Towards Continual Entity Learning in Language Models for Conversational Agents

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural language models (LM) trained on diverse corpora are known to work well on previously seen entities, however, updating these models with dynamically changing entities such as place names, song titles and shopping items requires re-training from scratch and collecting full sentences containing these entities. We aim to address this issue, by introducing entity-aware language models (EALM), where we integrate entity models trained on catalogues of entities into the pre-trained LMs. Our combined language model adaptively adds information from the entity models into the pre-trained LM depending on the sentence context. Our entity models can be updated independently of the pre-trained LM, enabling us to influence the distribution of entities output by the final LM, without any further training of the pre-trained LM. We show significant perplexity improvements on task-oriented dialogue datasets, especially on long-tailed utterances, with an ability to continually adapt to new entities (to an extent).

1 Introduction

Neural Language Models have become the de facto standard for modeling natural language; however, they still lack some of the flexibilities offered by their n-gram counterparts [3] in being able to continually update to new entities at a minimal cost [30] or quickly adapt to personalized content [1]. This issue is more prominent in task-oriented dialogue systems where utterances are typically rich in entities such as place names, shopping items, song titles, person names etc. To address some of these issues, we introduce entity-aware language models (EALM), where we attempt to decouple entity models from the pre-trained LMs. We achieve this by linearly interpolating the final layer output representations of the pre-trained LM and independently trained entity models. The interpolation probabilities at each timestep are determined by a contextual fusion layer based on the sentence context. This approach allows us to better predict entities present in text, using the information provided by the entity models along with the distribution learned by the pre-trained LM. Our final language model consists of three components (1) A pre-trained language model, (2) Entity models, and (3) A Contextual fusion layer to combine the pre-trained LM with entity models.

Our entity models are neural LMs trained on catalogues containing entities. They share the input and output layers with the pre-trained LM and these parameters are frozen throughout the training, essentially mapping entities to the pre-trained LM’s embedding space. This modular approach allows us to update our entity models with new information, independently of the pre-trained LM. The improved entity models can be directly plugged back into the final LM without any further training. This enables our LM to continually learn new entities without requiring to update the pre-trained LM.
It is important to note that our final LM should not always rely on entity models while predicting entities in text. Entities in natural language can be classified into two categories based on the context in which they appear: (1) fact-based entities that depend on real-world facts, and (2) preference-based entities that depend on external factors such as time, location, speaker, etc. For example, in the utterances play hello by adele and play any song by adele, artist adele in the first utterance can be classified as a fact-based entity and is determined by the utterance context alone (plus some real-world knowledge), whereas, adele in the second is a preference-based entity that depends on auxiliary information outside of the utterance. Unlike [15, 28, 44], that are designed to improve the representation of fact-based entities, in this paper, we only focus on preference-based entities.

Our main contributions are as follows: (1) We propose an approach to integrate entity models trained on catalogues of entities into a pre-trained LM, (2) We show that this improves our LM’s capability in predicting long-tailed entities and continually learn new entities to an extent.

### 1.1 Our Approach

Given utterance context \(w_{0...t-1} = (w_0, ..., w_{t-1})\), the goal of an autoregressive language model is to estimate the probability distribution \(p(w_t|w_{0...t-1})\) of the next token \(w_t\), where \(w_0\) is typically the start token <s> used to predict the first token. We can factorize this probability distribution as,

\[
p(w_t|w_{0...t-1}) = \sum_{i=0}^{N} p(w_t|w_{0...t-1}, E_i) \cdot p(E_i|w_{0...t-1}) \tag{1}
\]

where \(E_0\) represents the pre-trained LM and \(E_1, ..., E_N\) represent the entity LMs. However, entity models cannot accurately predict \(p(w_t|w_{0...t-1}, E_i)\) as they are only trained on entity catalogues. As a result, we use the utterance context \(w_{0...t-1}\) to determine the entity context \(w_{t-1*}, ..., w_{t-1}\) containing only entity terms and approximate \(p(w_t|w_{0...t-1}, E_i)\) as,

\[
p(w_t|w_{0...t-1}, E_i) \approx p(w_t|[w_0; w_{t-1*}, ..., w_{t-1}]), E_i) \quad \forall i = 1...N
\]

\[
l^* = \arg \max_{l \in [0, t-1]} p(l|w_{0...t-1}, E_i) \tag{2}
\]

where \([w_0; w_{t-1*}, ..., w_{t-1}] = (w_0, w_{t-1}, ..., w_{t-1})\). We assume \([w_0; w_{t-1*}, ..., w_{t-1}] = w_0\) for notational simplicity. We further simplify Eq. \(2\) using markov assumption on the entity models by only considering \(k\) tokens preceding the current timestep, \(l^* = \arg \max_{l \in [0, k]} p(l|w_{0...t-1}, E_i)\). We show the utterance contexts of our entity models at different timestamps using an example in Table \[\]

To convert this into a continuous optimization problem, we rewrite Eq. \(2\) as an expectation over all possible entity contexts and our updated equation looks like,

\[
p(w_t|w_{0...t-1}, E_i) \approx \sum_{l=0}^{k} p(l|w_{0...t-1}, E_i) \cdot p(w_t|[w_0; w_{t-l*}, ..., w_{t-1}]), E_i) \quad \forall i = 1...N \tag{3}
\]

### 2 Entity-Aware Language Models

In this section, we describe our proposed Entity-Aware Language Models (EALM). Our LM has three components that are trained sequentially: (1) A pre-trained language model, (2) Entity models and, (3) A contextual fusion layer. Our LM components and their interactions are outlined in Figure \[\]

Given an utterance context \(w_{0...t-1} = (w_0, ..., w_{t-1})\), our LM first converts these tokens into a dense representation \(X = (x_0, ..., x_{t-1})\), where \(X \in \mathbb{R}^{l \times d}\) using an input embedding layer. The pre-trained LM and the entity models process this input representation as described below.

\(^1\) \(l - l^*\) denotes the start index of the entity in the utterance, if the current phrase is an entity, \(t - l^* = 0\) otherwise.
2.1 Pre-trained Language Model

We use a multi-layer Transformer decoder similar to [39, 31] as our pre-trained LM. Given the input representation \( X \), we first add positional information to it, 
\[
H_0^P = X + P_e \quad \text{where} \quad P_e \in \mathbb{R}^{t \times d}.
\]

\[
H^P = \text{GlobalAttentionTransformer}(H_0^P) \quad \text{(4)}
\]

The last token representation \( h_{t-1}^P \) of \( H^P \), where \( h_{t-1}^P \in \mathbb{R}^d \) and \( H^P \in \mathbb{R}^{t \times d} \), encodes \( w_0 \ldots t-1 \).

2.2 Entity Models

Unlike the pre-trained LM that takes the entire utterance context \( w_0 \ldots t-1 \) as input, entity models only process the last \( k \) tokens as described in Eq. 3. We, therefore, use local attention [26] and relative positional embeddings [45] for our entity model self-attention layers. In addition to this, our entity models generate \( k+1 \) outputs at every timestep, by varying the utterance context length from 0 through \( k \) based on Eq. 3. To summarize, our entity models take \( X' = (x_0, x_{t-k}, \ldots, x_{t-1}) \) as input and generate the current timestep’s output \( h_{t-1}^{E_{l}} \in \mathbb{R}^{k+1 \times d} \) by effectively masking the attention context based on the utterance context length \( l \), where \( l \in [0, k] \),

\[
h_{t-1}^{E_{l}} = \text{LocalAttentionTransformer}(H_0^{E_{l}}) \quad \text{(5)}
\]

Note that \( k+1 \) here does not represent the time dimension, but rather represent the \( k+1 \) different outputs obtained by changing the utterance context length from 0 to \( k \).
2.3 Contextual Fusion Layer

The contextual fusion layer integrates information from the entity models into the pre-trained LM by linearly interpolating their final layer output representations based on the utterance context. The fusion layer represents the entity models and the pre-trained LM using a class embedding matrix $W_c \in \mathbb{R}^{N+1 \times d}$, where $N$ is the number of entity models. It also creates a new encoding $h_{l-1}^{E} \in \mathbb{R}^d$ of the utterance context $w_{0...t-1}$ using an additional transformer decoder on top of Eq. 4.

The goal of the contextual fusion layer is twofold: (1) Approximate the probability distribution of the next token given by our entity models by identifying their respective entity contexts (Eq. 3) and (2) Determine the probability distribution of the next token given by our final LM by estimating the interpolation weights (Eq. 1).

**Entity Contexts** Given $h_i^{E_i}$, $W_c^{E_i}$ and $h_i^{C}$, we first compute the probability distribution $p_{context}^{E_i} \in \mathbb{R}^{k+1}$ over all possible entity contexts at the current timestep as shown in Figure 1. We then create a compact representation $o_{l-1}^{E_i} \in \mathbb{R}^d$ of the entity model’s output as follows,

$$
p_{context}^{E_i} = \text{EntityOutputMixer}(W_c^{E_i}, h_i^{E_i}, h_i^{C})
$$

$$
o_{l-1}^{E_i} = \sum_{l=0}^{k} p_{context}^{E_i} \cdot h_i^{E_i} \quad \forall i = 1...N
$$

where the left superscript $l$ indicates the $l^{th}$ row of the matrix or vector.

**Entity Fusion** Given $h_i^{P}$, $o_i^{E_i}$, $W_c$ and $h_i^{C}$ we compute the interpolation probabilities $p_{fusion}^{E_i} \in \mathbb{R}^{N+1}$ at every timestep as shown in Figure 1. We combine the output representations of the entity models and the pre-trained LM as follows,

$$
^i p_{fusion} = \text{Softmax}(\text{PretrainedEntityFuser}(o_{l-1}^{E_i}, W_c^{E_i}, h_i^{C}))
$$

$$
h_i^{EALM} = \sum_{i=0}^{N} ^i p_{fusion} \cdot o_{l-1}^{E_i}
$$

where $E_0$ represents the pre-trained LM and $o_{l-1}^{E_0} = h_i^{P}$. We use the interpolated representation $h_i^{EALM}$ to obtain the probability distribution of the next token using,

$$
p^{EALM}(w_{t}|w_{0...t-1}) = \text{Softmax}(h_i^{EALM}W_o)
$$

We freeze the parameters of the pre-trained LM, input and output embedding layers while training the entity models. We also freeze entity models during our contextual fusion layer training. All our LMs and the Fusion Layer are trained to minimize the negative log-likelihood of the utterances or entities.

3 Experiments

Our training data consists of transcribed utterances from user interactions with a voice assistant. We use entity catalogues to train entity models. Please refer [A.1] for our detailed experimental setup. We evaluate our model on its ability to represent the long-tail and its effectiveness in continually learning new entities without forgetting old ones. We provide a description of the test sets used for evaluation in Table 1. All our test sets are created from a much bigger general test set unless specified. We make an assumption that entities are new if they are not present in our training data and catalogues. We evaluated all our test sets on the pre-trained model and Table 2 shows the relative percentage difference in perplexity of our tail and new test sets compared to the entity seen test set. We use the tail/new test sets as the reference (denominator) for this table.

---

3 This is slightly different from Eq. 1 where the interpolation is done on the probability distributions.
Table 1: Description of our testsets that are de-identified, transcribed and annotated.

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Entities are seen during training</th>
<th>Entities are present in our catalogues</th>
<th>Every utterance contains an entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>48k</td>
<td>some of them</td>
<td>some of them</td>
<td>no</td>
</tr>
<tr>
<td>EntitySeen</td>
<td>18k</td>
<td>yes (a few times)</td>
<td>some of them</td>
<td>yes</td>
</tr>
<tr>
<td>EntityTail</td>
<td>8k</td>
<td>only individual words</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>EntityNew</td>
<td>14k</td>
<td>only individual words</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>EntitySpanish</td>
<td>23k</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>EntityTailNew</td>
<td>23k</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 2: Relative difference in perplexities of our test sets when evaluated on our pre-trained LM.

<table>
<thead>
<tr>
<th>General</th>
<th>EntitySeen</th>
<th>EntityTail</th>
<th>EntityNew</th>
<th>EntitySpanish</th>
<th>EntityTailNew</th>
</tr>
</thead>
<tbody>
<tr>
<td>-11.46%</td>
<td>0%</td>
<td>47.55%</td>
<td>54.83%</td>
<td>85.8%</td>
<td>88.75%</td>
</tr>
</tbody>
</table>

**Evaluation on Long-Tail** We show the relative reduction in perplexity obtained by our baseline EALM compared to the pre-trained LM, on all our test sets, in the first part of Table 3. We can see that our baseline EALM consistently outperforms the standalone pre-trained model on all our test sets. The improvement is much more significant on tail entities (10.37%), that are unseen during training, but are present in our catalogues.

**Continual Learning of Entities** To evaluate our model on whether it can continually learn entities, we re-trained all our entity models by adding entities from the tail and new test sets (including dev sets), to the top five percent (ranked by popularity) of our catalogues. We did not retrain the contextual fusion layer. We see a big improvement from 5.63% to 14.38% on utterances containing new entities, refer Table 3. While this is encouraging, the improvement needed to match the performance of our pre-trained LM on popular entities, assuming we have full utterances containing these entities can be inferred as 54.83% from Table 2. We also evaluated our method on whether it can learn popular entities from other languages, which is not atypical in task-oriented dialogue systems. We can see that our model achieved a huge overall perplexity reduction of 39.12%. Again, this is still significantly lower compared to the 85.8% (refer Table 2) improvement needed to match the performance on existing popular entities. Refer A.2 for further discussion on continual learning.

**Future Work**

Our language models can be directly used for shallow fusion [38, 4] and n-best hypothesis rescoring [14, 36] in seq2seq based speech recognition systems. Our approach can also be extended to integrate entity models directly into the decoders of these systems. To improve the recognition of tail entities, we can adapt our entity models based on context such as user, location, season etc [21]. To scale this, we might have to explore sharing parameters across entity models. We can also use our approach for quick hotfixes in production systems. Another interesting line of work would be to use our EALM for intent classification and entity extraction tasks by making use of the interpolation probabilities Eq. 7 as additional cues indicating the presence of entities. Our LM may not have a huge impact on tasks external [40] to the dialogue systems which benefit more from fact-based entities and their relations.

**Related Work**

**External Memory** Adding external memory to the neural networks was first proposed in [42, 37] where they used attention mechanism [2] to access memory, based on the current context. [22, 13] extended this to LMs to increase the probability of recently seen words and later [12] scaled this to larger contexts using k-nearest neighbors to retrieve from memory. Recently, [13] proposed to learn

5
Table 3: Table showing the effectiveness of EALM on different testsets. All results shown are percentage reduction in perplexity compared to the pretrained LM. We removed the % sign.

<table>
<thead>
<tr>
<th>Testset</th>
<th>All</th>
<th>Song</th>
<th>Album</th>
<th>Celeb</th>
<th>Video</th>
<th>Item</th>
<th>Place</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Our base EALM. Entity models are trained on full catalogues.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General</td>
<td>1.56</td>
<td>3.77</td>
<td>1.88</td>
<td>3.32</td>
<td>2.81</td>
<td>2.72</td>
<td>1.91</td>
<td>0.99</td>
</tr>
<tr>
<td>EntitySeen</td>
<td>5.46</td>
<td>8.83</td>
<td>5.42</td>
<td>9.33</td>
<td>0.52</td>
<td>2.15</td>
<td>1.38</td>
<td>2.74</td>
</tr>
<tr>
<td>EntityTail</td>
<td>10.37</td>
<td>16.52</td>
<td>9.82</td>
<td>16.19</td>
<td>4.22</td>
<td>7.69</td>
<td>4.02</td>
<td>5.14</td>
</tr>
<tr>
<td>EntityNew</td>
<td>5.63</td>
<td>8.66</td>
<td>6.46</td>
<td>8.39</td>
<td>3.54</td>
<td>5.69</td>
<td>3.57</td>
<td>0.81</td>
</tr>
<tr>
<td>EntitySpanish</td>
<td>21.9</td>
<td>28.32</td>
<td>23.24</td>
<td>29.32</td>
<td>4.22</td>
<td>7.69</td>
<td>4.02</td>
<td>5.14</td>
</tr>
<tr>
<td><strong>Entity models are re-trained by adding tail and new entities and plugged back into the EALM.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General</td>
<td>1.73</td>
<td>3.71</td>
<td>2.46</td>
<td>3.32</td>
<td>3.98</td>
<td>1.57</td>
<td>1.48</td>
<td></td>
</tr>
<tr>
<td>EntitySeen</td>
<td>5.34</td>
<td>8.36</td>
<td>5.45</td>
<td>8.92</td>
<td>0.31</td>
<td>2.57</td>
<td>1.5</td>
<td>2.45</td>
</tr>
<tr>
<td>EntityTail</td>
<td>14.79</td>
<td>19.89</td>
<td>13.29</td>
<td>20.44</td>
<td>6.18</td>
<td>13.89</td>
<td>7.23</td>
<td>5.43</td>
</tr>
<tr>
<td>EntityNew</td>
<td>14.38</td>
<td>16</td>
<td>13.66</td>
<td>18.03</td>
<td>9.35</td>
<td>15.73</td>
<td>9.54</td>
<td>7.2</td>
</tr>
<tr>
<td><strong>Entity models are re-trained by adding spanish entities and plugged back into the EALM.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General</td>
<td>1.25</td>
<td>2.99</td>
<td>1.45</td>
<td>2.95</td>
<td>2.95</td>
<td>2.97</td>
<td>1.39</td>
<td>0.84</td>
</tr>
<tr>
<td>EntitySpanish</td>
<td>39.12</td>
<td>40.53</td>
<td>37.93</td>
<td>48.14</td>
<td>24.37</td>
<td>41.21</td>
<td>28.03</td>
<td>14.02</td>
</tr>
</tbody>
</table>

new information by just populating the memory with more data using a pre-trained LM trained on a small corpus. These methods are orthogonal to our work as they can only improve the prediction of fact-based or recently seen entities and cannot be easily extended to preference-based entities, using only a few nearest neighbors for interpolation or computing attention over millions of entities \[29\].

**Continual Learning** Continual learning approaches focus on learning new information without forgetting the previously acquired knowledge. Current continual learning approaches can be classified into three categories: (i) regularization \[16, 5\], (ii) memory \[6, 7\] and (iii) expansion \[32, 33\] based approaches. They are still an active area of research and require full sentences to learn new entities. It is also not straightforward on how to use these approaches for personalization and contextualization.

**Entity Models** Earlier work on building separate models for entities and combining them with a general language model used annotated data and class \(n\)-gram LMs \[41\]. \[27\] extended this to neural models, by predicting the entity type along with the next word and improved the representation of tail entities. \[19\] followed a similar approach but instead encoded the type information in a latent variable with the help of type specific vocabularies and did not use any labeled data. However, these approaches only work for entities seen during training and cannot be extended to support personalization or continual learning. Our work resembles \[9\], where they interpolate component models trained on entity catalogues with a pre-trained LM. Unlike our approach, they explicitly reset the hidden states of the entity models, using a binary activation process, to indicate the start of entities in the utterance context and use reinforcement learning to learn the discrete activation policy.

**6 Conclusion**

We proposed a technique to integrate entity models trained on catalogues of entities into pre-trained LMs, significantly improving the perplexity on long-tailed entities. We also showed that our LM can continually learn new entities at a low cost, outperforming the pre-trained LM on these entities, without any signs of forgetting. We discussed the limitations of our continual learning approach in matching the performance of a pre-trained LM directly trained on these new entities. Nevertheless, our LMs can still be used to learn new entities or hotfix trending entities between deployment cycles and can be easily extended to support personalization and contextualization in conversational agents.
References


Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association*

9


A Appendix

A.1 Experimental Setup

Our training pipeline consists of three stages: (1) Pre-training a language model (2) Training entity models and (3) Training a contextual fusion layer to combine the pre-trained model with entity models. We only use transformer decoder blocks [39] and linear layers in all our networks. We use $T_{\text{layers}}$ to represent the number of layers in the transformer, $d_{\text{model}}$ for its hidden, input and output dimensions, $d_{\text{ff}}$ for the feedforward dimension, $T_{\text{heads}}$ as the number of attention heads and $T_{\text{dropout}}$ for dropout between the layers. All our models use the AdamW optimizer [20] with a weight decay of 0.1 and a step-wise decaying learning rate scheduler [10] with warmup. We use $lr_{\text{warmup}}$ to define number of tokens until warmup and $lr_{\text{decay}}$ to define the number of tokens after which we decay the learning rate by a factor of 0.9. We use $lr_{\text{start}}$, $lr_{\text{max}}$ and $lr_{\text{end}}$ to define the starting, maximum and final learning rates respectively. We conduct all our experiments on utterances belonging to Entertainment, Shopping, LocalSearch and Communication domains.

Pre-trained Language Model Training  Our training data consists of transcribed utterances from user interactions with a voice assistant. The total number of tokens available for training is 40 million. All the utterances are de-identified to preserve the privacy of the users. Our pre-trained model parameters are $T_{\text{layers}} = 12$, $d_{\text{model}} = 512$, $d_{\text{ff}} = 1024$, $T_{\text{heads}} = 8$ and $T_{\text{dropout}} = 0.1$. The parameters of our learning rate scheduler are $lr_{\text{start}} = 1e^{-6}$, $lr_{\text{max}} = 6e^{-4}$, $lr_{\text{end}} = 1e^{-6}$, $lr_{\text{warmup}} = 2^{22}$ and $lr_{\text{decay}} = 2^{22}$. Our vocabulary consists of 8192 character level subword byte-pair encodings trained using [34]. We truncate all utterances that are longer than 32 subword units after tokenization and use absolute positional embeddings of size 512 to represent the position of these tokens. We accumulate gradients for two steps before updating the parameters of our network [23]. We do not use any bias parameters for the output linear layer before softmax.

Entity Models Training  We used the following entity models for our experiments: (1) song name, (2) album name, (3) celebrity name, (4) video name, (5) shopping item, (6) place name and (7) person name. The size of our entity catalogues ranges from from 2 million to 10 million entries depending on the entity model. The entities in each catalogue are assigned a popularity score using some simple heuristics and we sample the entities based on their popularity during training. Our entity model parameters are $T_{\text{layers}} = 4-6$, $d_{\text{model}} = 512$, $d_{\text{ff}} = 1024$, $T_{\text{heads}} = 4$ and $T_{\text{dropout}} = 0.1$. Our learning rate scheduler parameters are same as the pre-trained model except for $lr_{\text{decay}} = 2^{25}$. We use a markov dependency length (Eq. 3) of $k = 4$ for all our entity models.

Contextual Fusion Layer Training  We only use 8% of our training data to train the contextual fusion layer. Our catalogues cover around 40% of the entities present in this data. We use a one layer transformer with new positional embeddings to represent the utterance context from Section 2.3. The output dimension of all our linear layers is 512 except for the ones preceding a softmax layer. We use a dropout of 0.25 on the entity model outputs and 0.1 for the rest of the network. The number of epochs used for training is 4 and our learning rate scheduler parameters are $lr_{\text{start}} = 1e^{-6}$, $lr_{\text{max}} = 6e^{-5}$, $lr_{\text{end}} = 1e^{-7}$, $lr_{\text{warmup}} = 2^{22}$ and $lr_{\text{decay}} = 2^{22}$. Our batch size is 256 and gradient accumulation is done for two steps before parameter updates. We freeze all the parameters of the pre-trained LM and entity models during training. Our approach is fully self-supervised once we collect decent sized catalogues, containing some percentage of the entities present in the data used to train this layer. Our method works better if we train this layer on entities not seen by the pre-trained LM, however, we did not do that for this paper. We also found that the contextual fusion layer ignores certain entity models if there is a huge imbalance in the data used to train this layer, especially, for entities appearing in similar contexts such as songs and videos.
A.2 Discussion

We discuss the strengths and drawbacks of the EALM in the following sections. We evaluate our model on its ability to represent the long-tail and its effectiveness in continually learning new entities without forgetting old ones.

A.2.1 Evaluation on Long-Tail

We first evaluate our EALM trained using our full catalogues on all our test sets. Entities from the test sets are not manually added to the catalogues and the presence of them is purely coincidental. We have divided our test sets into different groups as shown in Table 1 based on this information. We show the relative reduction in perplexity obtained by our baseline EALM compared to the pre-trained LM, on all our test sets, in the first part of Table 3. We also show a breakdown of these results by entity type obtained by evaluating only on utterances which contain at least one entity of that type. Note that the tail and new test sets only contain entities where each word is seen at least once in the training data. We do this to avoid evaluating on bad transcriptions with misspelled words.

We can see that our baseline EALM consistently outperforms the standalone pre-trained model on all our test sets. The improvement is much more significant on tail entities (10.37%), that are unseen during training, but are present in our catalogues. Surprisingly, we found that our model also does well on new entities, which are not present in our catalogues. We assume this is because of the regularization effect of our entity models on the pre-trained LM, which probably overfits to seen entities. Our pre-trained model performs very poorly on the spanish entities test set as expected, since these entities contain new words not seen during training combined with the fact that our catalogues contain other spanish entities explains the significant boost seen on this test set.

A.2.2 Continual Learning of Popular Entities

Next, we evaluate our model on whether it can continually learn popular entities. To test this, we re-trained all our entity models by adding entities from the tail and new test sets (including dev sets), to the top five percent (ranked by popularity) of our catalogues. We did not retrain the contextual fusion layer. We see a big improvement from 5.63% to 14.38% on utterances containing new entities, refer Table 3. While this is encouraging, we are still significantly worse compared to the performance of our pre-trained LM on the already existing popular entities in the general test set. From Table 2 we can infer that the improvement needed on the entity new test set to match the performance of our pre-trained LM on the entity seen test set is close to 54.83%. However, retraining the pre-trained LM frequently and collecting utterances containing these entities is expensive.

We also evaluated our method on whether it can learn popular entities from other languages, which is not atypical in task-oriented dialogue systems. We can see that our model achieved a huge overall perplexity reduction of 39.12%. Again, this is still significantly lower compared to the 85.8% (refer Table 2) improvement needed to match the performance on existing popular entities.

Notice that even though we see a further reduction in perplexity on the general test set when we retrained our entity models with tail and new entities, because of some possible overlap, we see that our improvement on the general test set reduced from 1.56% to 1.25% after adding the spanish entities. This is expected because the contextual fusion layer was trained on previous entity models. We measure this more accurately in the next section. Unlike other current continual learning approaches, that suffer from significant forgetting, our method only showed a minor degradation on the general test set. This can be explained by Table 6 where we see that the interpolation probabilities of our entity models are almost always capped around 0.05. We have also observed that this number can only reach 0.2 to 0.4 for new/tail entities explaining the difficulty of our system in being able to truly continually learn new entities, and is always limited by the performance of the pre-trained LM on these entities. However, we have seen cases where this number can touch 0.4 to 0.8, in a controlled setting, using fewer entity models, a weaker pre-trained LM and some supervision.
A.2.3 Continual Learning of Tail Entities

Finally, we evaluate whether we can add entities to the tail of our catalogues and still see significant improvements. To test this, we have created three sets of catalogues containing top 25%, 50% and 100% of entities from the original catalogues. To further control our experiments, we removed all entities that are present in the training data and added back 50% of them to these catalogues. We trained two sets of entity models with different random seeds using the 25% catalogue (Entity25, Retrained25) and one set each with the 50% and 100% catalogues (Retrained50, Retrained100). We also artificially created a test set (EntityTailNew) using entities randomly picked from the bottom 75% of our original catalogues. We train our contextual fusion layer using Entity25 and replace it with others for our experiments. We show our results in Table 5.

Our results suggest that replacing the entity models comes at a small cost, since the fusion layer is trained on the previous entity models. However, we are still better than the standalone pre-trained model. We are able to continually learn tail entities but the improvements are smaller compared to learning the popular entities. Note that a lot of our tail entities contain both words that are not seen during training and misspelled words, which explains why our pre-trained model does poorly on these entities compared to the EntityNew test set, refer Table 2.

Table 4: Example showing the relevant utterance context and interpolation probabilities using $k = 4$.

We have created this utterance artificially and it is not taken from our de-identified customer data.

<table>
<thead>
<tr>
<th></th>
<th>Play</th>
<th>a</th>
<th>sky</th>
<th>full</th>
<th>of</th>
<th>stars</th>
<th>by</th>
<th>coldplay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Song Model</td>
<td>&lt;s&gt;</td>
<td>&lt;s&gt;</td>
<td>a</td>
<td>sky</td>
<td>full</td>
<td>of</td>
<td>stars</td>
<td>&lt;s&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;s&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;s&gt;</td>
</tr>
<tr>
<td>Celeb Model</td>
<td>&lt;s&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;s&gt;</td>
</tr>
<tr>
<td>Other Models</td>
<td>&lt;s&gt;</td>
<td>&lt;s&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;s&gt;</td>
</tr>
</tbody>
</table>

Table 5: Table showing percentage reduction in perplexity compared to the pre-trained LM.

<table>
<thead>
<tr>
<th>Test set</th>
<th>EALM trained with Entity25</th>
<th>EALM + Retrained25</th>
<th>EALM + Retrained50</th>
<th>EALM + Retrained100</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>1.55</td>
<td>1.32</td>
<td>1.33</td>
<td>1.33</td>
</tr>
<tr>
<td>EntityTailNew</td>
<td>14.52</td>
<td>14.02</td>
<td>16.95</td>
<td>17.97</td>
</tr>
</tbody>
</table>

Table 6: Table showing the interpolation probabilities of our EALM based on Eq. 7. ‘a sky full of stars’ is a song and ‘a sky full of stars for a roof’ is an album (soundtrack) present in our catalogues.

<table>
<thead>
<tr>
<th></th>
<th>Play</th>
<th>a</th>
<th>sky</th>
<th>full</th>
<th>of</th>
<th>stars</th>
<th>by</th>
<th>coldplay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-trained LM</td>
<td>1</td>
<td>0.92</td>
<td>0.94</td>
<td>0.92</td>
<td>0.93</td>
<td>0.85</td>
<td>0.97</td>
<td>0.97</td>
</tr>
<tr>
<td>Song Model</td>
<td>0</td>
<td>0.02</td>
<td>0.03</td>
<td>0.03</td>
<td>0.03</td>
<td>0.06</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Album Model</td>
<td>0</td>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
<td>0.03</td>
<td>0.08</td>
<td>0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>Celeb Model</td>
<td>0</td>
<td>0.02</td>
<td>0.01</td>
<td>0.02</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>Video Model</td>
<td>0</td>
<td>0.01</td>
<td>0</td>
<td>0.01</td>
<td>0</td>
<td>0.01</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Item Model</td>
<td>0</td>
<td>0.02</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Place Model</td>
<td>0</td>
<td>0</td>
<td>0.01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Contact Model</td>
<td>0</td>
<td>0.01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>