# NLMT-v2: A Variational Autoencoder for Language Recommendation

Felipe Gomes Lopes

September 27, 2025

**Abstract**

This paper introduces NLMT-v2, a language recommendation system designed to suggest new languages for users to learn based on their existing linguistic skills. We propose a novel approach using a Variational Autoencoder (VAE) to learn a latent representation of user language profiles. Our model is trained on a dataset of user language proficiency and can generate personalized recommendations. We also present a framework for hyperparameter tuning using Optuna and expose the recommendation engine through a RESTful API with FastAPI. The project is open-source and available at `https://github.com/Katho162/nlmt-v2` under a CC BY 4.0 license.

## 1 Introduction

The pursuit of learning new languages is a common goal for many individuals, driven by personal, professional, and cultural motivations. However, choosing which language to learn next can be a daunting task. In this paper, we address this challenge by proposing NLMT-v2, a recommender system that provides personalized language suggestions. NLMT stands for Next-Gen Language Management Tool. Our system leverages the power of deep learning to model user language profiles and recommend languages that are likely to be of interest to the user.

## 2 Related Work

Recommendation systems are a well-established field of research, with applications ranging from e-commerce to content streaming. Traditional approaches include collaborative filtering and content-based filtering. More recently, deep learning models, such as autoencoders, have shown great promise in learning complex user preferences and item representations. Our work builds upon these advancements by applying a Variational Autoencoder to the domain of language recommendation.

## 3 Methodology

Our approach consists of three main components: data preprocessing, the VAE model architecture, and the loss function.

### 3.1 Data Preprocessing

The raw data is sourced from ‘roles.csv‘, which contains entries detailing user language skills across various proficiency levels. The initial step involves loading this CSV data into a pandas DataFrame. To prepare the data for the VAE, we transform the multi-level proficiency information into a binary representation. For each unique language identified across the dataset (e.g., "Studying English", "Fluent Spanish", "Native German"), we create a dedicated column. If a user has any recorded proficiency (Studying, Fluent, or Native) in a particular language, the corresponding entry in the binary representation is set to 1; otherwise, it is 0. This process effectively creates a one-hot encoded vector for each user, representing their known languages. The resulting DataFrame is then converted to ‘numpy.float32‘ and subsequently to a ‘torch.tensor‘ for compatibility with PyTorch.

## 3.2 Model Architecture

We employ a Variational Autoencoder (VAE) to learn a latent representation of user language profiles. The VAE consists of two main parts: an encoder and a decoder.

### 3.2.1 Encoder

The encoder, $q_\phi(z|x)$, is responsible for mapping the high-dimensional input user language profile $x \in \mathbb{R}^D$ (where $D$ is the number of unique languages) to a lower-dimensional latent space. Our encoder is a multi-layer perceptron (MLP) with two hidden layers and ReLU activation functions. Specifically, the architecture is as follows:

- Input Layer: $D$ neurons (number of languages)
- Hidden Layer 1: 512 neurons, followed by ReLU activation
- Hidden Layer 2: 256 neurons, followed by ReLU activation

From the output of the second hidden layer, two separate linear layers project to the mean ($\mu$) and the log-variance ($\log(\sigma^2)$) of the latent distribution. These represent the parameters of a Gaussian distribution in the latent space, $z \in \mathbb{R}^L$, where $L$ is the latent dimension.

### 3.2.2 Decoder

The decoder, $p_\theta(x|z)$, takes a sample $z$ from the latent distribution and aims to reconstruct the original user language profile $x$. It mirrors the encoder's architecture:

- Input Layer: $L$ neurons (latent dimension)
- Hidden Layer 1: 256 neurons, followed by ReLU activation
- Hidden Layer 2: 512 neurons, followed by ReLU activation
- Output Layer: $D$ neurons (number of languages)

The output layer typically uses a sigmoid activation function (implicitly handled by 'BCEWithLogitsLoss') to produce probabilities for each language.

### 3.2.3 Reparameterization Trick

To allow for backpropagation through the sampling process from the latent distribution, we utilize the reparameterization trick. Instead of directly sampling $z \sim \mathcal{N}(\mu, \sigma^2)$, we sample $\epsilon \sim \mathcal{N}(0, I)$ and then compute $z$ as:

$$z = \mu + \epsilon \cdot \sigma \tag{1}$$

where $\sigma = \exp(0.5 \cdot \log(\sigma^2))$ is the standard deviation. This reparameterization makes the sampling process differentiable with respect to $\mu$ and $\sigma$.

### 3.2.4 Language Embeddings (Optional)

An optional 'nn.Embedding' layer is included, which can be used to incorporate specific embeddings for known languages into the latent space. This can potentially enrich the latent representation by providing additional context for the languages a user already knows.

## 3.3 Loss Function

The VAE is trained by minimizing a loss function that balances two objectives: accurate reconstruction of the input and regularization of the latent space. This loss function is given by:

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)) \tag{2}$$

where:

- The first term, $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$, is the reconstruction loss, which measures how well the decoder reconstructs the input $x$ from a latent sample $z$.

- The second term, $D_{KL}(q_\phi(z|x)||p(z))$, is the Kullback-Leibler (KL) divergence, which acts as a regularizer by forcing the approximate posterior $q_\phi(z|x)$ to be close to a prior distribution $p(z)$, typically a standard normal distribution $\mathcal{N}(0, I)$.

For our binary input data, the reconstruction loss is implemented using Binary Cross-Entropy (BCE) with logits:

$$\text{BCE}(x, \hat{x}) = -\sum_{i=1}^{D}[x_i \log(\sigma(\hat{x}_i)) + (1 - x_i) \log(1 - \sigma(\hat{x}_i))] \tag{3}$$

where $x$ is the true input, $\hat{x}$ is the reconstructed output (logits), and $\sigma$ is the sigmoid function.

The KL divergence for two Gaussian distributions, $q_\phi(z|x) = \mathcal{N}(\mu, \sigma^2)$ and $p(z) = \mathcal{N}(0, I)$, is given by:

$$D_{KL}(\mathcal{N}(\mu, \sigma^2)||\mathcal{N}(0, I)) = 0.5 \sum_{j=1}^{L}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) \tag{4}$$

Minimizing this term encourages the latent space to be well-behaved and prevents the encoder from collapsing into a trivial solution.

# References


[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[2] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[4] Xiangnan He, Lihan Liao, Hanwang Zhang, Yongfeng Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.