Unsupervised Learning of Morphology with Multiple Codebook VQ-VAE

Anonymous ACL submission

Abstract

This paper presents an interpretable unsupervised morphological learning model, showing comparable performance to supervised models in learning complex morphological rules as evidenced by its application to the problem of morphological inflection within the SIGMORPHON Shared Tasks. The significance of our unsupervised approach lies in its alignment with how humans naturally acquire rules from raw data without supervision. To achieve this, we construct a model with multiple codebooks of VQ-VAE employing continuous and discrete latent variables during word generation. We evaluate the model's performance under high and low-resource scenarios, and use probing techniques to examine encoded information in latent representations. We also evaluate its generalization capabilities by testing unseen suffixation scenarios within the SIGMORPHON-UniMorph 2022 Shared Task 0. Our results demonstrate our model's ability to distinguish word structures into lemmas and suffixes, with each codebook specialized for different morphological features, contributing to the interpretability of our model and effectively performing morphological inflection on both seen and unseen morphological features [1].

1 Introduction

In this paper, we introduce an interpretable unsupervised morphological learning model that achieves performances comparable to supervised models in the acquisition of complex morphological rules. We demonstrate its abilities in addressing one of the most studied problems in the literature, morphological inflection in the SIGMORPHON Shared Tasks (Cotterell et al., 2016, 2017, 2018; Vylomova et al., 2020; Pimentel et al., 2021; Kodner et al., 2022; Goldman et al., 2023).

The unsupervised acquisition of morphological rules in humans is a natural process during language learning. This involves the analysis of word structures, recognition of stems and affixes, association of consistent meanings, and the integration of these elements into novel combinations, as explained by Clark (2017). These rules govern the appropriate structure of words to convey their intended meanings. For instance, when forming the present participle of the verb *"to bike"* it becomes *"biking,"* not *"bikeing"* necessitating the exclusion of the last vowel. Similarly, when evaluating the feasibility of a goal, we consider its *"attainability" (attain+able+ity)*, not *"attainityable" (attain+ity+able)*; maintaining the correct sequence of suffixes is crucial in this context. Given the inherent ability of humans to learn morphology unsupervisedly, it is essential to develop unsupervised neural models that can replicate this process. This analogy suggests that it should be feasible for a model to acquire morphological rules without explicit supervision.

In the intersection of computation and morphology, researchers have developed computational approaches to explore human morphology learning theories and address practical applications like spell checking, correction, automatic speech recognition, and statistical machine translation. The two-level morphology model (Koskenniemi, 1983), prevalent in the early stages, highlights the complexity of morphology, incorporating phonological alterations beyond a simple arrangement of morphemes. For example, in Turkish words like *bahçemden* and *garajımdan*, both indicating movement from a possessed place, the morpheme sequences differ (*+m+den* vs. *+ım+dan*) based on Turkish phonological rules. Two-level morphology dissects this into lexical and phonological levels, resulting in the correct surface forms. Finite-state transducers, exemplified by a Turkish morphological analyzer (Oflazer, 1993), have been utilized to study morphological processes across languages within the two-level formalism.

[1] Our code, data and experimental results will be available following the review period.

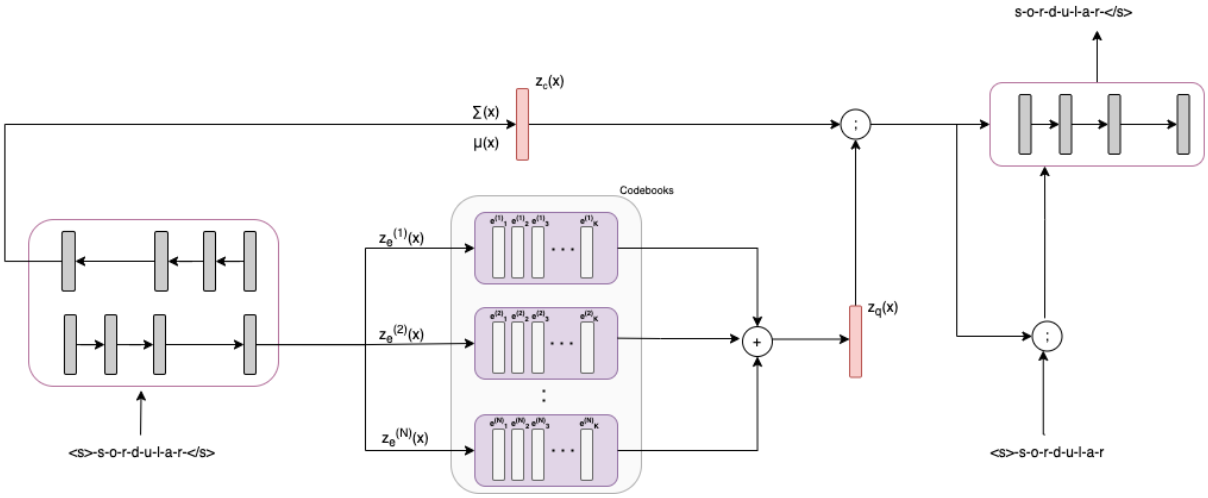

Figure 1: Our model: Multiple Codebook VQ-VAE

To evaluate unsupervised models from a morphological standpoint, it is essential to establish expectations for a model proficient in "learning" morphology. As outlined in (Goldsmith et al., 2017), the questions an unsupervised morphology learner should address include identifying component morphemes in words, recognizing alternative forms (allomorphs) like *-ler* and *-lar* in Turkish, understanding conditions for their usage, explaining alternative forms through phonological generalizations, determining permissible combinations of feature specifications, and unraveling morphological realization of each combination.

In this work, we propose the Multiple Codebook Vector Quantized Variational Autoencoder (VQ-VAE) (Van Den Oord et al., 2017) as an unsupervised morphology learner for text. Our approach entails establishing a continuous space and utilizing multiple codebooks. The model integrates codebook entries with the continuous space to generate words. We expect the model to discretize various morphological features in codebooks, thereby representing a word's lemma in continuous space. For example, one codebook may encode person features (e.g., 1st person singular, 3rd person plural), another may represent the tense of the word (e.g., present, future, past), and a separate codebook may handle the polarity of the word (positive or negative).

We evaluate our model's performance in morphological inflection, addressing the challenges it faces in learning crucial abilities such as allomorph recognition, phonological generalizations, and the realization of diverse morphological feature combinations. Additionally, we examine its generalization capabilities under both high and low resource data scenarios by testing it on unseen suffixation scenarios within the SIGMORPHON-UniMorph 2022 Shared Task 0 (Kodner et al., 2022). To gain insights into the model's learning, we further employ probing techniques for interpreting encoded information within latent representations.

Our primary contributions are:

- We introduce a novel and interpretable unsupervised model that achieves comparable performance to supervised models in learning morphological rules.
- The model exhibits robust performance in both high and low-resource scenarios for morphological inflection tasks.
- The model segregates word lemmas into continuous variables and their suffixes into discrete variables within codebooks. Additionally, across random runs, the model specializes each codebook with a unique morphological feature, thereby enhancing its interpretability.

2 Model

We extend the idea of Vector Quantised-Variational Autoencoders (VQ-VAE) (Van Den Oord et al., 2017) for text. The original VQ-VAE is an encoder-decoder model that aims to model image and speech data using discrete latent variables picked from a codebook having embeddings. The encoder outputs are replaced with the nearest vectors in

l_2 distance from the codebook. Then, the codebook embeddings are fed to the decoder, and the data reconstruction is aimed. In our model, we employ multiple codebooks, in contrast to the original VQ-VAE that uses only one. Additionally, we incorporate continuous variables, following the approach of the original VAE. Our expectation is that the model will specialize each codebook to capture distinct morphological features of a word, and as a result, the continuous space will be utilized to encode the lemma of a word. Specifically, we construct a VQ-VAE model with continuous and discrete variables with the following blocks: bidirectional GRU encoder, low-dimensional continuous space, varying number of codebooks for discrete space, and a unidirectional GRU decoder, as seen in Fig. 1. While the continuous part is regulated by KL divergence to standard Gaussian prior as in regular VAE, the discrete latent variables are obtained with quantization through multiple codebooks. The encoder q with parameters ϕ has the last forward hidden state $\overrightarrow{h_t}$, and the last backward hidden state $\overleftarrow{h_t}$ with d dimensional vectors. The mean μ and variance σ are learned by applying a linear transformation to the last backward hidden state $\overleftarrow{h_t}$. Then, using μ and σ, we estimate the continuous latent variable z_c. To make the learning step differentiable, we use the reparameterization trick (Kingma and Welling, 2013) and calculate $z_c = \mu_\phi(x) + \sigma_\phi(x) * \epsilon$ where $\epsilon \sim N(0, 1)$.

For quantization, we define the latent embedding space of codebooks as $e \in R^{N \times K \times D}$ where N is the number of codebooks, K is the number of entries in each codebook, D is the dimension of each embedding vector $e^{(n)}$ in the codebook. The $\overrightarrow{h_t}$ vector from the encoder is then linearly transformed into N vectors with dimension D. For each linearly transformed vector from encoder $z_e^{(n)}(x)$, the nearest embedding from $codebook^{(n)}$ is calculated:

$$\mathrm{q}(\mathrm{z}_q^{(n)} = k|x) = \begin{cases} 1 & \text{for k = argmin}_j ||z_e^{(n)}(x) - e_j^{(n)}||_2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Then we sum the quantized vectors $z_q^{(1)}, z_q^{(2)}, ...z_q^{(N)}$ and obtain $z_q(x)$ vector. We finally concatenate the quantized vector with the continous vector and feed it to the decoder as an initial hidden state. At each time step of decoding, we concatenate the continous vector z_c, quantized vector z_q, and the target token embedding.

The total objective for our model becomes:

$$\begin{aligned} L = \; & E_{z_c \sim q_\phi(z|x)}[\log p(x|z_c, z_q)] \\ & + \sum_{n=1}^{N} ||sg[z_e^{(n)}(x)] - e^{(n)}||_2^2 \\ & + \sum_{n=1}^{N} \beta ||z_e^{(n)}(x) - sg[e^{(n)}]||_2^2 \\ & - KL(q(z_c|x)||p(z_c))\} \end{aligned} \tag{2}$$

The initial component of the loss involves the reconstruction loss, where the model conditions on the continuous latent variable z_c and discrete latent variable z_q to reconstruct the observed data x. The subsequent element pertains to the overall vector quantization loss for each vector $z_e^{(n)}(x)$. Similar to the original VQ-VAE, the stop gradient operation (denoted as sg) is employed to facilitate the learning of codebook embeddings $e^{(n)}$. This operation ensures that the gradient of the applied term becomes zero during forward computation, converting it into a non-updated constant. In the second term, to minimize the l_2 distance between encoder outputs and codebook embeddings, only the codebook embeddings are updated. The third term involves updating only the encoder outputs, weighted by the parameter β to prevent the encoder outputs from growing faster than the codebook embeddings. Lastly, in the fourth term, we regulate the continuous vector using a standard Gaussian distribution.

3 Evaluation

In this section, we evaluate the performance of our unsupervised model in morphological inflection (see Section 3.1) and probe the latent variables of the model for morphological features (see Section 3.2). We conduct further evaluation of our model in the context of Sigmorphon-UniMorph 2022 Shared Task 0 (Kodner et al., 2022) in Section 3.3.

3.1 Morphological Inflection

At morphological inflection problem, a model takes a word's lemma and a morphological feature set as input, and generates the inflected target form of the word.

```
e.g.:
vermek + V;DECL;OBLIG;PL;2;NEG;PST
-> vermemeliydiniz
```

Morphological inflection, highlighted in (Cotterell et al., 2016), is crucial for generating and

analyzing words in a language based on inflected forms. This task aids in understanding word shapes and suffixation patterns, allowing models to generalize to unseen words by learning inflection rules. Particularly challenging in languages like Turkish with rich inflectional morphology, the task involves learning various morphological processes.

3.1.1 Experiments

For this problem, we conduct experiments using 4, 6, 8, and 12 codebooks, each containing 6, 8, and 12 entries. To determine the convergence of the model, we evaluate model's copying exact match accuracy and model's sampling quality: This involves sampling vectors from the continuous space and using a fixed entry combination from codebooks. We expect to observe inflections of different lemmas sharing the same suffix. This approach ensures that the model leverages the codebooks to generate a word. We provide results for the best model with 4 codebooks and 8 entries per codebook. Full results of models with different codebook-entry configurations can be found in the Appendix B.

	train	**test**
# total words	404896	1446
# unique lemma	588	536
# unique feature sets	703	616

Table 1: Dataset statistics

We filter the Turkish Unimorph dataset (McCarthy et al., 2020) for verbs. The dataset includes triples in the format (lemma, inflected form, feature set), such as (çıkarmak, çıkaracağım, V;DECL;IND;SG;1;POS;FUT). We augment the dataset with verbs from the large training set of Turkish in Sigmorphon 2022 Shared Task-0 (Kodner et al., 2022). In this way we have a dataset with 404,896 words, featuring 588 unique lemmas and 703 unique morphological feature sets. For evaluation, we also use the shared task test set which contains 1,446 verbs. It's important to note that all lemmas and feature sets are encountered during training, although not together in the same triple.

During training, our unsupervised model relies solely on observing the raw surface forms of words without explicit morphological feature sets. To address the inflection task using our unsupervised model, we initially associate codebook entries with the corresponding feature sets. This process involves the following steps: At test time, we present all target words in the test set to the model and observe its selection of codebook entries for each word while copying them. For example, to map the relevant codebook entries for a feature set like V;DECL;IND;SG;1;POS;FUT, we identify the most frequently selected codebook entries when copying words with this specific feature set. We then use these mapped entries in conjunction with a word's lemma to inflect it into the target form with that particular feature set. This inflection, using the mapped entries, is referred to as a top-1 match. Moreover, we track the second most frequently chosen codebook entries, labeling it as a top-2 match.

3.1.2 Baselines

We use the baseline models provided by the recent SIGMORPHON Shared Tasks, which have been consistently employed in previous iterations of the shared task.

Unsupervised We use the non-neural baseline model provided by the shared task as an unsupervised baseline model. The model initially aligns input/output training examples using the Levenshtein distance. The system presupposes that each input-output pair can be segmented into a prefixation part (`Pr`), a stem part (`St`), and a suffixation part (`Su`), based on the presence of initial or trailing zeroes in the inputs or outputs. Subsequently, the system extracts a set of prefix-changing rules based on the `Pr` pairings and a set of suffix-changing rules based on `St+Su` pairings. During generation, the longest suffix rule that is applicable to a lemma form to be inflected is employed.

We also perform unsupervised training on the closely related work by Zhou and Neubig (2017), initially trained using a mix of supervised and semi-supervised approaches. Their semi-supervised method involves reconstructing target and source words using inferred labels and training MLP classifiers for each morphological feature label. They employ a continuous vector for encoding word lemmas, regularized by KL divergence towards a standard Gaussian prior. Morphological feature encoding utilizes MLPs as discriminative classifiers, incorporating the Gumbel-Max trick for differentiating discrete latent variables. An attention mechanism facilitates feature label inference, and the lemma vector, attention vector, and target token are concatenated to the decoder at each time step. KL annealing scheduling and input operation

Lemma	Feature set	Codebook entries	Inflected word
dondurmak	V;OBLIG;SG;2;POS;PST;INTR	7;5;6;3	dondurmalı mıydın
ekşimek	V;OBLIG;SG;2;POS;PST;INTR	7;5;6;3	ekşimeli miydin
sanmak	V;DECL;PL;1;POS;PST;INFR	1;1;0;0	sanmışız
ölçmek	V;DECL;PL;1;POS;PST;INFR	1;1;0;0	ölçmüşüz
götürmek	V;DECL;PL;1;NEG;FUT;INFR	1;7;5;6	götürmeyecekmişiz
taşmak	V;DECL;PL;1;NEG;FUT;INFR	1;7;5;6	taşmayacakmışız

Table 2: Inflection results. The model employs the same codebook combinations for identical feature sets and can apply different harmony rules, such as *-meli miydin* / *-malı mıydın* and *-mışız* /*-müşüz* and *-meyecekmişiz* /*-mayacakmışız*.

Model	E.M. Acc.
Ours (top-1 match)	0.94
Ours (top-2 match)	**0.98**
Baseline (Unsupervised)	0.38
Baseline (Supervised)	0.99

Table 3: Performance of models on verbs in morphological inflection: Our model demonstrates comparable performance to the supervised baseline, achieving nearly 100% accuracy. E.M. Acc.: Exact match accuracy.

dropout are employed to prevent posterior collapse during generation. However, in our unsupervised setups, the model struggles to distinguish lemmas and suffixes as effectively as in supervised cases. We are unable to identify any specifications for classifiers related to morphological features, preventing us from mapping the morphological features to classes. Consequently, the model's capability to perform morphological inflection is hindered. Additional details can be found in the Appendix D.

Supervised We employ a baseline from the recent years of the shared tasks (Pimentel et al., 2021; Kodner et al., 2022; Goldman et al., 2023), which inspired many other works on the inflection problem such as Yang et al. (2022); Merzhevich et al. (2022); Forster and Meister (2020); Canby et al. (2020), specifically a character-level transducer proposed by Wu et al. (2021). This transducer is based on transformers, utilizing special position and type embeddings for morphological features and word characters. In their approach, positional encodings for features are set to 0, as the order of features is not considered important, and only word characters are counted. Additionally, a special type token is introduced to indicate whether a token represents a feature or a word character.

3.1.3 Results & Analysis

The model achieves a 94% accuracy in top-1 matches and a 98% accuracy in top-2 matches for inflection, as shown in Table 3. While the unsupervised baseline exhibits poor performance on the task, the supervised baseline demonstrates nearly perfect performance, and our results indicate comparable performance to that model. We also investigate the model's codebook selection for given words. Our findings reveal that **the model selects the same codebook-entry combinations for words that share the same suffix**, as shown in Table 4. Moreover, by employing these identical entries, **the model learns to apply morphosyntactic rules, preserving vowel harmony** as illustrated in Table 2. By employing the top-2 match selection instead of the top-1, 56 errors were resolved, with 2 errors pertaining to lemma corrections and the remaining errors involving suffix adjustments. Therefore, the results indicate that **the model performs strongly in inflection by effectively mapping the appropriate suffix to the codebook entries.**

3.2 Probing

Probing is a technique used to interpret neural models by identifying encoded information in their representations. The use of classifiers enables us to evaluate if these representations correspond to human classification patterns. For morphological evaluation, a probing procedure can be employed to analyze the morphological features of words. In this section, we evaluate our model's ability to capture the tense, person, and polarity features of verbs (e.g., *okuyacaklar* -> 3rd person plural, future tense, positive).

3.2.1 Experiments

We analyze the representation of morphological features in both continuous vector and discrete codebook vectors. To achieve this, we maintain

Codebook entries	Words
7;5;3;7	fotoğraf çekiyor olmalıydın süründürüyor olmalıydın yontuluyor olmalıydın eğleniyor olmalıydın
2;7;5;6	programlattırmayacakmışım süründürtmeyecekmişim kırdırtmayacakmışım göndermeyecekmişim
4;5;5;6	kanıtlamadıydınız birleşmediydiniz eğilmediydiniz dolmadıydınız

Table 4: Model's codebook entry selections. It employs the same entry combinations for words that have the same suffix. Combination 1: (past perfect cont. tense) Combination 2: (negative inferential future tense). Combination 3: (negative past tense)

fixed model parameters and introduce a linear layer on the model's continuous latent variables z_c, quantized variables which are separate codebook embeddings, and their sum z_q. This linear layer is trained to predict the morphological feature. The 'person' feature encompasses 6 classes: singular and plural for 1st, 2nd, and 3rd persons. The 'tense' feature consists of 3 classes: present, past, and future. Finally, the 'polarity' feature comprises 2 classes: positive and negative. We use the majority of classes in the test set as our baseline.

3.2.2 Results & Analysis

As indicated in Table 5, the continuous vector z_c, intended to encode the lemma, exhibits performance close to the baseline score for each morphological tag classification. This was anticipated since it is not supposed to contain information related to the suffix. Conversely, the quantized vector z_q encodes a significant portion of suffix-related information and effectively clusters the words in its space (refer to Fig. 2). Notably, there is a clear distinction in the person tag for words within codebook-0, whereas the other codebooks exhibit performances comparable to the baseline. Regarding the tense feature, codebook-1 seems to encode that information. However, for polarity, there isn't a significant discrimination, as codebook-2 and codebook-3 display similar performances. The results suggest that **across random runs, the model specializes distinct codebooks for different morphological features.** Full results can be found in Appendix C.

	Person	Tense	Polarity
z_c	0.25	0.48	0.63
z_q	0.99	0.98	0.86
cbook-0	**0.98**	0.50	0.52
cbook-1	0.20	**0.88**	0.54
cbook-2	0.20	0.54	**0.75**
cbook-3	0.18	0.49	0.73
baseline	0.18	0.48	0.52

Table 5: Model's probing results. Codebooks are specialized for different morphological features, while continuous part exhibits significantly lower performance. cbook: codebook.

In summary of Sec. 3.1 and Sec. 3.2, we present a model that encodes lemmas into continuous vectors, translating morphological features in the suffix into codebook entries. We also show that these codebooks specialize in various morphological features, such as Person, Tense, and Polarity. Furthermore, we demonstrate the model's capability to inflect a lemma with a suffix mapped in codebook entries, generating a newly inflected word not encountered during training.

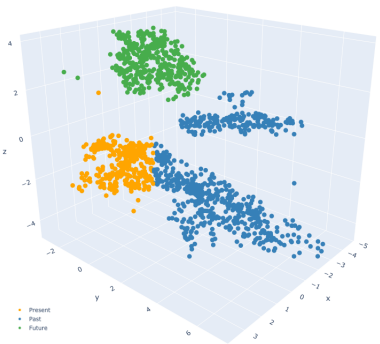

Figure 2: Visualization of tense probe logits in quantized vector z_q. The model clusters words based on tense suffixation.

3.3 Evaluation on SIGMORPHON-UniMorph 2022 Shared Task 0

In this section, we show that our **model exhibits comparable performance even in low-resource scenarios when compared to supervised models.** In the SIGMORPHON 22 Shared Task 0 (Kodner et al., 2022), the primary focus is to evaluate the capacity of models in generalizing to unseen lemmas and features. The task includes conditions of both large and small datasets, organized based on overlaps in lemmas and features. We focus on sce-

Gold target	Ours
büyüyor olacaklar	büyümüş olacaklar
kullanıyor olmamalısınız	kullanıyor olmamalıyız
delecek olmayacakmışız	deler olacakmışız
dedikodu yaparlardı	dedikodu yapacaklardı
hareket ediyorsun	hareket ediyor olmalısın

Table 6: Model's errors with unseen feature sets. Although it correctly identifies lemmas, it struggles to inflect them into the accurate target forms.

narios with lemma overlap in the task, where test pair lemmas are included in the training data, but their feature sets are novel.

3.4 Experiments

We filter the original large dataset, reducing it from 7,000 to 5,273 instances by selecting only verbs. In the test set, we have 1,446 instances with 731 featuring both overlap and 715 showing lemma overlap, implying the presence of words with novel feature sets. Our model is trained using 6 codebooks, each containing 8 entries.

3.5 Models

All models, except Flexica (Sherbakov and Vylomova, 2022), use transformers in a supervised fashion. CLUZH (Wehrli et al., 2022) is a character-level neural transducer handling edit actions like insertion, deletion, substitution, and copy. UBC (Yang et al., 2022) improves Wu et al. (2021) with reverse positional embeddings for better suffix handling. TüM-M (Merzhevich et al., 2022) also adapts Wu et al. (2021) for predicting a distribution over states of FST. OSU (Elsner and Court, 2022) uses a transformer with an analogical exemplar model for inflection, effective when target cell examples are available. Flexica employs refined alignment patterns, learning transformation patterns through maximal continuous matches between lemmas and inflected forms. Extraction involves finding the longest common substring, recursively extending until no more common characters are found, and then enriching patterns with concrete characters from training samples.

3.6 Results & Analysis

As indicated in Table 7, our model surpasses three systems in both top-1 and top-2 matches. In top-2 matches, our model achieves a 88% accuracy with 171 mistakes out of 1,446 test instances. Despite having no unseen lemma between our training and

System	E.M. Acc.
UBC	0.98
CLUZH	0.92
OSU	0.48
Flexica	0.38
TüM-M	0.22
Ours (top1-match)	0.81
Ours (top2-match)	0.88

Table 7: Performance of submitted systems for verbs in the large training condition in the SIGMORPHON-UniMorph 2022 Shared Task-0. E.M. Acc.: Exact match accuracy.

test set, almost half of the test set comprises words with novel feature sets. We observe that our model accurately captures 91% of cases for seen feature sets, while for unseen feature sets, the model correctly generates 85% of the words.

Error analysis We analyze our model's errors in top-2 matches for seen and unseen features. We observe that %63 of ours models errors cause because of the unseen feature sets. Out of errors, the models generated novel words that were not encountered during training. As seen in Table 6, in most of the cases, our model fails to form the correct inflected target word due to incorrect suffixation. However, we observe that the model still preserves harmony rules, such as the -meli/-malı obligation suffix, where models CLUZH and OSU struggle. For instance, with the lemma ending with the vowel *a*, such as açılmak, it should be *açılmalıyım*, not *açılmeliyim*. Similarly, with the lemma *asmak* and the related 3rd person plural, it should be *asmamalısınız*, not *asmamelisiniz*. In these examples, our model is able to preserve vowel harmony where CLUZH and OSU fail.

4 Importance of Directionality

In this section, we investigate the impact of our directional choice, where we assign the last backward hidden state $\overleftarrow{h_t}$ to the continuous vector, aimed at encoding the lemma, and the last forward hidden state $\overrightarrow{h_t}$ to the codebooks, intended to encode morphological features in the suffix. Given the structure of agglutinative languages, where the lemma starts on the left and suffixation occurs on the right, we anticipate this approach to be effective, introducing a form of inductive bias. To understand its effect, we concatenate the last forward and back-

ward hidden states $[\overrightarrow{h_t}; \overleftarrow{h_t}]$, and input the resulting vector into both the continuous vector and the codebooks. We conduct experiments with 4, 6, and 8 codebooks, each having 6, 8, and 12 entries, while maintaining other model dimensions. The experiments are rerun with three different random initializations, revealing three types of observed problems: (1) Suffix information is not entirely encoded in the discrete part, but partially encoded in the continuous part with lemma. (2) Lemma information is not entirely encoded in the continuous part but is partially embedded in the discrete part with suffixes, leading to a significant increase in codebook entry usage. This suggests that the model does not effectively cluster words based on suffixation, instead encoding most of the word information into the codebook. (3) Lemma information is entirely encoded in the discrete part, while suffix information is entirely encoded in the continuous part. We give futher evidences for these problems in Appendix E).

In every setup, the absence of separation between lemma and suffix into continuous and discrete parts also interferes with the mapping of morphological tags in codebook entries. Consequently, morphological inflection cannot be performed well. While the first two problems are partially evident in several runs of the model with an inductive bias, we were still able to achieve good convergence with the majority of configurations, which is challenging to replicate without using the directionality. Consequently, we argue that **the directionality helps the model in distinguishing between lemma in the continuous and suffix in the discrete parts.** Nevertheless, further experiments without directionality may provide better insights.

5 Related Work

The unsupervised study of morpheme boundaries dates back years. Harris (1955)'s pioneering work introduces a heuristic based on letter successor/predecessor tokens, counting the different letters after a morpheme candidate x. Subsequent works enhance this approach by analyzing the frequency distribution of successor tokens and calculating entropy to measure predictability.The Morfessor family, including Morfessor Baseline (Creutz and Lagus, 2002), Morfessor FlatCat (Grönroos et al., 2014), and Morfessor EM+Prune (Grönroos et al., 2020), utilizes generative models for language morpheme learning. Morfessor Baseline optimizes parameters through MAP estimation, adhering to the Maximum Length Description (MDL) principle. Morfessor EM+Prune starts with a seed lexicon of the most frequent subwords and prunes during training. Additionally, Adaptor Grammar (Johnson et al., 2006) and MorphAGram (Eskander et al., 2020) contribute to unsupervised morphological segmentation, incorporating adaptors like the Pitman-Yor Process (Pitman and Yor, 1997). Previous work in morphological inflection includes supervised learning techniques. Durrett and DeNero (2013) employs alignment and learns edit operations, while Kann and Schütze (2016) proposes a neural approach using an encoder-decoder architecture with soft attention (Bahdanau et al., 2015) and stacked GRUs (Cho et al., 2014). Anastasopoulos and Neubig (2019) proposes data augmentation by generating hallucinated data in lemma-feature tag-target pairs. They replace shared substrings longer than three characters with random characters, resulting in hallucinated lemma-tag triples. Some probing studies on RNNs include Shi et al. (2016); Conneau et al. (2018). Criticisms regarding probe reliability and classification limitations have prompted the consideration of simpler probes, emphasizing information-theoretic measures over accuracy (Hewitt and Liang, 2019; Voita and Titov, 2020; Pimentel et al., 2020). The studies also explore causal relations and latent ontologies, providing insights into feature usage and representations (Vanmassenhove et al., 2017; Elazar et al., 2021; Giulianelli et al., 2018; Lasri et al., 2022).

6 Conclusion & Future Work

This work presents a novel and interpretable unsupervised model that achieves performance comparable to supervised models in learning morphological rules. The model exhibits robust performance on morphological inflection tasks, particularly in low-resource scenarios, producing results comparable to those of supervised models. Notably, the model separates the lemma of words into continuous variables and their suffix into discrete variables within codebooks. Across multiple runs, the model consistently specializes each codebook with distinct morphological features, contributing to enhanced interpretability. Future work may involve exploring different morphological tasks, such as unsupervised paradigm completion and unsupervised paradigm clustering.

Limitations

Our proposed model strategically incorporates the bidirectionality of the encoder as a bias in its architecture. This bias is leveraged to effectively capture word lemmas on the left and suffixation on the right. Our tailored approach significantly enhances the model's performance in Turkish, an agglutinative language characterized by heavy suffixation. In agglutinative languages, lemmas typically appear on the left, while suffixation occurs on the right. Therefore, we anticipate our model to demonstrate robust performance in favor of agglutinative languages, such as Finnish, Hungarian, and others. However, further experimentation may be needed to adjust directionality for languages with varying morphological typologies.

Ethics Statement

We foresee no ethical concerns related to the methods outlined in this paper.

References

Antonios Anastasopoulos and Graham Neubig. 2019. Pushing the limits of low-resource morphological inflection. 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 984–996, Hong Kong, China. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Marc Canby, Aidana Karipbayeva, Bryan Lunt, Sahand Mozaffari, Charlotte Yoder, and Julia Hockenmaier. 2020. University of Illinois submission to the SIGMORPHON 2020 shared task 0: Typologically diverse morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 137–145, Online. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *SSST@EMNLP*.

Eve V Clark. 2017. Morphology in language acquisition. *The handbook of morphology*, pages 374–389.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. *ArXiv*, abs/1805.01070.

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

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. Conll-sigmorphon 2017 shared task: Universal morphological reinflection in 52 languages. In *CoNLL Shared Task*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 2016 Meeting of SIGMORPHON*, Berlin, Germany. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30. Association for Computational Linguistics.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *NAACL*.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175.

Micha Elsner and Sara Court. 2022. OSU at SigMorphon 2022: Analogical inflection with rule features. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 220–225, Seattle, Washington. Association for Computational Linguistics.

Ramy Eskander, Francesca Callejas, Elizabeth Nichols, Judith L. Klavans, and Smaranda Muresan. 2020. Morphagram, evaluation and framework for unsupervised morphological segmentation. In *LREC*.

Martina Forster and Clara Meister. 2020. SIGMORPHON 2020 task 0 system description: ETH Zürich team. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 106–110, Online. Association for Computational Linguistics.

Mario Giulianelli, John Harding, Florian Mohnert, Dieuwke Hupkes, and Willem H. Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *BlackboxNLP@EMNLP*.

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

John A. Goldsmith, Jackson L. Lee, and Aris Xanthos. 2017. Computational learning of morphology.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2020. Morfessor EM+Prune: Improved subword segmentation with expectation maximization and pruning. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3944–3953, Marseille, France. European Language Resources Association.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1177–1185, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Zellig S. Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. *ArXiv*, abs/1909.03368.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. *Advances in neural information processing systems*, 19.

Katharina Kann and Hinrich Schütze. 2016. Med: The lmu system for the sigmorphon 2016 shared task on morphological reinflection. In *SIGMORPHON*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

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

Kimmo Koskenniemi. 1983. Two-level model for morphological analysis. In *IJCAI*.

Karim Lasri, Tiago Pimentel, Alessandro Lenci, T. Poibeau, and Ryan Cotterell. 2022. Probing for the usage of grammatical number. In *ACL*.

Arya D. McCarthy, Christo Kirov, Matteo Grella, Amrit Nidhi, Patrick Xia, Kyle Gorman, Ekaterina Vylomova, Sabrina J. Mielke, Garrett Nicolai, Miikka Silfverberg, Timofey Arkhangelskiy, Nataly Krizhanovsky, Andrew Krizhanovsky, Elena Klyachko, Alexey Sorokin, John Mansfield, Valts Ernštreits, Yuval Pinter, Cassandra L. Jacobs, Ryan Cotterell, Mans Hulden, and David Yarowsky. 2020. UniMorph 3.0: Universal Morphology. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3922–3931, Marseille, France. European Language Resources Association.

Tatiana Merzhevich, Nkonye Gbadegoye, Leander Girrbach, Jingwen Li, and Ryan Soh-Eun Shim. 2022. SIGMORPHON 2022 task 0 submission description: Modelling morphological inflection with data-driven and rule-based approaches. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 204–211, Seattle, Washington. Association for Computational Linguistics.

Kemal Oflazer. 1993. Two-level description of turkish morphology. *Literary and Linguistic Computing*, 9:137–148.

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

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. *ArXiv*, abs/2004.03061.

Jim Pitman and Marc Yor. 1997. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pages 855–900.

Andreas Sherbakov and Ekaterina Vylomova. 2022. Morphology is not just a naive Bayes – UniMelb submission to SIGMORPHON 2022 ST on morphological inflection. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 240–246, Seattle, Washington. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *EMNLP*.

Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Eva Vanmassenhove, Jinhua Du, and Andy Way. 2017. Investigating 'aspect' in nmt and smt: Translating the english simple past and present perfect.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. *ArXiv*, abs/2003.12298.

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

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

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2021. Applying the transformer to character-level transduction. In *EACL*.

Changbing Yang, Ruixin (Ray) Yang, Garrett Nicolai, and Miikka Silfverberg. 2022. Generalizing morphological inflection systems to unseen lemmas. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 226–235, Seattle, Washington. Association for Computational Linguistics.

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. *arXiv preprint arXiv:1704.01691*.

A Hyperparameter details

For our main model used in Section 3.1 and Section 3.2, the configuration includes a bidirectional GRU encoder with a hidden size of 256, an unidirectional GRU decoder with a hidden size of 1024, a continuous vector of 100 dimensions, 4 codebooks with 8 entries per each codebook, 128 dimensions in each codebook entry, 128 dimensions in encoder-decoder input token embeddings, decoder input dropout set to 0.2, a batch size of 64, Adam optimizer with β values of (0.5, 0.99), a learning rate of 0.0005, KL weight of 1.0 with an annealing strategy starting from epoch 5, and a total of 50 epochs.

For our main model used in Section 3.3, the configuration comprises a bidirectional GRU encoder with a hidden size of 256, an unidirectional GRU decoder with a hidden size of 256, a continuous vector of 100 dimensions, 6 codebooks with 8 entries per each codebook, 128 dimensions in each codebook entry, 128 dimensions in encoder-decoder input token embeddings, decoder input dropout set to 0.1, a batch size of 16, Adam optimizer with β values of (0.5, 0.99), a learning rate of 0.0005, KL weight of 0.05 with an annealing strategy starting from epoch 10, and a total of 500 epochs.

B Different codebook-entry configurations

Test acc.	0.93
Inflection acc. (Top-1)	0.30
Inflection acc. (Top-2)	0.40
Train # used entries	1122
Test # used entries	577

Table 8: 4x6 Training results. KL=1.0.

	Person	**Tense**	**Polarity**
z_c	0.27	0.54	0.95
z_q	0.96	0.86	0.59
cbook-0	0.31	**0.72**	0.54
cbook-1	0.18	0.48	**0.56**
cbook-2	**0.72**	0.52	0.53
cbook-3	0.26	0.53	**0.56**
baseline	0.18	0.48	0.52

Table 9: 4x6 Probing accuracy results.

Test acc.	0.87
Inflection acc. (Top-1)	0.48
Inflection acc. (Top-2)	0.74
Train # used entries	11891
Test # used entries	1259

Table 10: 4x12 Training results. KL=1.0.

	Person	**Tense**	**Polarity**
z_c	0.28	0.48	0.71
z_q	0.96	0.96	0.98
cbook-0	**0.75**	0.49	0.53
cbook-1	0.30	0.62	**0.86**
cbook-2	0.20	0.59	0.53
cbook-3	0.19	**0.67**	0.69
baseline	0.18	0.48	0.52

Table 11: 4x12 Probing accuracy results.

Test acc.	0.94
Inflection acc. (Top-1)	0.54
Inflection acc. (Top-2)	0.83
Train # used entries	12089
Test # used entries	1270

Table 12: 6x6 Training results. KL=1.0.

	Person	**Tense**	**Polarity**
z_c	0.26	0.51	0.64
z_q	0.98	0.95	0.83
cbook-0	0.50	0.50	0.55
cbook-1	0.20	0.61	0.54
cbook-2	0.20	**0.69**	**0.78**
cbook-3	0.19	0.57	0.57
cbook-4	**0.65**	0.57	0.55
cbook-5	0.19	0.57	0.53
baseline	0.18	0.48	0.52

Table 13: 6x6 Probing accuracy results.

Test acc.	0.99
Inflection acc. (Top-1)	0.72
Inflection acc. (Top-2)	0.92
Train # used entries	23104
Test # used entries	1291

Table 14: 6x8 Training results. KL=0.5.

	Person	**Tense**	**Polarity**
z_c	0.25	0.51	0.64
z_q	0.99	0.95	0.87
cbook-0	**0.85**	0.51	0.52
cbook-1	0.18	0.48	0.56
cbook-2	0.20	**0.90**	0.53
cbook-3	0.33	0.54	0.63
cbook-4	0.18	0.50	0.65
cbook-5	0.19	0.50	**0.85**
baseline	0.18	0.48	0.52

Table 15: 6x8 Probing accuracy results.

Table 16: Summary of training and probing results. We present the best performances of various configurations with KL values of 0.2, 0.5, and 1.0. Since models employing 12-codebooks exhibit poor performance in both inflection and probing tasks; we exclude them from our analysis.

Test acc.	0.99
Inflection acc. (Top-1)	0.82
Inflection acc. (Top-2)	0.95
Train # used entries	31197
Test # used entries	1327

Table 17: 6x12 Training results. KL=0.5.

	Person	**Tense**	**Polarity**
z_c	0.30	0.51	0.74
z_q	0.99	0.95	0.91
cbook-0	0.20	**0.78**	0.68
cbook-1	0.20	0.55	0.67
cbook-2	0.19	0.68	**0.74**
cbook-3	0.18	0.49	0.59
cbook-4	0.67	0.62	0.54
cbook-5	**0.68**	0.52	0.66
baseline	0.18	0.48	0.52

Table 18: 6x12 Probing accuracy results.

Test acc.	0.99
Inflection acc. (Top-1)	0.87
Inflection acc. (Top-2)	0.96
Train # used entries	27073
Test # used entries	1312

Table 19: 8x6 Training results. KL=0.5.

	Person	**Tense**	**Polarity**
z_c	0.26	0.49	0.65
z_q	0.98	0.86	0.95
cbook-0	0.19	0.48	0.53
cbook-1	**0.84**	0.49	0.54
cbook-2	0.68	0.49	0.53
cbook-3	0.18	0.49	0.57
cbook-4	0.20	0.60	0.59
cbook-5	0.18	0.50	0.71
cbook-6	0.19	**0.65**	**0.89**
cbook-7	0.27	0.62	0.54
baseline	0.18	0.48	0.52

Table 20: 8x6 Probing accuracy results.

Test acc.	0.99
Inflection acc. (Top-1)	0.84
Inflection acc. (Top-2)	0.95
Train # used entries	29251
Test # used entries	1277

Table 21: 8x8 Training results. KL=0.5.

	Person	**Tense**	**Polarity**
z_c	0.28	0.53	0.71
z_q	0.99	0.99	0.99
cbook-0	0.20	0.66	0.54
cbook-1	0.36	0.50	0.60
cbook-2	0.19	0.58	0.67
cbook-3	0.19	**0.68**	0.82
cbook-4	0.19	0.49	0.57
cbook-5	0.64	0.49	0.53
cbook-6	**0.90**	0.66	0.53
cbook-7	0.20	0.58	**0.96**
baseline	0.18	0.48	0.52

Table 22: 8x8 Probing accuracy results.

Table 23: Summary of training and probing results. We present the best performances of various configurations with KL values of 0.2, 0.5, and 1.0. Since models employing 12-codebooks exhibit poor performance in both inflection and probing tasks; we exclude them from our analysis.

C 5 Different random runs with 4x8 codebooks

Test acc.	0.98
Inflection acc. (Top-1)	0.73
Inflection acc. (Top-2)	0.82
Train # used entries	2656
Test # used entries	811

Table 24: RUN 1: Training results.

	Person	**Tense**	**Polarity**
z_c	0.25	0.50	0.75
z_q	0.99	0.90	0.65
cbook-0	0.20	0.53	0.55
cbook-1	0.49	0.50	0.54
cbook-2	0.21	**0.81**	**0.63**
cbook-3	**0.58**	0.49	0.56
baseline	0.18	0.48	0.52

Table 25: RUN 1 Probing accuracy results.

Test acc.	0.95
Inflection acc. (Top-1)	0.39
Inflection acc. (Top-2)	0.59
Train # used entries	3085
Test # used entries	933

Table 26: RUN 2: Training results.

	Person	**Tense**	**Polarity**
z_c	0.29	0.50	0.69
z_q	0.98	0.88	0.81
cbook-0	0.19	**0.64**	0.51
cbook-1	**0.90**	0.55	0.52
cbook-2	0.18	0.49	0.54
cbook-3	0.20	0.62	**0.74**
baseline	0.18	0.48	0.52

Table 27: RUN 2 Probing accuracy results.

Test acc.	0.98
Inflection acc. (Top-1)	0.94
Inflection acc. (Top-2)	0.98
Train # used entries	2621
Test # used entries	779

Table 28: RUN3: Training results.

	Person	**Tense**	**Polarity**
z_c	0.25	0.48	0.63
z_q	0.99	0.98	0.86
cbook-0	**0.98**	0.51	0.52
cbook-1	0.20	**0.88**	0.53
cbook-2	0.20	0.55	**0.74**
cbook-3	0.18	0.50	0.72
baseline	0.18	0.48	0.52

Table 29: RUN 3 Probing accuracy results.

Test acc.	0.95
Inflection acc. (Top-1)	0.74
Inflection acc. (Top-2)	0.93
Train # used entries	2624
Test # used entries	921

Table 30: RUN 4: Training results.

	Person	**Tense**	**Polarity**
z_c	0.27	0.54	0.68
z_q	0.92	0.88	0.83
cbook-0	0.21	**0.72**	**0.79**
cbook-1	0.30	0.50	0.50
cbook-2	0.45	0.54	0.56
cbook-3	**0.50**	0.48	0.62
baseline	0.18	0.48	0.52

Table 31: RUN 4 Probing accuracy results.

Test acc.	0.98
Inflection acc. (Top-1)	0.96
Inflection acc. (Top-2)	0.97
Train # used entries	2478
Test # used entries	736

Table 32: RUN 5: Training results.

	Person	**Tense**	**Polarity**
z_c	0.25	0.49	0.67
z_q	0.95	0.96	0.90
cbook-0	**0.59**	0.49	0.54
cbook-1	0.33	0.52	**0.78**
cbook-2	0.35	**0.64**	0.56
cbook-3	0.18	0.58	0.70
baseline	0.18	0.48	0.52

Table 33: RUN 5 Probing accuracy results.

D Related Model: MSVAE

We experiment with 4,6 and 8 MLP classifiers, each designed with 8 classes for every morphological feature. Additionally, we adjust the KL ratio to 1.0 to encourage the model to use discrete vectors from the classifiers. We observe that the model uses a small subset of classes for the test set, (which originally had 616 unique feature sets) suggesting that it exclusively relies on the continuous vector and does not make use of the discrete vectors from the classifiers. Consequently, the model fails to differentiate the lemma via the continuous part and the suffix-related morphological features via the classifiers. This results in inconsistencies in sampling as seen in Table 36, generating different suffixations even when the same morphological classes are given as input for the word.

Setting	Copy acc.	# Used Classes
4x8	0.94	53
6x8	0.96	54
8x8	0.96	60

Table 34: Training results of MSVAE with various number of classifiers.

Predicted classes	Words
2;6;6;6	bıkıyor olacaktım iyileşiyor olmayacaklar mıymış gizlenmez misiniz açılmalı mıydınız
5;4;1;1	kaynaştırılıyor olmalı mıyım hava atacak olacak mıymışsın güzelleştiriyor olacakmışsınız öğretiyor olmayacaklar mıydı
7;6;1;1	gülünçleşir olmayacakmışsın buharlaşıyor olacak mısınız fındık kıracak olacaklarmış ilerletiyor olmayacaklar mı

Table 35: Model's classifications with 4x8 MLPs. The model fails to use combinations specific to the same suffix.

sample 1	otostop çekermişsin
sample 2	darılmalı mıydık
sample 3	üzmemişlermiş
sample 4	sünüyor olmaz mıydı
sample 5	havlu atıyor muydunuz

Table 36: Sampled words with 4x8 MLPs. Continous vectors are sampled from a Gaussian distribution, and a specific class combination is selected from the classifiers. We do not observe consistent patterns in suffix usage.

E Importance of Direction

Problem (1): Suffix information is not entirely encoded in the discrete part, but partially encoded in the continuous part with lemma. An example of this case occurs with a model with 4 codebooks and 8 entries. The model only achieves a 12% accuracy in top-1 match and 18% accuracy in top-2 match for inflection. This is confirmed by sampled words as in Table 37 and probing experiments as seen in Table 38.

sample 1	bulamadı mı
sample 2	dalamadı mı
sample 3	coşmadım mı
sample 4	kopmadım
sample 5	boyamadım

Table 37: Sampled words with 4x8 codebooks. Continuous vectors are sampled from a Gaussian distribution, and a specific entry combination is selected from the codebooks. The model exhibits a slight inconsistency with respect to suffix.

	Person	**Tense**	**Polarity**
z_c	0.67	**0.70**	**0.88**
z_q	**0.97**	0.63	0.72
cbook-0	0.55	0.64	0.53
cbook-1	0.20	0.50	0.70
cbook-2	0.19	0.59	0.67
cbook-3	0.36	0.49	0.54
baseline	0.18	0.48	0.52

Table 38: Probing results for the model with 4-codebooks x 8-entries with no inductive bias. Suffix-related information is encoded into a continuous vector, which is expected to solely represent the lemma.

Problem (3): Lemma information is entirely encoded in the discrete part, while suffix information is entirely encoded in the continuous part. An example of this case occurs with a model with 6 codebooks and 6 entries. The model achieves a 3% accuracy in top-1 match and 9% accuracy in top-2 match for inflection. This is confirmed by sampled words as in Table 39 and probing experiments as seen in Table 40.

sample 1	canlandırılmış mıymış
sample 2	canlandırtılmış mıydık
sample 3	canlandırtılmışız
sample 4	canlandırılmıştım
sample 5	canlandırılmış olmamalıyız

Table 39: Sampled words with 6x6 codebooks. Continuous vectors are sampled from a Gaussian distribution, and a specific entry combination is selected from the codebooks. The model uses the same lemma but alters the suffixation, which is expected to be the opposite.

	Person	**Tense**	**Polarity**
z_c	**0.99**	**0.83**	**0.97**
z_q	0.30	0.63	0.50
cbook-0	0.30	0.49	0.54
cbook-1	0.20	0.48	0.56
cbook-2	0.20	0.48	0.65
cbook-3	0.20	0.49	0.53
cbook-4	0.19	0.61	0.60
cbook-5	0.20	0.48	0.54
baseline	0.18	0.48	0.52

Table 40: Probing results for the model with 6-codebooks x 6-entries with no inductive bias. Suffix-related information is encoded into a continuous vector, which is expected to solely represent the lemma.