Enhancing Latent Space Clustering in Multi-filter Seq2Seq Model: A Reinforcement Learning Approach

Yunhao Yang
Zhaokun Xue

1 University of Texas at Austin, Department of Computer Science
110 Inner Campus Drive
Austin, Texas 78705
yunhaoyang234@utexas.edu

Zhaokun Xue
xuezhaokun@utexas.edu

Abstract
In sequence-to-sequence language processing tasks, sentences with heterogeneous semantics or grammatical structures may increase the difficulty of convergence while training the network. To resolve this problem, we introduce a model that concentrates the each of the heterogeneous features in the input-output sequences. Build upon the encoder-decoder architecture, we design a latent-enhanced multi-filter seq2seq model (LMS2S) that analyzes the latent space representations using a clustering algorithm. The representations are generated from an encoder and a latent space enhancer. A cluster classifier is applied to group the representations into clusters. A soft actor-critic reinforcement learning algorithm is applied to the cluster classifier to enhance the clustering quality by maximizing the Silhouette score. Then, multiple filters are trained by the features only from their corresponding clusters, the heterogeneity of the training data can be resolved accordingly. Our experiments on semantic parsing and machine translation demonstrate the positive correlation between the clustering quality and the model's performance, as well as show the enhancement our model has made with respect to the ordinary encoder-decoder model.

Introduction
A sequence to sequence (seq2seq) model maps a fixed-length input with a fixed-length output where both the input and output can have variable length. An encoder-decoder model is one approach to solving sequence-to-sequence prediction problems by using recurrent neural networks (RNN) as the underlying component. A challenge for the seq2seq models is that the training data with heterogeneous features may increase the difficulty of convergence. One way to resolve heterogeneity in the dataset is applying representation learning to analyze the heterogeneous features.

Representation Learning
Representation learning is the technique of automatically extracting representations from raw data and thus make it computationally feasible to process the data. Because different representations can differ in the explanatory factors of variation behind the data, the success of a given algorithm depend on the quality of the extracted representation [Bengio, Courville, and Vincent 2013].

The representation of a sentence is the output of the encoder, lying in the latent space. Latent space refers to an abstract multi-dimensional space containing feature values that cannot be interpreted directly. Instead, the values in the latent space are meaningful internal representations that are encoded from some externally observed data. The latent space intends to provide data that is machine recognizable/understandable to a computer through a quantitative spatial representation or modeling.

Latent space can be considered as a set of hidden features or internal representations over the input data, thus a small variation in the latent space may be caused by a large difference in the input data. The process of learning a latent space is to tune the parameters within the encoder and decoder to obtain a better representation of the data. A well-learned latent space would help the machine learning model better understanding the features and easily identifying the principal components of the data.

To resolve the heterogeneity in the dataset, we apply representation learning to identify and categorize the heterogeneous features. Therefore, the quality of the representations in the latent space is important. Reinforcement learning is an approach of enhancing latent space representations.

Reinforcement Learning
Reinforcement learning (RL) is an area of machine learning that develops approximate methods for solving dynamic optimization problems. Unlike other machine learning algorithms, the learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them [Sutton and Barto 2018]. Due to this fact, RL can be used in post processing enhancements by setting appropriate action and reward function. RL algorithms typically consist of an agent that takes actions. The action is determined based on the environment observations and the policy. The new action updates the state of the environment and generates new observations. This loop is continuing until reaching the terminating state (eg. game over) or the maximum learning steps.
allowed. The rewards are given based on the outcomes of the actions. The learning policy is optimized for maximizing the cumulative reward. We are planning to apply an RL algorithm to improve the latent space representations by defining appropriate action and reward functions. Hence we can better analyze the various features in the dataset.

Related Research

Encoder-decoder network is one of the classical models for solving sequence to sequence tasks. In machine translation, an encoder on the language can create inner representations on the latent space and a decoder then unveils these coding into desired output.

Research has shown that latent space representations can exploit the features and better preserve the key attributes of the raw data (Yang and Whinston 2021). Thus the learned representations can be used in the sequence to sequence tasks for feature analysis.

Since the latent space representations can better demonstrate the features of the raw input, the representations can also be divided into different groups where each group consists of homogeneous data. There are some researches exploring hierarchical structures on the latent space to obtain richer representations (Bouchacourt, Tomioka, and Nowozin 2018). There are several other researches performing clustering or learning a mixture model within the latent space encoding, including Hard K-means Clustering (Yang et al. 2017), Soft K-means Clustering (Jabi et al. 2019), Gaussian Mixture Model (VAR: Dilokthanakul et al. 2016).

These researches have shown that dividing latent space can improve the quality of the representations in some aspects. Following this idea, a Multi-filter Gaussian Mixture Autoencoder (Yang and Xue 2021) is introduced for seq2seq tasks. It improves the encoder-decoder model by dividing latent space and using multiple decoders. This research has also demonstrate the positive relationship between the quality of latent space clustering and model’s overall performance.

Reinforcement learning can be a technique for improving the quality of latent space clustering. Research has suggested that reinforcement learning can be used to achieve more appropriate clusters (Barbakh and Fyte 2007) by designing proper reward functions.

Multi-filter Network Architecture

The network consists of an encoder and a dummy decoder from an ordinary encoder-decoder model; a latent-space enhancer (multi-layer perceptron) that takes in the final hidden states from the encoder as inputs and projects them on the latent space; a cluster classifier for determining which cluster the latent space representation belongs to; certain number of decoders for various sets of features.

Latent Enhanced Encoder-Decoder Model

The latent enhanced encoder-decoder model consists of an encoder $R$, a dummy decoder $Q_0$, and an enhancer $T$ that connects to the encoder and projects the encoder outputs into the latent space. The encoder takes in a sequence of inputs $x$, then returns a sequence of outputs and a final hidden state $h$.

$$h = h_{|x|}, \quad \text{where} \quad \overline{x}_i, (h_i, c_i) = R^i(x_i)$$

where $R_i$ is the $i^{th}$ recurrent cell, $\overline{x}_i$, $h_i$, and $c_i$ are the $i^{th}$ output, hidden state, and cell state, respectively.

The encoder consists of an embedding layer that transforms the input text into an embedded vector, and a bidirectional long short-term memory classifier that generates the outputs and hidden states.

The final hidden states generated by the encoder are the representations of the input-output pairs. Space where the representations lying in is denoted as the hidden space $Z$. Then, an enhancer is applied in $Z$ for enhancing the representations.

$$r_e = T(h)$$

The enhancer is a multi-layer perception that transforms the representation $h$ from $Z$ to a new space $Z_F$. The space $Z_F$ is called latent space. The enhanced representation, denoted as $r_e$, is feed as the input of the decoder. In addition, the $r_e$ is also feed as the input of the cluster classifier $C$ for cluster assignment.

The dummy decoder $Q_0$ generates outputs from the enhanced representations $r_e$. The decoder consists of a long short-term memory network, a linear layer that converts the output from hidden dimension to output dimension, and a LogSoftmax layer. A dot product attention mechanism (Luong, Pham, and Manning 2015) is appended to the decoder, after the LSTM network. The attention weights are computed using the hidden state $\overline{h}$ generated by the LSTM network and the sequence of the encoder’s outputs $h$.

To optimize the parameters in this encoder-decoder model, a Negative Likelihood Loss is utilized to update gradients. The outputs of the decoder are gone through a LogSoftmax layer and then used to compute the Negative Likelihood Loss:

$$\mathcal{L}(X, y) = (l_1, ..., l_N)^T, \quad l_n = -w_y \log \frac{e^{r_{y_n}}}{\sum e^{r_{yn}}}$$

(1)

where $T$ is the output tensor and $y$ is the label tensor.

The loss is used to optimize all the parameters in the encoder, decoder, as well as the enhancer. After the model is fine-tuned, the encoder and enhancer are preserved for generate the enhanced representations.

Cluster Assignment

To assign data into clusters, a cluster classifier $C$ can be constructed. The classifier $C$ takes the enhanced representation as input and output a probability vector $v_C$ for cluster assignment. The classifier $C$ consists of two linear layers that transform the input from hidden dimension (space $Z_F$) to $n$-dimension, where $n$ is the number of clusters. Then, a softmax layer is added at the end of the classifier $C$. At this stage, the output of $C$ is considered as a probability vector $v_C$ that consists of the probabilities of assigning to each cluster. Hence if $v_C$ is the probability vector of sample representation $h_k$, then
Figure 1: the encoder-decoder model on the top consists of an encoder $R$ and a dummy decoder $Q_0$. The encoder-decoder model learns the representations of the input-output pairs. The representation is the final hidden state $h$ from the encoder lying on the hidden space $Z$. An enhancer $T$ is applied to the hidden space $Z$ to project the hidden state $h$ on the latent space $Z_T$. The transformed vector on space $Z_T$ is called enhanced representation $r_e$. Following the enhancer, there is cluster classifier $C$ assigns the enhanced representation $r_e$ into a cluster. Each cluster $c_i$ is corresponding to a filter $Q_i$, which is only trained and used to decode the vectors from its correspondence cluster.

We can take the cluster $c_j$ with the highest probability as the cluster the sample representation $x$ belongs to:

$$h_x \in c_j, \text{ where } j = \text{arg max}(v_p).$$

Then, the representation $h_x$ can be fed into the filter $Q_j$ that corresponding to $c_j$. Hence the output can be constructed.

The parameters in $C$ will not be adjusted by any loss computed from the decoder’s outputs. Instead, we applied a reinforcement learning algorithm to optimize the parameters in $C$, which is discussed in the later section.

**Multi-filter Decoding**

The model has a certain number of filters $Q_1, ..., Q_n$ for decoding purposes, $n$ is the total number of filters. Each filter shares an identical structure with the dummy decoder $Q_0$ in the encoder-decoder model. The filters are designed for constructing the target outputs from the enhanced representations in the space $Z_T$.

Suppose the data record is assigned to the $i^{th}$ cluster by the cluster classifier. For each RNN cell in the filter $Q_j$, it takes in the output, hidden state, and cell state from the previous RNN cell and generate new output and states.

$$x_i, (\vec{h}_i, \vec{c}_i) = Q_j(x_{i-1}, (\vec{h}_{i-1}, \vec{c}_{i-1}))$$

In the training stage, the cluster classifier $C$ has divided the data into $n$ clusters, where each cluster. Based on the clustering result, we set up $n$ filters that each cluster is associated with one filter. All the filters consist of identical network structures, but their parameters are tuned independently. A filter $Q_i$ is trained only using the data assigned to its corresponding cluster $c_i$. The loss used to train the filters is the Negative Likelihood Loss $1$ The losses for each filter are computed separately and their gradients are updated separately as well. Through the entire training procedure, the parameters in the encoder $R$ and the transformer $T$ are fixed, thus they will not be updated.

In the evaluation stage, the input data is first encoded and projected on the latent space $Z_T$ by the encoder and enhancer. Then, the cluster classifier $C$ classifies which cluster the input data belongs to. The sample classified into the cluster $c_i$ will go through the corresponding filter $Q_i$ for constructing the output.

In this way, we can construct a set of feature-specific filters that resolves the heterogeneity in the dataset.

**Latent-Enhancing Algorithm**

To optimize the latent space, we introduce a Soft Actor Critic (SAC) [Haarnoja et al. 2018] reinforcement learning algorithm that is utilized for optimizing the trainable pa-
The random variable $X$ is defined by

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \quad (2)$$

The SAC uses entropy regularization by maximizing the trade-off between expected return and a discounted entropy term of current policy. This trade-off closely relates to the exploration-exploitation trade-off in that increased entropy leads to more exploratory search and increases learning rate of future timesteps. A policy with decent entropy leads to more exploratory search and increases the reward function by $R$, the regularization term of the policy gets trapped in a local optimum (Achiam 2018).

If we denote the policy by $\pi$, the discount factor by $\gamma$, the reward function by $R$, the regularization term of the entropy can be added to the target value for the policy as:

$$\pi^* = \arg \max_\pi \mathbb{E} \sum_{t=0}^{\infty} \gamma^t (R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot|s_t))) \quad (3)$$

Similarly, there is the entropy regularization term added to the action value $Q^\pi$ for every timestep excluding the initial timestep where action has been selected already:

$$Q^s_t(s, a) = \mathbb{E} \sum_{s'} \sum_{a'} \gamma Q(s', a') + \alpha \mathbb{E} \sum_{t=1}^{\infty} \gamma^t H(\pi(\cdot|s_t)) | s_0 = s, a_0 = a \quad (4)$$

the Bellman equation for $Q^s$ is

$$Q^s_t(s, a) = \mathbb{E} \sum_{s', a'} \gamma (Q^s_t(s', a') + \alpha H(\pi(\cdot|s'))) \quad (5)$$

SAC learns a policy $\pi_0$ and two Q-functions $Q_{\phi_1}, Q_{\phi_2}$ to reduce maximization bias. Both Q-functions are learned with mean-squared Bellman error (MSBE) loss. The two Q-functions share the same target Q-networks, which is updated by polyak averaging of the Q-network parameters during training. The target term would also include an entropy regularization term (Achiam 2018).

### SAC Application on Latent Space Enhancement

The reward function is based on the Silhouette Coefficient $S_c$:

$$R = k \cdot S_c + b \quad (6)$$

where $k$ and $b$ are the user-defined constants. To maximize the reward, the RL model learns actions that adjust the trainable weights in the classifier $C$. The action function is defined as:

$$T_i = \tilde{a} \cdot T'_i \quad (7)$$

where $\tilde{a}$ is the action, $T'_i$ is the set of old parameters of the $i^{th}$ layer of $T$, $T_i$ is the set of updated parameters of the $i^{th}$ layer of $T$.

The reinforcement learning is applied after the encoder and the dummy decoder is fine tuned, prior to training the filters. A target Silhouette Coefficient $S_{\text{target}}^c$ is set as the terminate state of the RL model. The learning process immediately stop once the model reaches the maximum steps allowed, or the Silhouette Coefficient $S_c$ reaches the target score: $S_c \geq S_{\text{target}}^c$. This reinforcement learning model is expected to enhance the final results by improving the quality of clustering. After the RL algorithm improves the clustering quality, we can start training the multiple filters.

### Experiments

The experiments are going to prove the effectiveness of the multi-filter architecture. To achieve this, we conduct several comparative experiments. These experiments compare the performance from our latent-enhanced multi-filter model with the traditional encoder-decoder model, as well as some other baselines. The experiments are conducted on two of the classical sequence to sequence tasks—semantic parsing and machine translation.

In addition to showing the performance improvement, our experiments also demonstrate the positive correlation between the model’s performance the quality of the latent space clustering. Hence we can show the necessity of our latent-enhancing reinforcement learning algorithm.

All the experiments are conducted using a 2GHz Quad-Core Intel Core i5 CPU with 16 GB memory. The code is written in Python 3.7 and ran under a Linux system.

### Performance Examination on Semantic Parsing

This set of experiments uses the Geo-query dataset (Zelle and Mooney 1996) that performs semantic parsing on geographical questions.

There are two metrics for evaluation: the token level accuracy, and the denotation accuracy. The token level accuracy is based on simple token-level comparison against the reference logical form. And the denotation accuracy is the percentage of denotation match, as in (Jia and Liang 2016) and (Liang, Jordan, and Klein 2011).

In this comparative experiment, the models are trained and tested under the same environment settings. The training set consists of 480 samples and the development set consists of 120 samples for testing purpose. All the models are tuned in 10 epochs, with word-embedding dimension 150, hidden dimension (which is equal to the latent space dimension) 200, learning rate 0.001, dropout rate 0.2, and the LSTM network in the encoder is set to bidirectional. To enhance the latent space clustering, we utilize multi-layer perceptron (MLP) as the policy for SAC. In the SAC algorithm, we set the maximum number of learning step to 500, $k = 100$ and $b = 25$ in Equation 6.

Target Silhouette score $S_{\text{target}}^c = 0.55$, and keep the default settings from the SAC implementation from Stable-Baselines3 (Raffin et al. 2019).

To determine the optimal number of filters, we first train the encoder and the enhancer to obtain the latent space representations of the training data. Then, we apply
the soft actor-critic algorithm to tune the cluster classifiers with two, three, and four clusters separately. By comparing the clustering outcomes and the Silhouette score, we can find the best number of filter for this semantic parsing problem.

We choose a 2-filter LES2S model to train and test on the Geo-query dataset. Our model achieves the best accuracies compare to all the baselines. More importantly, our model significantly improves both accuracies compare to the ordinary encoder-decoder model. This outcome proves that the multi-filter architecture is able to resolve heterogeneous features in the dataset. So that we show the necessity of the multi-filter architecture.

**Performance Examination on Machine Translation**

To justify the performance of our model in machine translation tasks, we use the Multi30k English-French dataset (Elliott et al. 2016). This dataset is split to 29,000 training data records and 1,000 testing data records.

In the training stage, the model is tuned in 20 epochs, with word-embedding dimension 150, hidden dimension and latent space dimension 200, learning rate 0.001, dropout rate 0.2, and the LSTM network in the encoder is set to bidirectional. In the SAC algorithm for latent enhancing, we utilize multi-layer perceptron (MLP) policy, set the max number of learning step to 500, $k = 75$ and $b = 15$ in Equation 6 target Silhouette score $S_{\text{target}} = 1$, and keep the default settings from Stable-Baselines3 (Raffin et al. 2019).

To determine the best number filters, we first fine-tuned the encoder and the enhancer thus we can obtain the latent space representations of the training data. Then, we apply the SAC reinforcement learning algorithm to train the cluster classifiers for two, three, and four clusters separately. By observing the clustering results and the Silhouette score, we can determine the optimal number of filters for this task. Figure 3 presents the clustering results of the latent space representations. We can observe that the cluster classifier generates two clusters regardless of the number of filters assigned to.

Because the SAC algorithm optimizes the parameters of the cluster classifier to maximizing the Silhouette score. The reinforcement learning model learns that the Silhouette score can be maximized when the latent space representations are separated into two clusters. Therefore, even we increase the number of clusters, the classifier will form empty clusters so that there are still two "real" clusters. Then, we can use 2-filter LES2S model for the following experiments.

To evaluate and compare the results, BLEU (bilingual evaluation understudy) score is utilized. BLEU score is a quick and inexpensive metric for evaluating the quality of text which has been machine-translated from one natural language to another (Papineni et al. 2002). To compute BLEU score, first we use the test corpus' reference length $r$ and candidate translation corpus' length $c$ to calculate the brevity penalty:

$$BP = \exp(1 - r/c)$$

then, the BLEU score is computed by:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$
where $p_n$ is the N-gram precision. In the experiments, we use $N = 4$ and uniform weights $w_n = 1/4$. The BLEU scores are presented in Table 2.

Table 2: Performance comparison between the original encoder-decoder model, our LMS2S model, and several other baselines in machine translation task. The table shows the average BLEU score across five identical experiments.

| Model                          | BLEU |
|--------------------------------|------|
| Baseline (text-only NMT)       | 44.3 |
| SHEF_ShelfClassProj_C (Elliott et al. 2017) | 43.6 |
| CUNI Neural Monkey Multimodel MT_C (Helcl and Libovicky 2017) | 49.9 |
| LIUMCVC_NMT_C (Caglayan et al. 2017) | 53.3 |
| DCU-ADAPT MultiMT C (Elliott et al. 2017) | 54.1 |
| LES2S                          | 55.7 |

Table 2 lists the BLEU scores of the baseline models and our LES2S model. Among these results, our model achieves the best performance. Moreover, the comparative experiment between the ordinary encoder-decoder model and our model also shows the effectiveness of the multi-filter architecture.

Latent Space Clustering Enhancement

In both tasks, we use the soft actor-critic (SAC) algorithm to enhance the clustering quality. The results have shown that the SAC algorithm is able to improve the Silhouette score (generate better clusters).

To explore how the Silhouette score affects the model’s performance, we train a set of LES2S models for each task. Every model has trained under the same hyper-parameter settings: 20 epochs for Multi30k and 10 epochs for Geoquery, word-embedding dimension 150, hidden dimension and latent space dimension 200, learning rate 0.001, dropout rate 0.2. And the filter number is set to 4.

By setting the learning steps in the SAC algorithm to 10, 20, 30, 50, 100, 200, 500, respectively, we can get clustering results in terms of Silhouette scores. Then, we compare the models’ performances with Silhouette scores to show the significance of the latent space clustering.

Figure 4 shows the model’s performance vs. Silhouette scores. We can observe that the Silhouette score is positively correlated to both of the evaluation metrics of the two tasks. These results show that we can improve the model’s performance by optimizing the latent space clustering. Hence we have proved the significance of the latent space clustering.

Conclusion

We introduce the Latent-Enhanced multi-filter Seq2Seq model that resolves the problem of heterogeneous features in the dataset. Our model is able to achieve better performance by concentrating heterogeneous features simultaneously using the multi-filter architecture.

Moreover, we have shown the enhancement of the model’s performance while the reinforcement learning algorithm improves the clustering quality. Due to the assumption that data with similar features would be clustered together, better clustering quality leads to more accurate feature analysis. The filters can be trained by a set of records that contain similar features. Hence the overall performance can be improved.
References

Achiam, J. 2018. Spinning Up in Deep Reinforcement Learning. GitHub repository.

Barbakh, W.; and Fyfe, C. 2007. Clustering with Reinforcement Learning. In IDEAL.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828.

Bose, S.; and Huber, M. 2016. Semi-Unsupervised Clustering Using Reinforcement Learning. In FLAIRS Conference.

Bouchacourt, D.; Tomioka, R.; and Nowozin, S. 2018. Multi-Level Variational Autoencoder: Learning Disentangled Representations From Grouped Observations. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, 2095–2102. AAAI Press.

Caglayan, O.; Aransa, W.; Bardet, A.; García-Martínez, M.; Bougares, F.; Barrault, L.; Masana, M.; Herranz, L.; and van de Weijer, J. 2017. LIUM-CVC Submissions for WMT17 Multimodal Translation Task. *Proceedings of the Second Conference on Machine Translation*.

Dilokthanakul, N.; Mediano, P. A.; Garnelo, M.; Lee, M. C.; Salimbeni, H.; Arulkumaran, K.; and Shanahan, M. 2016. Deep unsupervised clustering with gaussian mixture variational autoencoders. arXiv preprint arXiv:1611.02648.

Dong, L.; and Lapata, M. 2016. Language to Logical Form with Neural Attention. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Elliott, D.; Frank, S.; Barrault, L.; Bougares, E.; and Specia, L. 2017. Findings of the Second Shared Task on Multimodal Machine Translation and Multilingual Image Description. *Proceedings of the Second Conference on Machine Translation*.

Elliott, D.; Frank, S.; Sima’an, K.; and Specia, L. 2016. Multi30K: Multilingual English-German Image Descriptions. *Proceedings of the 5th Workshop on Vision and Language*.

Ge, R.; and Mooney, R. J. 2005. A Statistical Semantic Parser That Integrates Syntax and Semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL ’05, 9–16. USA: Association for Computational Linguistics.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 1861–1870. PMLR.

Helcl, J.; and Libovický, J. 2017. Neural Monkey: An Open-source Tool for Sequence Learning. *The Prague Bulletin of Mathematical Linguistics*, 5–17.

Jabi, M.; Pedersoli, M.; Mitiche, A.; and Ayed, I. B. 2019. Deep clustering: On the link between discriminative models and k-means. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Jia, R.; and Liang, P. 2016. Data Recombination for Neural Semantic Parsing. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Kate, R. J.; and Mooney, R. J. 2006. Using String-Kernels for Learning Semantic Parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 913–920. Sydney, Australia: Association for Computational Linguistics.

Li, M.; Jordan, M.; and Klein, D. 2011. Learning Dependency-Based Compositional Semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 590–599. Portland, Oregon, USA: Association for Computational Linguistics.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective Approaches to Attention-based Neural Machine Translation. arXiv:1508.04025.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics.

Raffin, A.; Hill, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; and Dormann, N. 2019. Stable Baselines3. https://github.com/DLR-RM/stable-baselines3.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book. ISBN 0262039249.

Wong, Y. W.; and Mooney, R. 2006. Learning for Semantic Parsing with Statistical Machine Translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, 439–446. New York City, USA: Association for Computational Linguistics.

Yang, B.; Fu, X.; Sidiropoulos, N. D.; and Hong, M. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, 3861–3870. PMLR.

Yang, Y.; and Whinston, A. 2021. Identifying Mislabeled Images in Supervised Learning Utilizing Autoencoder. arXiv:2011.03667.

Yang, Y.; and Xue, Z. 2021. Representation Learning in Sequence to Sequence Tasks: Multi-filter Gaussian Mixture Autoencoder. arXiv:2105.08840.

Zelle, J. M.; and Mooney, R. J. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI’96, 1050–1055. AAAI Press. ISBN 026251091X.