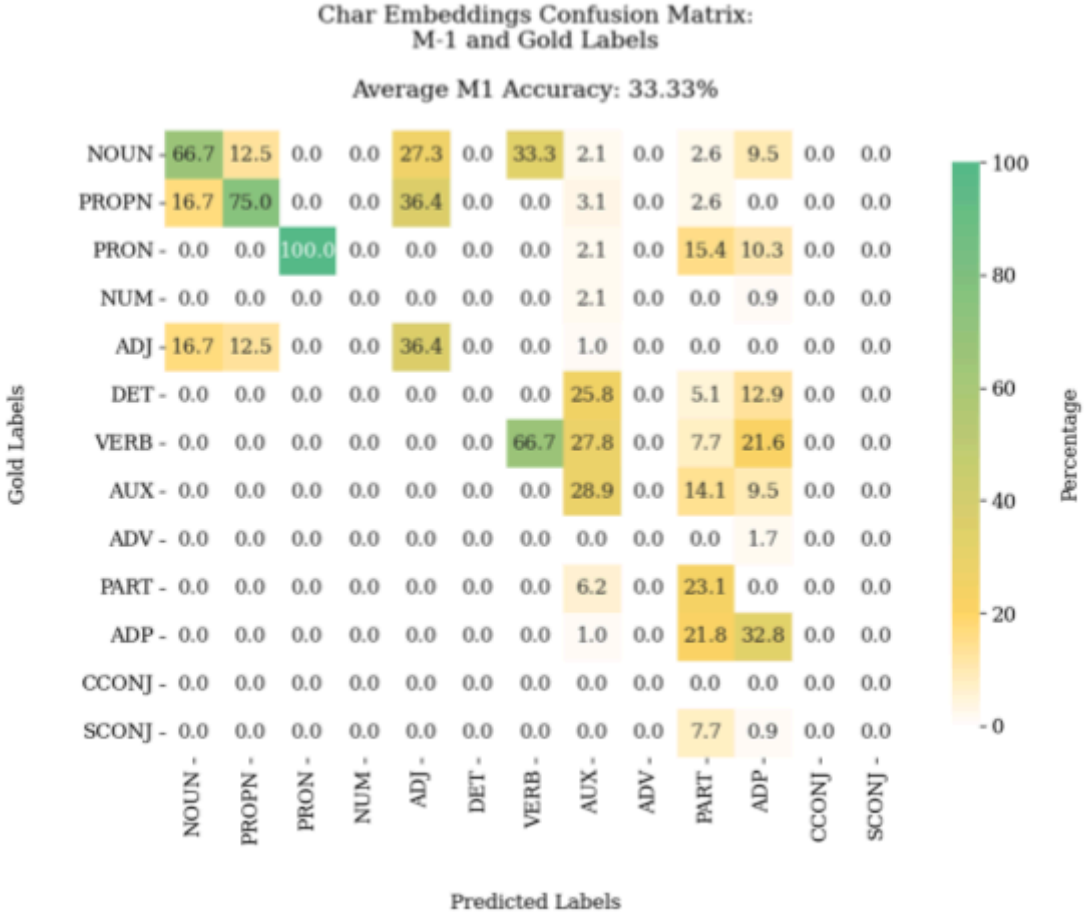


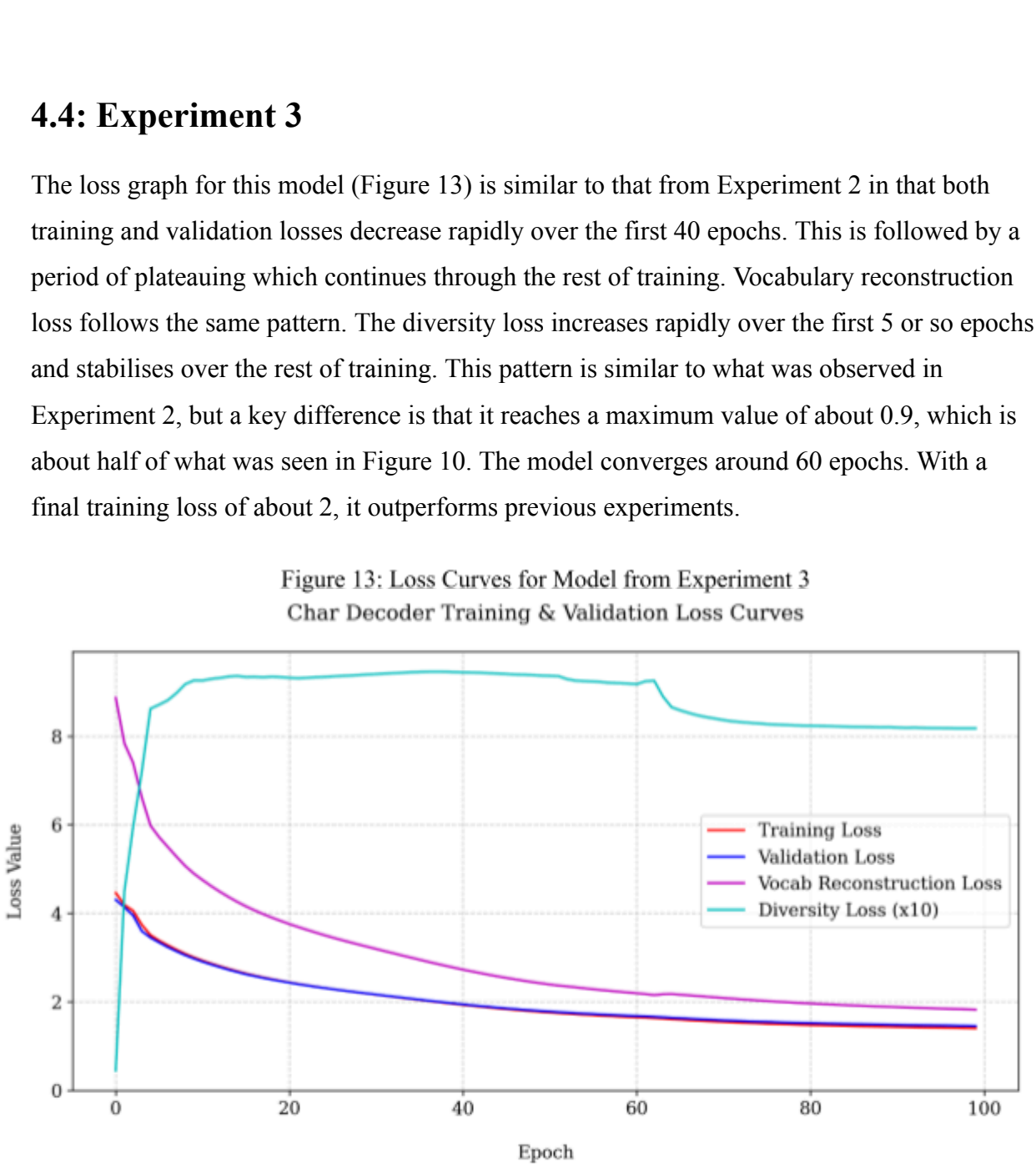
Table C: Top Words per Latent Tag Generated by Character Embedding Model

Tag	M-1 Tag	Top Words						
8	Proper Noun	HI:	अधिकारों	गिरने	पहलगाम	नवरात्रि	सक्सेना	दीजिए	दिमाग
			adhikaron	*girne*	*pahalgam*	*navratri*	*saxena*	*dijiye*	*dimaag*
		EN:	rights	to fall	Pahalgam	Navratri	Saxena	give	brain
32	Adjective	HI:	नेशनल	बढ़ोतरी	प्रसिद्ध	घंटों	सूखा	शांतिपूर्ण	
			neshunal	*badotari*	*prasiddh*	*ghanton*	*sukha*	*shantipoorn*	
		EN:	national	increase	famous	hours	dry	peaceful	
54	Noun	HI:	बैठक	अध्यक्ष	क्षेत्र	पासवान	चलते	मिली	
			Baithak	*adhyaksh*	*kshetr*	*paswan*	*chalate*	*mili*	
		EN:	meeting	chairman	area	Paswan	go	found	

4.4: Experiment 3

The loss graph for this model (Figure 13) is similar to that from Experiment 2 in that both training and validation losses decrease rapidly over the first 40 epochs. This is followed by a period of plateauing which continues through the rest of training. Vocabulary reconstruction loss follows the same pattern. The diversity loss increases rapidly over the first 5 or so epochs and stabilises over the rest of training. This pattern is similar to what was observed in Experiment 2, but a key difference is that it reaches a maximum value of about 0.9, which is about half of what was seen in Figure 10. The model converges around 60 epochs. With a final training loss of about 2, it outperforms previous experiments.

Figure 13: Loss Curves for Model from Experiment 3



The latent tag to gold tag comparison shown in Figure 14 shows patterns that are similar to those in Experiment 2. Adjectives, nouns, proper nouns, and verbs are once again the most distributed among latent tags. One difference is that this model seems to be slightly worse at clustering certain tags. For example, adpositions exist in about 25 tags here, while in Experiment 2 they were distributed among about 15 tags. Other functional classes either follow a similar pattern or show relatively little change, as in both conjunctions.

The character decoder model uses 10 out of 13 possible M-1 tags (Figure 15), an improvement from Experiment 2. This model adds back determiner, adverb, and both conjunction types while removing adjective and particle M-1 tags. This is the first of our experiments to classify at least one latent tag group with the M-1 tag of adverb. Most tags achieve an accuracy of 30% or higher, with several crossing the 50% threshold. The M-1 verb group has the worst accuracy and purity, mixing 8 gold tags together. Adpositions tend to be mixed with verbs at a high rate. In comparison to Experiment 2, this model decreases accuracy in several lexical classes. Even so, this model achieves the highest overall M-1 accuracy at almost 50% out of all experiments. This can be explained by its usage of more M-1 tags and increased purity of most tags.

Figure 14:
Char Decoder Confusion Matrix:
Latent Tags and Gold Labels

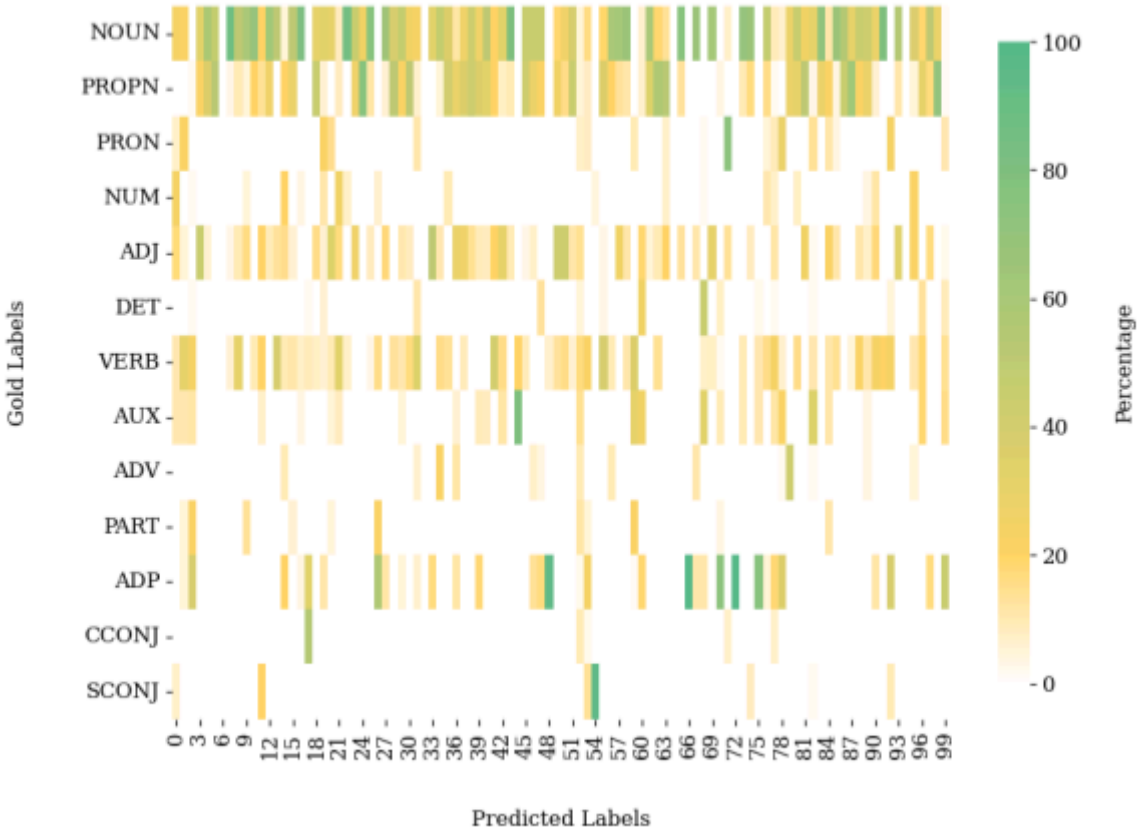
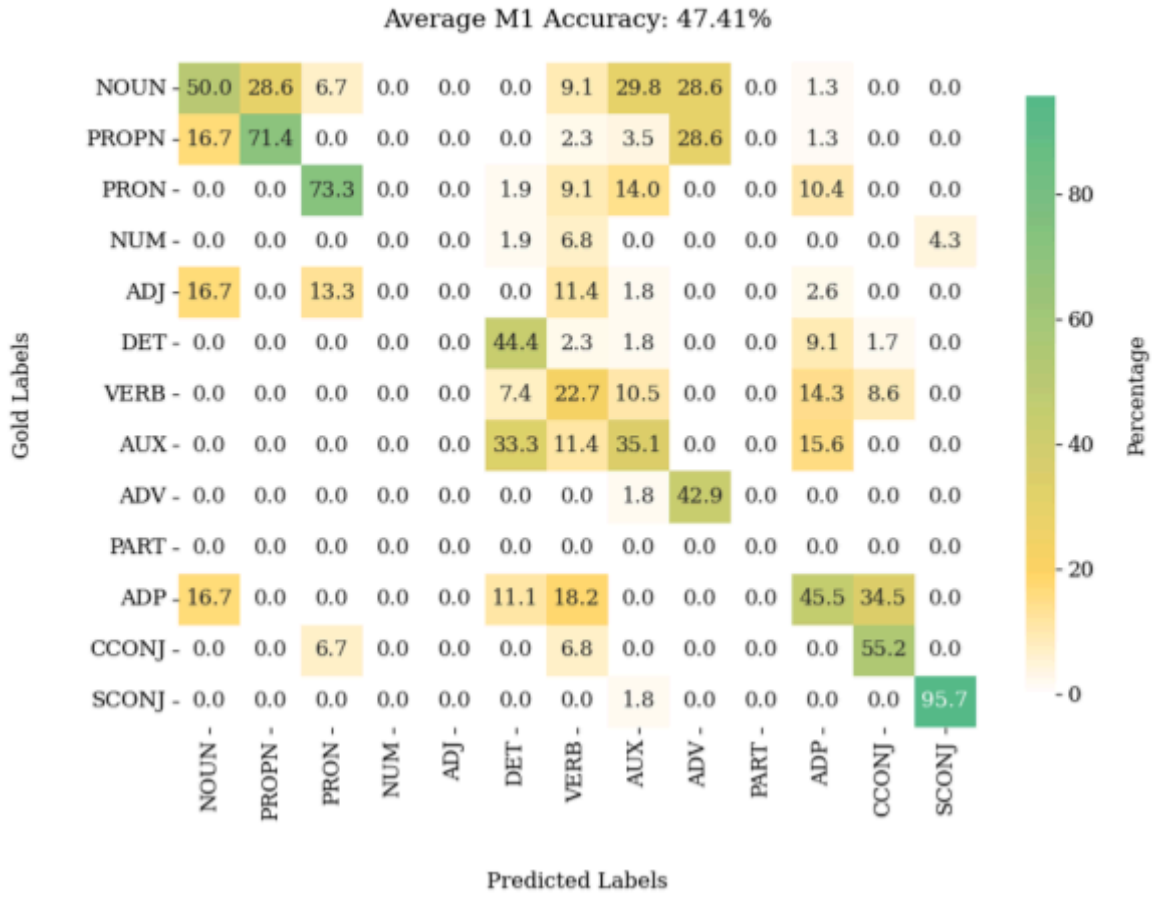


Figure 15:
Char Decoder Confusion Matrix:
M-1 and Gold Labels



The sample data for this model (Table D) shows continued mixing of gold tags to varying degrees. Tag 27 only has one adposition, its M-1 assigned tag, out of its top 8 words. The others are distributed between numbers, particles, verbs, and pronouns. Tag 42, with the M-1 label of verb, contains words like बंद (“close”) that can function as a verb or noun depending on context. This means that it contains at most 4 verbs out of its top 10 words. Tag 96 is the most pure tag in this sample, with 6 out of its 8 tags matching its assigned M-1 tag. It is also the only one that shows morphological purpose; including words like *vichar*, *chaar*, and *charcha.*

Table D: Top Words per Latent Tag Generated by Character Decoder Model

Tag	M-1 Tag	Top Words										
27	Ad-position	HI:	की	भी	दो	करने	देने	ले	देकर	कही		
			ki	*bhi*	*do*	*karne*	*dene*	*le*	*dekar*	*kahin*		
		EN:	of	too	two	to	to give	take	by giving	somewhere		
53	Verb	HI:	गए	व	हो	तथा	वजह	करेंगे	हमले	आगे	इसे	बंद
			gaye	*va*	*ho*	*tatha*	*vajah*	*karenge*	*hamle*	*aage*	*ise*	*band*
		EN:	went	and	be	also	reason	do	attacks	ahead	this	close
96	Noun	HI:	वर्ष	विचार	चार	सीमा	चर्चा	मिली	बड़ी	मिल		
			varsh	*vichar*	*chaar*	*seem*	*charcha*	*mili*	*bade*	*milab*		
		EN:	year	idea	four	limit	discussion	found	big	mill		

5: Discussion of Results

5.1: Summary of Key Findings

We evaluated each model by comparing training and validation loss, overall and tag-specific M-1 accuracy, and clustering of latent tags. Our results show each experiment to have its strengths and weaknesses, but it appears that the most effective model was that of Experiment 3. The best model is confident in its predictions for functional classes and slightly less so for lexical classes, although even in the latter it makes accurate predictions often. The model in Experiment 2 performs similarly. It predicts lexical classes slightly better and is worse at predicting functional classes, but its overall M-1 accuracy is worse.

In general, the models in our experiments showed incremental improvements with each additional component. This aligns with our initial hypothesis that the use of certain assumptions and information would assist our VQ-VAE model in its learning goal.
In Experiment 1, using HMM assumption *b* for reconstruction seemed to help the model by reducing contextual noise that exists in a language with free word order. This change made the biggest impact, causing training losses to decrease drastically and clearer clustering of latent tags. Combining character embeddings with BERT embeddings in Experiment 2 also had a positive effect on training loss, and it is here that we observed more confident clustering in functional classes. Finally, adding a character-level reconstruction loss term resulted in both the lowest training and validation losses. This also decreased clustering in lexical classes but maintained reasonable performance across most gold tags.

Performance improved across each experiment, but not as much as we had hoped. Only a subset of all 13 available POS tags were identified as M-1 tags in each model. The best model did not associate any latent tags with numbers, adjectives, or particles. Ideally, we would have created a model that would assign at least one latent tag to each gold tag through M-1 association. It is possible that our models cluster incorrectly due to the way that they are trained. In each experiment, diversity loss was much higher than in the baseline through most of training. This indicates that the model found unequal usage of latent tags to be beneficial. In Experiments 2 and 3, this rapid increase of diversity loss happened early in training. The

models may have begun clustering words too early, before they took the time to learn patterns of similarity between them.

A factor that could explain the limited efficacy of character-based information is that morphological bits like affixes and adpositions can only be used to differentiate certain parts of speech. In Hindi, both adjectives and nouns inflect in the same way for gender, case, and number, and affixes cannot be used to differentiate them (Butt, 2001). Verbs can be identified by tense affixes, but these also appear in auxiliaries. This may explain the frequent mixing between these categories, which contributes to lower M-1 accuracy.

Given that previous studies like that of Zhou et al. (2022) achieved success in other languages using Bi-LSTMs, we expected that our baseline model's results would be better. Our guess is that linguistic features of Hindi may account for the observed poor performance. The relatively free word order of the language combined with the phenomenon of argument dropping in Hindi creates randomness that our model needed to accommodate. Japanese and Korean are the closest languages in this regard that have been evaluated with these methods, with flexible word order aside from being verb-final languages. The results from Zhou et al. (2022) show that evaluation on Korean is consistently the worst among the 10 languages they test on; perhaps some shared linguistic between Korean and Hindi could be the reason.

5.2: Situating Results in the Broader Context

The overall M-1 accuracy of our best model is 47.41%. We calculated it in the same way as Stratos (2019) as this appears to be standard across POS induction work. We compare this to the two most similar examples of POS induction to ours, which are Zhou et al. (2022) and Gupta et al. (2022). Both of these research groups achieved M-1 scores of on average 70-75% over a standard set of 10 languages that use the same 12 universal POS tags (McDonald et al., 2013). The language set primarily consists of European languages aside from Japanese, Korean, and Indonesian.

In comparison to these works, our M-1 accuracy is substantially lower. However, for the three most frequent parts of speech in our training data, we achieve accuracy of about 50% or higher. Nouns and pronouns each achieve 50% and 72% accuracy respectively. Adpositions

achieve 45% accuracy. Our best model, in its un-optimized state, achieves competitive accuracy for nouns.

Because our work is the first example of unsupervised POS tagging for Hindi using neural methods that we are aware of, comparing our M-1 accuracy to these papers does not prove whether our methods are more or less effective than theirs. Our experiments are tailored to the linguistic context of Hindi, while their methods aim to be effective multilingually.

5.3: Limitations, Improvements, and Extensions on this Work

5.3.1: Limitations of this Study

The datasets available to us for this work posed limitations as a result of the domains included. Hindi annotated treebanks were desired due to their enablement of useful evaluation, and the UD Hindi corpus was the only accessible human-annotated corpus available. As mentioned previously, this corpus is restricted to the news domain, which tends to utilize more conventional structural patterns. Therefore, our models may not generalize well for Hindi's relatively free word orders.

As students, our access to computational resources was limited. We trained our models using Google Colab Pro+ and ran most of the training on Nvidia A100 GPUs. Colab Pro+ provided access to one GPU at a time which limited our ability to tune hyperparameters. We would have liked to train using a larger range of hyperparameters to more comprehensively assess how diversity and reconstruction loss weights, Gumbel temperature, and learning rate impact training. Even so, the results shown in this work are reasonable; we believe they could still improve with some tuning.

5.3.2: Model Refinement

Due to time and resource constraints, we prioritized creating models that were built correctly, and optimization of our models was very limited.

Dedicated testing of hyperparameters would be helpful in optimizing the model's results. Due to time and resource constraints, we used hyperparameter values that were reasonable with

respect to general guidance. Additionally, optimization of the learning rate could help our models learn better. For example, changing this could assist our baseline model, which converges very quickly given the current set of hyperparameters.

Our model results could be improved further through simulated annealing of certain hyperparameters; for example, diversity loss weight. We know that diversity loss weight affects the model's tendency to use more of the latent tags, with higher values encouraging more distributed use and lower values leading to more clustering. Simulated annealing would allow us to gradually decrease the diversity loss weight over the training period. This would encourage the model to prioritize using more of its allocated latent tags initially and gradually cluster tags over time as it grows more confident in the patterns it is learning. The same kind of annealing applied to the Gumbel temperature used in Gumbel-Softmax could achieve a similar effect.

Lastly, we know that the initialization of unsupervised model weights can affect how well the model learns. Zhou et al. (2022) make attempts at initializing their model intelligently, and it would be useful to do the same for our model and observe whether this improves results.

5.3.3: Extensions of this Work

The results of this work provide an interesting baseline. We pose the following extensions that would make them robust and better-situated in the broader research context.

For Hindi, we were lucky to have access to MuRIL, a BERT model that was pre-trained on Indian languages. Not all languages have the resources to have something like this. It would be interesting to observe performance in Hindi without the use of MuRIL for contextual embeddings. This would provide insight into how these methods generalize for low-resource settings. In a similar vein, it would be helpful to train our model using corpora for languages with similar features to Hindi. This would inform whether our methods are effective for POS induction in the linguistic contexts of morphological richness and free word order.

To situate our results better, we would like to compare our model results to those of the masked model POS induction created by Zhou et al. (2022), as this is the work that most

closely parallels ours. It would be beneficial to use the training data from our work to train their model or use their training data for our models and compare results. The comparison could provide insight into the benefits of masking and the impacts of language context on POS induction.

Lastly, it could be interesting to run these experiments individually and in a different order, in order to understand the impacts of each component as well as their effects on each other.

5.3.4: Future Research

In this work, we investigated unsupervised methods of POS tagging for Hindi. It would be interesting to investigate the efficacy of semi-supervised learning methods on the same problem, as this would provide the advantage of using classes provided in the dataset for training. The various models we tested in this work could be trained in a semi-supervised way on both UD data and other larger Hindi datasets. This would provide insight into the amount of data needed for POS induction work, and could expose the model on a broader range of domains.

It could also be interesting to apply curriculum learning to this work and observe how it impacts learning. For example, this could be done by giving the model proxies for complexity (perhaps sequences of increasing length or arguments) over the span of training. This would provide an avenue to incrementally observe what kinds of syntactic patterns a model can learn over the span of its curriculum.

Lastly, a larger undertaking would be to attempt to compare computational POS induction to human acquisition of syntactic categories.This would require much more work, especially in understanding the cognitive processes behind human language acquisition. We found preliminary work which lays out the current state of POS induction by outlining different computational approaches and evaluation methods (Dickson, 2023). It also identifies certain challenges that these approaches face. Future work could build upon this by more directly comparing different methods of POS induction to observed patterns of language acquisition.

5.4: Significance & Contributions

This work provides preliminary insight into the effectiveness of unsupervised methods for predicting syntactic structure in Hindi. Our work is the first of our knowledge to use unsupervised methods of POS tagging for South Asian languages.

Computationally, we establish new baselines for POS induction for Hindi in an unsupervised setting. We also contribute to existing VQ-VAE work by implementing a novel usage of the Gumbel Softmax procedure for the discretization of the latent space.

Much of the previous work we surveyed makes the assumption in reconstruction that any given word is dependent on the states that occur around it. We challenge this assumption and find that input reconstruction for Hindi POS induction improves when the reconstructed words are only conditioned on the latent tag. This shows that Markov assumption *b* can be more effective in some cases than using a Bi-LSTM for contextual use.

In sum, we show that it is possible to learn syntactic categories using unsupervised methods, even when using small datasets and in languages where syntactic structures are less predictable.

6: Conclusion

This research aimed to investigate the ability of current neural methods to achieve effective POS induction in Hindi. We answered the two research questions highlighted at the end of section 1. For the first, the results of our work indicate that recent methods of POS induction, though effective for a range of languages, are not particularly effective for Hindi. For the second, accounting for Hindi's free word order had a larger effect on model performance than accounting for its rich morphology did.

Prior to running experiments, we established a base architecture using VQ-VAEs and Gumbel-Softmax, which allowed us to harness the strengths of VAEs without sacrificing the discrete nature of latent tags. Our experiments were then ordered to assess model-intrinsic components prior to adding components that enhance the information provided to the model. In this way, we ensured that the model's base architecture was effective prior to making further changes.

We expected each experiment to improve upon the last, which happened with the exception of Experiment 2, whose overall accuracy decreased. We also expected to observe morphologically-driven latent tags, especially in Experiment 3, but this was not obvious to us upon qualitative evaluation. The results we observed raised the following new questions:

1. Does character-level information actually help the model, or does it also create additional noise?
2. How important are BERT embeddings in our work?
3. How much impact does initialization have on the results of our models?

Our work can be extended and built upon in a few ways. Semi-supervision, curriculum learning, and datasets could be tested to understand the impact of adjusting training approaches. To better understand the implications of the results, future studies could also use our methods of POS induction for languages that have similar linguistic features as Hindi. This would tell us whether these methods generalize cross-linguistically, either to languages like Hindi or across a wider range. More broadly, there also exists an opportunity to compare POS induction abilities to human acquisition of syntactic categories.

The contributions of this work to the field are the following.

1. We broaden the application of POS induction to South Asian languages, beginning with Hindi, and show that current approaches may not be effective for them.
2. We challenge the assumption in NLP that full sequence context is needed for predicting individual words.
3. We provide a baseline for POS induction performance on Hindi after incorporating architectural components that account for language-specific features.
4. We add to the body of work on VAEs by applying the Gumbel-Softmax procedure in a novel way to discretize a latent space for POS induction.

This work is some of the first dedicated to POS induction for a non-European language. We showed that using linguistic features to inform model design can result in something that is functional and works. We hope that this work inspires further investigation into unsupervised NLP tasks in a more diverse set of languages.

Acknowledgments

The ideas and execution behind this work are all that of myself and my collaborator, Chini Lahoti. We developed the research question, experiments, and evaluation methods ourselves. Our supervisor, Dr. Miloš Stanojević, provided essential feedback on our ideas, methods, and particularly tricky issues in our code.

As this work was collaborative with another student, the details of my contributions as per the Contribution Statement Guidelines.

- Conceptualisation - Student-led
- Experimental methodology - Student-led
- Implementation of experiment - Student-led
- Data collection - Student-led
- Statistical analyses - Student-led
- Visualisation - Student-led

Any errors are my own.

7: References

Bhat, R. A., Bhatt, R., Farudi, A., Klassen, P., Narasimhan, B., Palmer, M., ... & Xia, F. (2017). The Hindi/Urdu Treebank Project. In *Handbook of linguistic annotation* (pp. 659-697). Dordrecht: Springer Netherlands.

Bridle, J., Heading, A., & Mackay, D. (1991). Unsupervised Classifiers, Mutual Information and "Phantom Targets." *Advances in Neural Information Processing Systems 4 (NIPS 1991)*, 1096–1101.

Butt, M. (2001, August). Case, agreement, pronoun incorporation and pro-drop in South Asian languages. In *a workshop on the role of agreement in argument structure, at Utrecht University.*

Christodoulopoulos, C., Goldwater, S., & Steedman, M. (2010, October). Two Decades of Unsupervised POS induction: How far have we come? *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 575–584).

Dickson, N. (2023). Review of Unsupervised POS Tagging and Its Implications on Language Acquisition. *arXiv preprint arXiv:2312.10169.*

Gumbel, E. J. (1954). *Statistical theory of extreme values and some practical applications: a series of lectures* (Vol. 33). US Government Printing Office.

Goldwater, S., & Griffiths, T. (2007, June). A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th annual meeting of the association of computational linguistics* (pp. 744-751). Association for Computational Linguistics.

Gupta, V., Shi, H., Gimpel, K., & Sachan, M. (2022, June). Deep Clustering of Text Representations for Supervision-Free Probing of Syntax. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 36, No. 10, pp. 10720-10728).

Jang, E., Gu, S., & Poole, B. (2016). Categorical reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144.*

Johnson, M. (2007, June). Why doesn't EM find good HMM POS-taggers?. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (pp. 296-305).

Khanuja, S., Bansal, D., Mehtani, S., Khosla, S., Dey, A., Gopalan, B., & Talukdar, P. (2021). Muril: Multilingual representations for Indian languages. *arXiv preprint arXiv:2103.10730.*

Kingma, D. P., & Welling, M. (2013, December). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114.*

Kingma, D. P., Salimans, T., & Welling, M. (2015). Variational dropout and the local reparameterization trick. *Advances in neural information processing systems, 28*.

Maddison, C. J., Mnih, A., & Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712.*

McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., & Lee, J. (2013, August). Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 92-97).

Merialdo, B. (1994). Tagging English text with a probabilistic model. *Computational linguistics,* 20(2), 155-171.

Plank, B., Søgaard, A., & Goldberg, Y. (2016). Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss (pp. 412–418). Association for Computational Linguistics.

Qi, P., Dozat, T., Zhang, Y., & Manning, C. D. (2018). Universal Dependency Parsing from Scratch. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 160–170. https://doi.org/10.18653/v1/K18-2016

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. (2020). Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *ACL Anthology* (pp. 101–108). Association for Computational Linguistics.

Singh, S., Gupta, K., Shrivastava, M., & Bhattacharyya, P. (2006, July). Morphological Richness Offsets Resource Demand-Experiences in Constructing a POS Tagger for Hindi. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 779–786.

Stratos, K. (2019). Mutual information maximization for simple and accurate part-of-speech induction. *arXiv preprint arXiv:1804.07849*.

Stratos, K. & Collins, M. (2015, June). Simple Semi-Supervised POS Tagging. *In Proceedings of the 1st workshop on vector space modeling for natural language processing* (pp. 79–87). Association for Computational Linguistics.

Van Den Oord, A., & Vinyals, O. (2017). Neural discrete representation learning. *Advances in neural information processing systems, 30.*

Zhou, X., Zhang, S., & Bansal, M. (2022). Masked Part-Of-Speech Model: Does Modeling Long Context Help Unsupervised POS-tagging? *arXiv preprint arXiv:2206.14969.*

8: Appendix

8.1: Hyperparameter Settings for Training

Table E: List of hyperparameters used in training along with their values

Hyperparameter	Value
Max sequence length in words	32
Max sequence length in tokens	54
Max word length in characters	10
Character embedding dimension	64
Word embedding dimension	128
Number of latent tags	100
Codebook vector dimension	50
Embedding (BERT) dimension	768
Decoder hidden dimension	256
Number of decoder hidden layers	2
Learning rate	6e-5
Training epochs	100
Batch size	256
Gumbel temperature	1
Diversity weight	0.6
Vocabulary reconstruction loss weight	0.5 in Experiment 3, otherwise 1
Character reconstruction loss weight	0.5 in Experiment 3, otherwise 0