Correcting Linguistic Training Bias in an FAQ-bot using LSTM-VAE

Mayur Patidar, Puneet Agarwal, Lovekesh Vig, and Gautam Shroff

TCS Research, New Delhi.
{patidar.mayur,puneet.a,lovekesh.vig,gautam.shroff}@tcs.com

Abstract. We consider an automated assistant that has been deployed in a large multinational organization to answer employee FAQs. The system is based on an LSTM classifier that has been trained on a corpus of questions and answers carefully prepared by a small team of domain experts. We find that linguistic training bias creeps into the manually created training data due to specific phrases being used, with little or no variation, which biases the deep-learning classifier towards incorrect features. Further, often the FAQs as envisaged by the trainers are in fact incomplete, and transferring linguistic variations across question-answer pairs can uncover new question classes for which answers are missing. In this paper we demonstrate that an LSTM-based variational auto-encoder (VAE) can be used to automatically generate linguistically novel questions, which, (a) corrects classifier bias when added to the training set, (b) uncovers incompleteness in the set of answers and (c) improves the accuracy and generalization abilities of the base LSTM classifier, enabling it to learn from smaller training sets.

Keywords: Variational Autoencoders, Classification, Language Modelling, FAQ-bot

1 Introduction

The recent successes of deep learning techniques for NLP have seemingly obviated the need for careful crafting of features based on linguistic properties. Deep models purportedly can learn the features required for a task directly from data, provided it comes in sufficient volume and variety, typically obtained in 'natural' settings such as the web. However, in practical applications as we shall describe, training data is far more limited and is often created manually in a curated manner specifically for the task at hand. We show that linguistic training bias often creeps into such training data and degrades the performance of deep models. Further, manual curation can often result in incomplete task specification due to insufficient linguistic variation in the training data.

We have created and deployed a chatbot for the use of employees in our large organization, which answers human resource (HR) policy related questions in natural language. A deep-learning model in the form of an Long short-term memory (LSTM) [11] classifier was used for mapping questions to classes, with

In: P. Cellier, T. Charnois, A. Hotho, S. Matwin, M.-F. Moens, Y. Toussaint (Eds.): Proceedings of DMNLP, Workshop at ECML/PKDD, Skopje, Macedonia, 2017.

each class having a manually curated answer. A small team of HR officers (5 members) manually created the training data used to train this classifier.

We observed that the chosen deep-learning technique performed better than traditionally known approaches on a given test set (also created by the same HR officers). However, the deep model also sometimes classified a user query using clearly irrelevant features (i.e., words). For example, the question "*when my sick leave gets credited ?*" was classified into a category related to '*Adoption Leave*' resulting in a completely irrelevant answer. Upon analysis we discovered that this happens mainly because the words surrounding 'sick leave' (e.g., "gets") in the query occurred more often in the training data for '*Adoption Leave*'. As a result, if such words occur in users' query, the model ignores other important words (such as 'sick leave') and classifies the query into incorrect class, based on such words. We refer to this phenomenon as *linguistic training bias*, which creeps in merely because templates used for different classes are not exhaustive. Such examples led us to fear that such a system would not generalize well when exposed to real users.

More generally, relying on human curation often results in such linguistic training biases creeping into the training data, since every individual has a specific style of writing natural language and uses some words in specific context only. Deep models end up learning these biases, instead of the core concept words of the target classes.

In order to correct these biases we automatically generate meaningful sentences using a generative model, and then use them for training the classification model after suitable human annotation. We use a Variational Autoencoder (VAE) [15] as our generative model for generating novel sentences and utilize a Language Model (LM) [18] for selecting sentences based on likelihood. We model the VAE using RNNs comprising of LSTM units. As we shall demonstrate in Section 6, this approach gives us a gain of about 9% in accuracy of the classification, when tested with core concept words only.

Further, the HR officers created the classes and corresponding questions based on their experience of what are the frequently asked questions by various employees of the organization. However, when a chatbot is available, users tend to ask extra questions not asked otherwise. It is therefore imperative to have broader coverage of such classes. If we could show novel classes of questions to the HR officers, generated automatically, it becomes easier for them to accept or reject the proposed classes rather than having to imagine all such possibilities. As we demonstrate in Section 6.2 our approach generated many new classes that were accepted by the HR officers for deployment.

More specifically, the use of a deep generative model to augment training sentences resulted in the following important benefits, which form the key contributions of this paper:

1. Augmenting training data with automatically generated sentences is able to correct over-fitting due to linguistic training bias. To show this we present results on 'concept words', indicating potentially better generalization.

2. The newly generated sentences sometimes belonged to completely new classes not present in the original training data: 33 new classes were found in practice.
3. Augmenting training data with automatically generated sentences results in an improved accuracy (2%) of the deep-learning classifier.

We also present an improved approach for training VAEs: *weighted cost annealing*, based on our experience of training LSTM-VAE on a real-life dataset.

2 Related Work

VAEs have found widespread usage in image generation [9, 8], in text modeling [3, 17] and in recent work on sketch drawing [10]. Researchers have also used VAEs for semi-supervised learning [14, 26]. [3] have used VAE and showed positive results for novel sentence generation, missing words imputation, and have shared techniques for efficiently training VAEs. They also observed negative results on language modeling task. Dilated CNNs (Convolutional Neural Networks) and combinations of both RNN and CNN have been used for modeling the VAE encoder and decoder in [27, 23]. We have used a VAE architecture similar to [3] with some modification in the training procedure, referred here as *weighted cost annealing*.

As shown in [4, 7], deep learning models misclassify the pertubed input samples with high confidence including adversarial examples. Several methods have been proposed by researchers for training a robust model, for example, dropout [24] and training with noisy samples [16]. In the text domain [16] have shown that training data augmented with noisy training sentences leads to better classification performance. They have used wordnet [19], counter-fitting method as described in [20] and deep linguistic parser [5], sentence compression techniques for generating semantically and syntactically noisy training sentences, respectively. Unlike [16], we have augmented the training data with sentences generated by the LSTM-VAE. We have observed that the sentences generated by the LSTM-VAE are not always semantically and syntactically correct, i.e., it also acts as a noisy sentence generator. Importantly, we have observed improvement in classification accuracy after adding the generated sentences.

3 Problem Description

We assume that a dataset of frequently asked questions for building a chatbot comprises of sets of semantically similar questions $Q_i = \{q_1, ..., q_{n_i}\}$ and their corresponding answer a_i. A set of such questions Q_i and corresponding answer a_i are collectively referred to as a query set $s_i = \{Q_i, a_i\}$. Questions of a query set s_i are represented as $Q_i = Q(s_i)$. We assume that the dataset D comprises of many such query sets, i.e., $D = \{s_1, ..., s_m\}$. In our chatbot implementation, given a user's query q, the objective is to select the corresponding query set s

via a multi-class classification model, such that corresponding answer a could be shown.

Given all the questions in the training data D, $Q = \cup Q(s_i), \forall s_i \in D$, we intend to generate new questions Q' using LSTM-VAE. Some of the questions in Q' are semantically similar to one of the query sets of D, while the remaining questions do not belong to any of the existing classes. Using these newly generated queries Q', we present the results of our study comprising of a) analysis of the new query-sets s' and results of the review of these query sets by HR domain experts, b) comparison of classifiers trained on Q, and Q_{new} $(= \cup\{Q, Q'\}, \forall q \in Q_{new} \exists i, \text{s.t. } q \in s_i)$ on core concept words rather than the full natural language questions, and c) also analyze how the accuracy of a classification model improves.

4 Background

Recurrent neural networks (RNNs) are known for modeling sequential dependencies in the input, for e.g. in language modeling tasks where prediction of a word depends on the previous words in the sentence. Generally vanilla RNNs do not handle long term dependencies in the input due to the vanishing gradient problem [2], which is overcome by using RNNs with LSTM cells in the hidden layers. In this section, we describe the details of various components of our approach, comprising of an LSTM-VAE and a language model (LM).

4.1 Sequence Autoencoder

Autoencoders are neural networks used for learning lower dimension representations of data, referred to as encoding or latent representation ($\mathbf{z}$). Autoencoders are comprised of two components: an encoder ($\mathbf{z} = f_{enc}(\mathbf{x})$) and a decoder ($\hat{\mathbf{x}} = f_{dec}(\mathbf{z})$), where the encoder transforms the input $\mathbf{x}$ to $\mathbf{z}$ and the decoder tries to reconstruct the $\mathbf{x}$ from $\mathbf{z}$. Autoencoders are trained by minimizing the reconstruction error of the decoded data $\hat{\mathbf{x}}$ with respect to input $\mathbf{x}$, i.e., $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$.

The choice of encoder and decoder networks generally depends on the input data. A *sequence autoencoder* uses RNNs to encode (as well as to decode) sequential input data, e.g., a sentence. Latent representations of data, learned by the sequence autoencoder, can also be used for classification. [6] uses RNNs with LSTM units for both encoder and decoder of the sequence autoencoder. The sequence autoencoders cannot be used for generation of novel samples as they do not enforce a prior distribution on the latent representations. Also, as shown by [3], encodings learned by the sequence autoencoder are unable to capture semantic features.

4.2 Variational Autoencoder

The VAE is a generative model which unlike sequence autoencoders, is comprised of a probabilistic encoder ($q_\phi(\mathbf{z}|\mathbf{x})$, *recognition model*) and a decoder ($p_\theta(\mathbf{x}|\mathbf{z})$,

generative model). The posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is known to be computationally intractable. VAE approximates $p_\theta(\mathbf{z}|\mathbf{x})$ using the encoder $q_\phi(\mathbf{z}|\mathbf{x})$, which is assumed to be Gaussian and is parameterized by $\phi = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$, and the encoder learns to predict ϕ. As a result, it becomes possible to draw samples from this distribution.

In order to decode a sample $\mathbf{z}$ drawn from $q_\phi(\mathbf{z}|\mathbf{x})$, to the input $\mathbf{x}$, the reconstruction loss also needs to be minimized. The reconstruction loss is represented by $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(log\, p_\theta(\mathbf{x}|\mathbf{z}))$. If the VAE were trained similar to a sequence autoencoder, i.e., with reconstruction loss only, it would not allow enough variance in $q_\phi(\mathbf{x}|\mathbf{z})$, as a result the VAE would approximately map input $\mathbf{x}$ into a latent representation deterministically. This gets avoided by minimizing the reconstruction loss together with the KL-divergence between $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior $p_\theta(\mathbf{z})$.

$$\begin{aligned}\mathcal{L}(\phi, \theta, \mathbf{x}) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(log\, p_\theta(\mathbf{x}|\mathbf{z})) \\ +KL(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) \leq log\, p_\theta(\mathbf{x})\end{aligned} \tag{1}$$

Here, $p_\theta(\mathbf{z})$ is assumed to be multi-variate Gaussian $\mathcal{N}(0, I)$. [15] showed that this loss term is a variational lower bound on log likelihood of $\mathbf{x}$, i.e., $log\, p_\theta(\mathbf{x})$.

Sampling of $\mathbf{z}$ from $q_\phi(\mathbf{z}|\mathbf{x})$ is a non-continous operation, therefore it is not possible to train the encoder via back-propagation. The Re-parametrization trick introduced in [15] allows us to train VAE using stochastic gradient descent via back-propagation. Novel samples can be generated by sampling the $\mathbf{z}$ from $\mathcal{N}(0, I)$ and decoding the samples into input space, using the decoder model.

4.3 Language Model

An RNN language model (RNNLM) is a generative model, which learns the conditional probability distribution over the vocabulary words. It predicts the next word (w_{i+1}) given the representation of words seen so far $\mathbf{h}_i$ and current input $\mathbf{w}_i$ by maximizing the log likelihood of the next word $p(w_{i+1} \mid \mathbf{h}_i, \mathbf{w}_i) = \text{softmax}(W_s\mathbf{h}_i + b_s)$, averaged over sequence length N. The cross-entropy loss is used to train the language model.

$$\mathcal{L}_{CE\_LM} = -\frac{1}{N}\sum_{i=1}^{N} log(p(w_{i+1} \mid \mathbf{h}_i, \mathbf{w}_i)) \tag{2}$$

Generally performance of the RNNLM is measured using perplexity (lower is better), Perplexity = $\exp^{\mathcal{L}_{CE\_LM}}$. We have used RNNLM with LSTM units as described in [28].

4.4 Classification

Classification can be considered as a two step process with the first step requiring a representation of the data. The second step involves using this representation for classification. Data can be represented using a bag of words approach,

which ignores the word order information; or using hand-crafted features, which fail to generalize to multiple datasets / tasks. We learn the task-specific sentence representation using RNNs with LSTM units by representing the variable length sentence in a fixed length vector representation $\mathbf{h}$, obtained after passing the sentence through the RNN layer. We then apply softmax over the affine transformation of $\mathbf{h}$, i.e., $p(c \mid \mathbf{h}) = \text{softmax}(W_s\mathbf{h} + b_s)$. To learn the weights of the above model, we minimize the categorical cross entropy loss, i.e., $\mathcal{L}_{CE} = -\sum_{i=1}^{m} y \cdot log(p(c_i \mid \mathbf{h}))$, where c_i is one of the m class. Here, y is 1 only for the target class and zero otherwise.

5 LSTM-VAE with LM

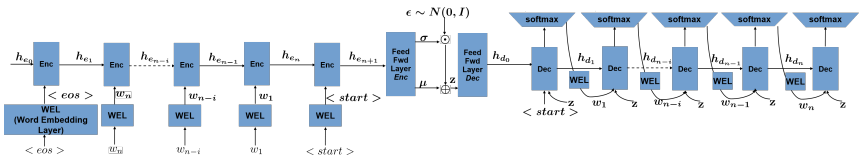

Fig. 1. LSTM-VAE Architecture

5.1 Variational Recurrent Autoencoder

Similar to [3], we have used single layer RNN with LSTM units as encoder and decoder of the VAE. We pass the variable length input sentence in the encoder in reverse order as shown in Figure 1. Words of a sentence are first converted into a vector representation after passing through the word embedding layer, before being fed to the LSTM layer. The final hidden state of the LSTM, $\mathbf{h_{e_{n+1}}}$ then passes through a feed forward layer, which predicts $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ of the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$.

After sampling and via the re-parameterization trick (explained later in this section), the sampled encoding $\mathbf{z}$ passes through a feed-forward layer to obtain $\mathbf{h_{d_0}}$, which is the start state of the decoder RNN. We pass the encoding $\mathbf{z}$ as input to the LSTM layer at every time-step. The vector representation of the word with the highest probability, predicted at time t, is also passed as input at next time-step $(t+1)$, as shown in Figure 1.

Training the LSTM-VAE It is not straightforward to train the LSTM-VAE using the loss function given in Eq. 1. As described in Section 4.2, if the KL-divergence loss term is not used, the model will converge to a state that gives fixed latent representations. Similarly, if the reconstruction loss is high it would lead to meaningless sentences. If equal weightage is given to both loss terms the

Source Sent: where can i apply earned leave?
Generated Sentences
where can i apply earned leave ?
where ***can can*** *check lwp guidelines ?*
where can i apply sick leave ?
where can i apply earned leave ?
where can i apply earned leave ?
where can i find lwp guidelines ?
where can i apply earned leave ?
where ***can can*** *apply casual leave ?*
how can i apply earned leave ?
where can i find lwp leave ?

Table 1. Sentence Generation

model learns to encode the input sequence into a desired latent representation too well, i.e., KL divergence (KLD) between approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and prior $p_\theta(\mathbf{z})$ reaches zero. As a result, the decoder network reduces to merely a language model (RNNLM) and it does not generate novel samples. The network should be trained such that both of the loss terms gradually decrease towards convergence.

This problem was also reported by [3], and they proposed cost annealing and word-dropout at the decoder as a remedy. For cost annealing, during the learning phase, after r_1 steps of training they increase the weight of the KL-divergence loss term gradually from 0 to 1, while retaining full reconstruction loss term from beginning to end. This approach did not work for us, and we could not train the model on our data. We therefore propose a variant of their approach, which worked well for us.

- *Weighted cost annealing*: We have used an improved version of the KL cost annealing presented in [3]. We utilize a weighted loss function as mentioned in Eq. 3 and started training the model with $\lambda = 0$, keeping it fixed for the first e epochs, i.e., $\lambda(0 - e) = 0$. We keep increasing it by r after every e epochs, i.e., $\lambda(e - 2e) = \lambda(0 - e) + r$. Here, e and r are assumed to be hyper parameters.

$$\begin{aligned}\mathcal{L}(\phi, \theta, \mathbf{x}) = \lambda \cdot KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\ -(1 - \lambda) \cdot \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(log\, p_\theta(\mathbf{x}|\mathbf{z}))\end{aligned} \tag{3}$$

- To avoid discrepancy between the training and inference procedures, we have used the special case of scheduled sampling mentioned in [1], i.e., we always take the word from predicted distribution as input instead of the actual word from the input sentence. We pass $\mathbf{z}$ at every step of the LSTM-decoder

with the highest probability word taken from the predicted distribution, i.e., greedy decoding $w_t = argmax_{w_t}\, p(w_t|w_{0,\ldots,t-1}, \mathbf{h_{d_{t-1}}}, \mathbf{z})$.
- To make the decoder rely more on the $\mathbf{z}$ during sentence decoding, we have used word dropout similar to [3] by passing the UNK token to next step instead of the word predicted by the decoder using greedy decoding. During decoding of a sentence using $\mathbf{z}$, k, the fraction of words are replaced by UNK randomly, where $k \in [0, 1]$ is also taken as a hyper-parameter.

Inference: Sentence Generation To generate sentences similar to input sentences, we have used the recognition model to get the parameters of the distribution corresponding to a sentence ($\boldsymbol{\mu}$,$\boldsymbol{\sigma}$). If we sample $\mathbf{z}$ from this distribution, the sampling step will become non-continuous and we will not be able to learn the parameters of the neural network using back propagation. The re-parameterization trick [15] comes to our rescue in this situation. Here, we sample ϵ and obtain $\mathbf{z}$ using the Eq. 4, which is a continuous function and therefore differentiable. These sampled encodings are decoded by the generative model using greedy decoding to obtain the sentences. Table 1 contains the set of sentences generated and the corresponding source sentences.

$$\mathbf{z} = \boldsymbol{\mu} + \epsilon \cdot \boldsymbol{\sigma}, where\, \epsilon \sim N(\mathbf{0}, \mathbf{I}) \tag{4}$$

5.2 Sentence Selection

It is difficult to use the sentences generated by the LSTM-VAE in a downstream task (*classification in our case*) without preprocessing because of the word repetition problem in the generated samples observed by various researchers [25]. Therefore, we discard all the sentences containing consecutively repeating words as highlighted in Table 1. The remaining sentences may still not be syntactically and semantically correct. We therefore choose the top-K good sentences sorted by likelihood via a LM (see Section 4.3) trained on the original training data.

5.3 LSTM based Sentence Classification

Base Classifier *(M_1)*: We use the single layer recurrent neural network with LSTM units for classification trained on the original data. We use this as a baseline for the classification task.

Base Classifier augmented with generated sentences *(M_2)*: To obtain the label for the novel sentences generated by VAE, we use M_1 and choose the top N sentences, based on the entropy of the softmax distribution, as candidates for augmenting the training data. We manually verify the label and correct the label if it is incorrectly classified by M_1; we also remove the sentences that clearly correspond to new classes. We augment the training data with this new dataset and train another classification model (M_2), as shown in Figure 2.

Base Classifier augmented with generated sentences based on word embedding *(M_3)*: Here, we try to augment the training set with additional sentences that are generated by replacing some words in each training sentence. All

the words in a sentence except english stopwords and leave types {sick, casual etc.} (to avoid change of class label) are selected as the candidate words for replacement. We replace a candidate word with a word corresponding to the nearest neighbour of the candidate word's embedding. The nearest neighbour is chosen based on the cosine similarity between two word embeddings obtained from glove [21]. We generate one sentence per each training sentence and augment the training data with these generated sentences to train another LSTM based classification model (M_3).

6 Results and Discussion

6.1 Experimental Setup and Training

Dataset Description: We use a dataset of Leave policy questions, created by HR officers[1], for chatbot creation, see Table 2 for more details. This dataset was divided in three parts (Training, Validation and Test) in 60:20:20 ratio. Training and Validation datasets are used for training the LSTM-VAE model, here we select the best model based on loss on validation data, i.e., $\mathcal{L}(\phi, \theta, \mathbf{x})$, see Eq. 3. New sentences are generated using only training data as input. The test data is used for reporting the performance metrics of the classification model, and is never exposed to LSTM-VAE model. Another test data comprising only the core concept words was also used for testing (see Section 6.3).

Characteristic	N	c	l	V
Value	2916	117	11	1108

Table 2. Characteristics of the Leave dataset N: Number of sentences, c: Number of classes, l: Average sentence length, V: Vocabulary size

Training Details: Word embeddings for tokens (delimited by space) are initialized randomly and learned during the training. We use Adam [13] for optimization and learning rate is selected from the range [1e-2,1e-3] for all the models, the choice of other hyper-parameters is given in Table 3. For regularization of classification model, we have used dropout [24, 22] on word embedding and LSTM layers, along with batch normalization [12]. For LM, we used truncated back-propagation for gradient calculation. Next, we present the details of a) new classes generated by our approach, b) results on core concept word testing demonstrating linguistic training bias, and finally show c) the improvement in accuracy using the generated sentences.

[1] A subset of this data set will be shared on demand for research purposes

Parameter	**Range**
Dim. of word embed.	$\{50, 100, ..., 300\}$
Dimension of $\mathbf{z}$ (30)	$\{20, 30, 50\}$
weighted cost annealing	
r (chose .05)	$\{.1, .05, .025\}$
e (chose 10)	$\{5, 10, 15\}$
k (LSTM-VAE word-drop)	$\{0.5, 0.75, 1.0\}$
Classification Model	
and *Language Model*	
Word drop rate	$\{0, 0.1, 0.2, ..., 0.5\}$

Table 3. Hyper Parameter Tuning ranges

6.2 New Classes generated by LSTM-VAE

When preparing the questions to train a chatbot as described in Section 3, it is hard to visualize all possible questions (i.e., query-sets) that users can ask when it is made live. Using LSTM-VAE we were able to generate many new query sets, which were accepted by the core HR Team. In Figure 2 we present the entire workflow followed to generate the novel queries. Out of the 175,000 queries generated using LSTM-VAE, we removed the queries already present in the training data, as well as those which have same word repeating more than once consecutively. After this stage we obtained about 5,700 sentences. These sentences were then tested using a LM (as described in Section 4.3), and we picked-up only top 1500 sentences based on the likelihood. Many of these sentences were found to be grammatically correct, while only some of them were semantically inconsistent. In this process it was discovered that 434 sentences did not belong to any of the existing classes. These sentences were given to the HR team for review, and they selected 120 sentences belonging to 33 new classes. We have made this chatbot live in our company and these 33 classes are in practical use (*see Key Contribution 2*).

Further, in Table 4, we show sample novel sentences generated by LSTM-VAE belonging to 9 different classes (separated by horizontal line in the table). For every novel sentence, we also show the most similar sentence taken from training data, identified using Jaccard Similarity.

6.3 Core Concept Word Testing

Next, our HR team tested the chatbot built using models M_1 and M_2 (see Section 5.3) by using only core concept words as inputs, rather than complete sentences: see Table 5. To build the model M_2, we added sentences generated by the process flow, shown in Figure 2. Here, out of 1500 sentences chosen by the LM 1066 were manually found to be belonging to existing classes, and we ran the model M_1

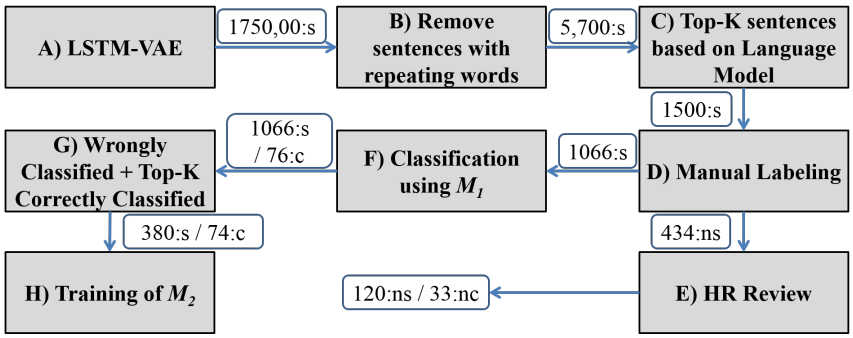

Fig. 2. Sentence Generation Process flow, where s: VAE generated sentences, c: classification of VAE generated sentences into existing classes, ns: novel sentences found during maual labelling of the generated sentences, nc: Classification of valid ns by HR officers into new classes

Novel Sentences generated by LSTM-VAE	**Most Similar Sentences from training data**
Can adoption leave be applied in advance	*In what all scenarios can adoption leave be applied*
Can i use adoption leave in advance	*Can i avail adoption leave in parts?*
How do i see more on leave ?	*Where do i get more information on adoption leave ?*
Please help with know more information on leave ?	*Pls help with more information on timesheet leave ?*
Where do i get more about on leave ?	*Where do i get more information on adoption leave ?*
Where do i see faq on leave ?	*Where do i get more information on adoption leave ?*
Can lwp be encashed ?	*Can casual leave be planned ?*
What is the eligibilty to encash sick leaves ?	*What is the method to encash adoption leave ?*
Are sick leaves credited during to separation ?	*Are sick leaves credited during lwp ?*
Can casual leave combined with earned leave ?	*Can al be combined with casual leave ?*
How many days leaves can credited in a quarter ?	*How many casual leaves are credited in a quarter ?*
Can i cancel my leave in system ?	*How can i cancel my availed casual leave ?*
What is the procedure to cancel for leave ?	*What is the procedure to cancel adoption leave ?*
I have resigned recently . i go for lwp ?	*I have resigned. can i go for casual leave ?*

Table 4. Novel sentences generated using LSTM-VAE and corresponding similar sentences present in the training data based on Jaccard-Similarity

Sentences from training data	**Concept Word Query**
Are business associates eligible to take al? *Are business associates eligible for al?*	*Adoption leave business associates*
Where can i find the policy on casual leave? *Where can i read more about casual leave ?* *Where do i get more info on casual leaves?* *Where i can find cl guidelines ?*	*Casual leave policy path*
How many flexi leaves will i get ? *What is flexi leave and how can i avail them ?*	*Flexi Leave Entitlement*
Can i apply ml if i am on lwp ? *How can i go about applying ml if on lwp ?* *Is it possible for me to apply ml when on lwp ?*	*Maternity leave entitlement while on LWP*
What happens to my unutilized sick leave ? *In event of separation can i encash my sick leave ?*	*Sick leave resign during separation ?*

Table 5. Training data queries and corresponding concept words queries

on these sentences. We chose top-250 correctly classified sentences based on low entropy and all the 130 wrongly classified sentences, and combined these with the training data to train the model M_2. These 380 sentences belong to only 74 out of 117 classes. As shown in Table 6, a classifier built using only the training dataset tends to have linguistic training bias. This bias gets reduced when the generated sentences are also used (*see Key Contribution 1*). This difference of 9% can be attributed to new sentences generated via perturbation of surrounding words and high quality sentences generated by LSTM-VAE.

6.4 Accuracy on Test set

We now analyze the impact of addition of this training data into the model on hold-out set. Here, a small gain of about 2% was observed in classification accuracy, as shown in Table 6 (*see Key Contribution 3*). While this gain is small, it is important that the classifier uses the right words; especially since the test set, being a subset of curated questions, also has linguistic training bias. It was observed that the model M_1 often classified the query using non-relevant words as shown in Table 6. Even if such a classifier had worked better on the test data, we believe it would perform worse on actual user queries; of course this can only be confirmed by a controlled A/B test on real users.

6.5 Analysis: Why Linguistic Training Bias gets reduced?

During user testing of the FAQ-bot, it was observed that often model's decision to classify a user query in certain class is based on non-concept words. It was suspected that this phenomenon occurs due to linguistic training bias, as explained

Dataset	M_1	M_2	M_3
Concept Word Testing	55.00%	**64.28%**	62.14%
Test Set	79.41%	**81.13%**	80.78%

Table 6. Accuracy of classifiers on *test set* and sentences containing *concept words (CW)* only, where number of classes is 117 for M_1, M_2 and M_3

in Section 1. New sentences generated by LSTM-VAE, which were wrongly classified by M_1, are likely to be of this type, and are therefore more value adding for the model. For example, many sentences get generated which contain the same surrounding words (e.g., 'gets') but belonging to different classes, e.g., "*When my casual leaves gets credited ?*" and "*When my maternity leaves gets credited ?*". When such sentences are added to the training set, it indirectly forces the model to learn to distinguish the classes based on some other words than such non-concept words. Perhaps this is one of the reasons why merely with an addition of 130 sentences (which is less than 10% of the original training data) accuracy gain of almost 10% was observed during concept-word based testing of the model as shown in Table 6. It would otherwise have required many more new sentences to get the same degree of gain in classification accuracy.

7 Conclusion

Based on our experience of building and deploying a FAQ-bot in practice, we have noted that linguistic training bias leads to over-fitting of deep-learning model. Such bias is more likely to occur when the training data is created by domain experts, who tend to use templates when writing natural language sentences. We have shown that such bias can be fixed to certain extent by generating novel sentences using a generative model. We also described an approach for such a generative model, which uses LSTM-VAE followed by sentence selection using a LM. We presented *weighted cost annealing*, an improved approach for training the LSTM-VAE, and showed that accuracy of a classification model can be increased significantly by using generated sentences. Most importantly, we have shown that our approach was able to generate newer classes of questions for our FAQ-chatbot, not present in the original training data, which were reviewed and accepted by the domain experts for deployment.

References

1. Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28*. 2015.
2. Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*

3. Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.
4. Wojciech Zaremba Christian Szegedy, Ilya Sutskever, Dumitru Erhan Joan Bruna, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.
5. Ann Copestake and Dan Flickinger. An open source grammar development environment and broad-coverage english grammar using hpsg. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, 2000.
6. Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems 28*. 2015.
7. Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.
8. Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. In *Advances in Neural Information Processing Systems 29*. 2016.
9. Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.
10. David Ha and Douglas Eck. A neural representation of sketch drawings. *CoRR*, 2017.
11. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 1997.
12. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.
13. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.
14. Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems 27*. 2014.
15. Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.
16. Yitong Li, Trevor Cohn, and Timothy Baldwin. Robust training under linguistic adversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, April 2017.
17. Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.
18. Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *11th Annual Conference of the International Speech Communication Association*, 2010.
19. George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to wordnet: an on-line lexical database. *International Journal of Lexicography*, 1990.
20. Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints. In *Proceedings of HLT-NAACL*, 2016.

21. Jeffrey Pennington, Richard Socher, and Christopher Manning. Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*, EMNLP, 2014.
22. Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.
23. Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A hybrid convolutional variational autoencoder for text generation. *CoRR*, 2017.
24. Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.
25. Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.
26. Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
27. Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. *CoRR*, 2017.
28. Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, 2014.

Appendix: Effect of different training procedures on sentence generation

– **Weighted cost annealing**: As mentioned in Section 5.1 the training of LSTM-VAE by minimizing the loss according to Equation 5 causes the KL-divergence loss to increase for few time-steps initially and then drop to zero as shown in Figure 3. This makes the decoder to behave like an RNNLM. To overcome the above issue we use weighted cost annealing i.e. we increase the weight of KL-divergence loss lineraly after every e epochs and simultaneously reduce the weight of the reconstruction loss. Due to this, even though the KL-divergence loss increases initially for few time-steps, it starts decreasing over the time-steps but remains non zero as shown in Figure 3.

$$\begin{aligned}\mathcal{L}(\phi, \theta, \mathbf{x}) = KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\ -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(log\, p_\theta(\mathbf{x}|\mathbf{z}))\end{aligned} \tag{5}$$

– As described in Section 5.1 the input to the LSTM-VAE decoder, for predicting the word at time $t+1$, is the output of the decoder at time t rather than the actual word from the input sentence. In our case we found that this helps the LSTM-VAE in generating more semantically and syntactically correct sentences as compared to those generated by inputting the actual word to the decoder. This can be observed from the Table 7.

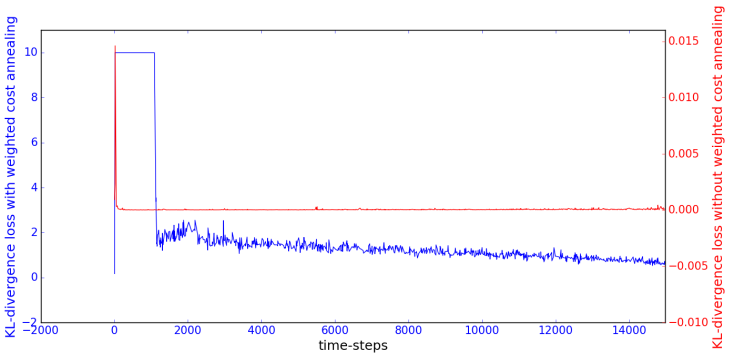

Fig. 3. KL-divergence loss over the training steps (horizontal axis)

Source Sent: path to apply sick leave?	
Predicted word as decoder input	**Actual Word as decoder input**
path to apply casual leave ?	*path sick sick sick cl cl cl ? ? ? ?*
path to apply sl ?	*path cl cl cl cl cl ? ? ? ? ?*
where to apply casual leave ?	*path sick sick sick cl cl cl ? ? ? ?*
path to apply earned leave ?	*who sick sick cl cl cl cl ? ? ? ?*
how to apply adoption leave ?	*path sick sick sick cl cl cl ? ? ? ?*
can to apply adoption leave ?	*path sick sick cl cl ? ? ? ?*
link to apply sick leave	*path sick sick sick cl cl cl ? ? ? ?*
path to apply leave ?	*path path sick sick cl cl cl el ? ? ? ?*
path path apply adoption leave ?	*path sick sick cl cl cl cl ? ? ? ?*
path to apply adoption leave ?	*path sick sick cl cl cl ? ? ? ?*

Table 7. Sentences generated using VAE under two different training scenario