Reference-Aware Language Models

Zichao Yang1,*, Phil Blunsom2,3, Chris Dyer1,2, and Wang Ling2
1Carnegie Mellon University, 2DeepMind, and 3University of Oxford
zichaoy@cs.cmu.edu, {pblunsom,cdyer,lingwang}@google.com

Abstract

We propose a general class of language models that treat reference as discrete stochastic latent variables. This decision allows for the creation of entity mentions by accessing external databases of references (required by, e.g., dialogue generation) or past internal state (required to explicitly model coreferentiality). Beyond simple copying, our coreference model can additionally refer to a referent using varied mention forms (e.g., a reference to "Jane" can be realized as "she"), a characteristic feature of reference in natural languages. Experiments on three representative applications show our model variants outperform models based on deterministic attention and standard language modeling baselines.

1 Introduction

Referring expressions (REs) in natural language are noun phrases (proper nouns, common nouns, and pronouns) that identify objects, entities, and events in an environment. REs occur frequently and they play a key role in communicating information efficiently. While REs are common in natural language, most previous work does not model them explicitly, either treating REs as ordinary words in the model or replacing them with special tokens that are filled in with a post processing step (Wen et al., 2015; Luong et al., 2015). Here we propose a language modeling framework that explicitly incorporates reference decisions. In part, this is based on the principle of pointer networks in which copies are made from another source (Gülçehre et al., 2016; Gu et al., 2016; Ling et al., 2016; Vinyals et al., 2015; Ahn et al., 2016; Merity et al., 2016). However, in the full version of our model, we go beyond simple copying and enable coreferent mentions to have different forms, a key characteristic of natural language reference.

Figure 1 depicts examples of REs in the context of the three tasks that we consider in this work. First, many models need to refer to a list of items (Kiddon et al., 2016; Wen et al., 2015). In the task of recipe generation from a list of ingredients (Kiddon et al., 2016), the generation of the recipe will frequently refer to these items. As shown in Figure 1, in the recipe "Blend soy milk and ...", soy milk refers to the ingredient summaries. Second, reference to a database is crucial in many applications. One example is in task oriented dialogue where access to a database is necessary to answer a user’s query (Young et al., 2013; Li et al., 2016; Vinyals and Le, 2015; Wen et al., 2015; Sordoni et al., 2015; Serban et al., 2016; Bordes and Weston, 2016; Williams and Zweig, 2016; Shang et al., 2015; Wen et al., 2016). Here we consider the domain of restaurant recommendation where a system refers to restaurants (name) and their attributes (address, phone number etc) in its responses. When the system
says “the nirala is a nice restaurant”, it refers to the restaurant name the nirala from the database. Finally, we address references within a document (Mikolov et al., 2010; Ji et al., 2015; Wang and Cho, 2015), as the generation of words will often refer to previously generated words. For instance the same entity will often be referred to throughout a document. In Figure 1, the entity you refers to I in a previous utterance. In this case, copying is insufficient—although the referent is the same, the form of the mention is different.

In this work we develop a language model that has a specific module for generating REs. A series of decisions (should I generate a RE? If yes, which entity in the context should I refer to? How should the RE be rendered?) augment a traditional recurrent neural network language model and the two components are combined as a mixture model. Selecting an entity in context is similar to familiar models of attention (Bahdanau et al., 2014), but rather than being a soft decision that reweights representations of elements in the context, it is treated as a hard decision over contextual elements which are stochastically selected and then copied or, if the task warrants it, transformed (e.g., a pronoun rather than a proper name is produced as output). In cases when the stochastic decision is not available in training, we treat it as a latent variable and marginalize it out. For each of the three tasks, we pick one representative application and demonstrate our reference aware model’s efficacy in evaluations against models that do not explicitly include a reference operation.

Our contributions are as follows:

• We propose a general framework to model reference in language. We consider reference to entries in lists, tables, and document context. We instantiate these tasks into three specific applications: recipe generation, dialogue modeling, and coreference based language models.

• We develop the first neural model of reference that goes being copying and can model (conditional on context) how to form the mention.

• We perform comprehensive evaluation of our models on the three data sets and verify our proposed models perform better than strong baselines.

2 Reference-aware language models

Here we propose a general framework for reference-aware language models.

We denote each document as a series of tokens \( x_1, \ldots, x_L \), where \( L \) is the number of tokens. Our goal is to maximize \( p(x_i \mid c_i) \), the probability of each word \( x_i \) given its previous context \( c_i = x_1, \ldots, x_{i-1} \). In contrast to traditional neural language models, we introduce a variable \( z_i \) at each position, which controls the decision on which source \( x_i \) is generated from. Then the conditional probability is given by:

\[
p(x_i, z_i \mid c_i) = p(x_i \mid z_i, c_i)p(z_i \mid c_i),
\]

where \( z_i \) has different meanings in different contexts. If \( x_i \) is from a reference list or a database, then \( z_i \) is one dimensional and \( z_i = 1 \) denotes \( x_i \) is generated as a reference. \( z_i \) can also model more complex decisions. In coreference based language model, \( z_i \) denotes a series of sequential decisions, such as whether \( x_i \) is an entity, if yes, which entity \( x_i \) refers to. When \( z_i \) is not observed, we will train our model to maximize the marginal probability over \( z_i \), i.e., \( p(x_i \mid c_i) = \sum_{z_i} p(x_i \mid z_i, c_i)p(z_i \mid c_i) \).

2.1 Reference to lists

We begin to instantiate the framework by considering reference to a list of items. Referring to a list of items has broad applications, such as generating documents based on summaries etc. Here we specifically consider the application of recipe generation conditioning on the ingredient lists. Table 1 illustrates the ingredient list and recipe for Spinach and Banana Power Smoothie. We can see that the ingredients soy milk, spinach leaves, and banana occur in the recipe.

![Recipe pointer](image)

Let the ingredients of a recipe be \( X = \{x_i\}_{i=1}^T \) and each ingredient contains \( L \) tokens.
Table 1: Ingredients and recipe for Spinach and Banana Power Smoothie.

<table>
<thead>
<tr>
<th>Ingredients</th>
<th>Recipe</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 cup plain soy milk</td>
<td>Blend soy milk and spinach leaves together in a blender until smooth. Add banana and pulse until thoroughly blended.</td>
</tr>
<tr>
<td>3/4 cup packed fresh spinach leaves</td>
<td></td>
</tr>
<tr>
<td>1 large banana, sliced</td>
<td></td>
</tr>
</tbody>
</table>

To solve this problem, we treat $z_i$ as a latent variable, we wish to maximize the marginal probability of $y_v$ over all possible values of $z_v$. In this way, the model can automatically learn when to refer to tokens in the ingredients. Thus, the probability of generating token $y_v$ is defined as:

$$p(y_v|s_v) = p_{\text{voc}}(y_v)p(0|s_v) + p_{\text{copy}}(y_v)p(1|s_v)$$

$$= p_{\text{voc}}(y_v)(1 - p(1|s_v)) + p_{\text{copy}}(y_v)p(1|s_v).$$

If no string match is found for $y_v$, we simply set $p_{\text{copy}}(y_v) = 0$ in the above objective.

2.2 Reference to databases

We then consider the more complicated task of reference to database entries. Referring to databases is quite common in question answering and dialogue systems, in which databases are external knowledge and they are resorted to reply users’ query. In our paper, we consider the application of task-oriented dialogue systems in the domain of restaurant recommendations. Different from lists that are one dimensional, databases are two-dimensional and referring to table entries requires sophisticated model design.

To better understand the model, we first make a brief introduction of the data set. We use dialogues from the second Dialogue State Tracking Challenge (DSTC2) (Henderson et al., 2014). Table 3 is one example dialogue from this dataset.

We can observe from this example, users get recommendations of restaurants based on queries that specify the area, price and food type of the restaurant. We can support the system’s decisions by incorporating a mechanism that allows the model to query the database to find restaurants that satisfy the users’ queries. A sample of our database (refer to data preparation part on how we construct the database) is shown in Table 2. We can observe that each restaurant contains 6 attributes that are generally referred in the dialogue dataset. As such, if the user requests a restaurant that serves “indian” food, we wish to train a model that can search for entries whose “food”...
M: Hello, welcome to the Cambridge restaurant system? You can ask for restaurants by area, price range or food type. How may I help you?

U: i want a moderately priced restaurant in the north town

M: the nirala is a nice restaurant in the north of town in the moderate price range

U: what is the address

M: Sure, the nirala is on 7 Milton Road Chesterton

Table 3: Example dialogue, M stands for Machine and U stands for User

Table 2: Fragment of database for dialogue system.

<table>
<thead>
<tr>
<th>NAME</th>
<th>PRICE RANGE</th>
<th>FOOD</th>
<th>AREA</th>
<th>ADDRESS</th>
<th>POST CODE</th>
<th>PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>all baba</td>
<td>moderate</td>
<td>lebanese</td>
<td>centre</td>
<td>59 Hills Road City Centre</td>
<td>CB 2, 1 NT</td>
<td>01462 343565</td>
</tr>
<tr>
<td>the nirala</td>
<td>moderate</td>
<td>indian</td>
<td>north</td>
<td>7 Milton Road Chesterton</td>
<td>CB 4, 1 UX</td>
<td>01223 360966</td>
</tr>
</tbody>
</table>

Consider a dialogue with \( T \) turns, the utterances from a user and a machine are denoted as \( X = \{ x_i \}_{i=1}^T \) and \( Y = \{ y_i \}_{i=1}^T \) respectively, where \( i \) is the \( i \)-th utterance. We define \( x_i = \{ x_{ij} \}_{j=1}^{|x_i|} \), \( y_i = \{ y_{iv} \}_{v=1}^{y_i} \), where \( x_{ij} \) \((y_{iv})\) denotes the \( j \)-th \((v\)-th\) token in the \( i \)-th utterance from the user (the machines). The dialogue sequence starts with a machine utterance and is given by \( \{ y_1, x_1, y_2, x_2, \ldots, y_T, x_T \} \). We would like to model the utterances from the machine

\[
p(y_1, y_2, \ldots, y_T | x_1, x_2, \ldots, x_T) = \prod_{i=1}^T p(y_i | y_{<i}, x_{<i}) = \prod_{i,v} p(y_{iv} | y_{<v}, y_{<i}, x_{<i}).
\]

We encode \( y_{<i} \) and \( x_{<i} \) into continuous space in a hierarchical way with LSTM: Sentence Encoder: For a given utterance \( x_i \), we encode it as \( h_{i,j} \equiv LSTM_E(W_E x_{ij}, h_{i,j-1}) \). The representation of \( x_i \) is given by the \( h_i^x = h_{i,|x_i|} \). The same process is applied to obtain the machine utterance representation \( h_i^y = h_{i,|y_i|} \). Turn Encoder: We further encode the sequence \( \{h_1^x, h_2^x, \ldots, h_i^y, h_i^x\} \) with another LSTM encoder. We shall refer the last hidden state as \( u_i \), which can be seen as the hierarchical encoding of the previous \( i \) utterances.

Decoder: We use \( u_{i-1} \) as the initial state of decoder LSTM and decode each token in \( y_i \). We can express the decoder as:

\[
s_{i,v}^y = \text{LSTM}_D(W_E y_{i,v-1}, s_{i,v-1}),
\]

\[
p_{i,v}^y = \text{softmax}(W s_{i,v}^y).
\]

We can also incorporate the attention mechanism in the decoder. As shown in Figure 3, we use the attention mechanism over the utterance in the previous turn. Due to space limit, we don’t present the attention based decoder mathematically and readers can refer to (Bahdanau et al., 2014) for details.

2.2.1 Incorporating Table Reference

We now extend the decoder in order to allow the model to condition the generation on a database. Pointer Switch: We use \( z_{i,v} \in \{0, 1\} \) to denote the decision of whether to copy one cell from the table. We compute this probability as follows:

\[
p(z_{i,v} | s_{i,v}) = \text{sigmoid}(W s_{i,v}).
\]

Thus, if \( z_{i,v} = 1 \), the next token \( y_{i,v} \) is generated from the database, whereas if \( z_{i,v} = 0 \), then the following token is generated from a softmax. We now describe how we generate tokens from the database.
We denote a table with $R$ rows and $C$ columns as $\{t_{r,c}\}, r \in [1, R], c \in [1, C]$, where $t_{r,c}$ is the cell in row $r$ and column $c$. The attribute of each column is denoted as $s_c$, where $c$ is the $c$-th attribute. $t_{r,c}$ and $s_c$ are one-hot vector.

**Table Encoding:** To encode the table, we first build an attribute vector and then an encoding vector for each cell. The attribute vector is simply an embedding lookup $g_c = W_E s_c$. For the encoding of each cell, we first concatenate embedding lookup of the cell with the corresponding attribute vector $g_c$ and then feed it through a one-layer MLP as follows: then $e_{r,c} = \text{tanh}(W [W_E t_{r,c}, g_c])$.

**Table Pointer:** The detailed process of calculating the probability distribution over the table is shown in Figure 4. The attention over cells in the table is conditioned on a given vector $q$, similarly to the attention model for sequences. However, rather than a sequence of vectors, we now operate over a table.

**Step 1:** Attention over the attributes to find out the attributes that a user asks about, $p^a = \text{ATTN}(\{g_c\}, q)$. Suppose a user says cheap, then we should focus on the price attribute.

**Step 2:** Conditional row representation calculation, $e_r = \sum_c p^a_c e_{r,c} \ \forall r$. So that $e_r$ contains the price information of the restaurant in row $r$.

**Step 3:** Attention over $e_r$ to find out the restaurants that satisfy users’ query, $p^r = \text{ATTN}(\{e_r\}, q)$. Restaurants with cheap price will be picked.

**Step 4:** Using the probabilities $p^r$, we compute the weighted average over the all rows $e_c = \sum_r p^r_r e_{r,c}$. $\{e_r\}$ contains the information of cheap restaurant.

**Step 5:** Attention over columns $\{e_r\}$ to compute the probabilities of copying each column $p^c = \text{ATTN}(\{e_r\}, q)$.

**Step 6:** To get the probability matrix of copying each cell, we simply compute the outer product $p^{copy} = p^r \otimes p^c$.

The overall process is as follows:

$$p^a = \text{ATTN}(\{g_c\}, q),$$

$$e_r = \sum_c p^a_c e_{r,c} \ \forall r,$$

$$p^r = \text{ATTN}(\{e_r\}, q),$$

$$e_c = \sum_r p^r_r e_{r,c} \ \forall c,$$

$$p^c = \text{ATTN}(\{e_c\}, q),$$

$$p^{copy} = p^r \otimes p^c.$$

If $z_{i,v} = 1$, we embed the above attention process in the decoder by replacing the conditioned state $q$ with the current decoder state $s_{i,v}$.

**Objective:** As in previous task, we can train the model in a fully supervised fashion, or we can treat the decision as a latent variable. We can get $p(y_{i,v}|s_{i,v})$ in a similar way.

### 2.3 Reference to document context

Finally, we address the references that happen in a document itself and build a language model that uses coreference links to point to previous entities. Before generating a word, we first make the decision on whether it is an entity mention. If so, we decide which entity this mention belongs to, then we generate the word based on that entity. Denote the document as $X = \{x_i\}_{i=1}^L$, and the entities are $E = \{e_i\}_{i=1}^N$, each entity has $M_i$ mentions, $e_i = \{m_{ij}\}_{j=1}^{M_i}$, such that $\{x_{m_{ij}}\}_{j=1}^{M_i}$ refer to the same entity. We use a LSTM to model the document, the hidden state of each token is $h_i = \text{LSTM}(W_E x_i, h_{i-1})$. We use a set $h_e = \{h^e_{i0}, h^e_{i1}, ..., h^e_{IM_i}\}$ to keep track of the entity states, where $h^e_j$ is the state of entity $j$.

**Word generation:** At each time step before generating the next word, we predict whether the word is an entity mention:

$$p^{\text{coref}}(v_i|h_{i-1}, h^e) = \text{ATTN}(h^e, h_{i-1}),$$

$$d_i = \sum_{v_i} p(v_i|h_{i-1}) h^e_{vi},$$

$$p(z_i|h_{i-1}) = \text{sigmoid}(W[h_{i-1}, d_i]),$$

where $z_i$ denotes whether the next word is an entity and if yes $v_i$ denotes which entity the next word corefers to. If the next word is an entity mention, then $p(x_i|v_i, h_{i-1}, h^e) =$
um and [I] think that is what - Go ahead [Linda]2. Well and thanks goes to [you]1 and to [the media]3 to help [us]4...So [our]4 hat is off to all of [you]s...

**Entity state update:** Since there are multiple mentions for each entity and the mentions appear dynamically, we need to keep track of the entity state in order to use coreference information in entity mention prediction. We update the entity state \( h^c \) at each time step. In the beginning, \( h^c = \{h_0^0\} \), \( h_0^0 \) denotes the state of an empty entity and is a learnable variable. If \( z_i = 1 \) and \( v_i = 0 \), then it indicates the next word is a new entity mention, then in the next step, we append \( h_i \) to \( h^c \), i.e., \( h^c = \{h^c, h_i\} \), if \( z_i = 1 \) and \( v_i > 0 \), then we update the corresponding entity state with the new hidden state, \( h^c[v_i] = h_i \). Another way to update the entity state is to use one LSTM to encode the mention states and get the new entity state. Here we use the latest entity mention state as the new entity state for simplicity. The detailed update process is shown in Figure 5.

Note that the stochastic decisions in this task are more complicated than previous two tasks. We need to make two sequential decisions: whether the next word is an entity mention, and if yes, which entity the mention corefers to. It is intractable to marginalize these decisions, so we train this model in a supervised fashion (refer to data preparation part on how we get coreference annotations).

3 Experiments

3.1 Data sets and preprocessing

**Recipes:** We crawled all recipes from www.allrecipes.com. There are about 31,000 recipes in total, and every recipe has an ingredient list and a corresponding recipe. We exclude the recipes that have less than 10 tokens or more than 500 tokens, those recipes take about 0.1% of all data set. On average each recipe has 118 tokens and 9 ingredients. We random shuffle the whole data set and take 80% as training and 10% for validation and test. We use a vocabulary size of 10,000 in the model.

**Dialogue:** We use the DSTC2 data set. We only use the dialogue transcripts from the data set. There are about 3,200 dialogues in total. The table is not available from DSTC2. To reconstruct the table, we crawled TripAdvisor for restaurants in the Cambridge area, where the dialog dataset was collected. Then, we remove restaurants that do not appear in the data set and create a database with 109 restaurants and their attributes (e.g. food type). Since this is a small data set, we use 5-fold cross validation and report the average result over the 5 partitions. There may be multiple tokens in each table cell, for example in Table 2, the name, address, post code and phone number have multiple tokens, we replace them with one special token. For the name, address, post code and phone number number have multiple tokens, we replace them with one special token. For the name, address, post code and phone number number have multiple tokens, we replace them with one special token.

**Coref LM:** We use the Xinhua News data set from Gigaword Fifth Edition and sample 100,000 documents that has length in range from 100 to 500. Each document has on average 234 tokens, so there are 23 million tokens in total. We process...
the documents to get coreference annotations and use the annotations, i.e., \( z_i, t_i \), in training. We take 80% as training and 10% as validation and test respectively. We ignore the entities that have only one mention and for the mentions that have multiple tokens, we take the token that is most frequent in the all the mentions for this entity. After preprocessing, tokens that are entity mentions take about 10% of all tokens. We use a vocabulary size of 50,000 in the model.

3.2 Baselines, model training and evaluation

We compare our model with baselines that do not model reference explicitly. For recipe generation and dialogue modeling, we compare our model with basic seq2seq and attention model. We also apply attention mechanism over the table for dialogue modeling as a baseline. For coreference based language model, we compare our model with simple RNN language model.

We train all models with simple stochastic gradient descent with gradient clipping. We use a one-layer LSTM for all RNN components. Hyperparameters are selected using grid search based on the validation set.

Evaluation of our model is challenging since it involves three rather different applications. We focus on evaluating the accuracy of predicting the reference tokens, which is the goal of our model. Specifically, we report the perplexity of all words, words that can be generated from reference and non-reference words. The perplexity is calculated by multiplying the probability of decision at each step all together. Note that for non-reference words, they also appear in the vocabulary. So it is a fair comparison to models that do not model reference explicitly. For the recipe task, we also generate the recipes using beam size of 10 and evaluate the generated recipes with BLEU. We didn’t use BLEU for dialogue generation since the database entries take only a very small part of all tokens in utterances.

3.3 Results and analysis

The results for recipe generation, dialogue and coref based language model are shown in Table 4, 5, and 6 respectively. The recipe results in Table 4 verifies that modeling reference explicitly improves performance. Latent and Pointer perform better than Seq2Seq and Attn model. The Latent model performs better than the Pointer model since tokens in ingredients that match with recipes do not necessarily come from the ingredients. Imposing a supervised signal gives wrong information to the model and hence makes the result worse. With latent decision, the model learns to when to copy and when to generate it from the vocabulary.

The findings for dialogue basically follow that of recipe generation, as shown in Table 5. Conditioning table performs better in predicting table tokens in general. Table Pointer has the lowest perplexity for tokens in the table. Since the table tokens appear rarely in the dialogue transcripts, the overall perplexity does not differ much and the non-table token perplexity are similar. With attention mechanism over the table, the perplexity of table token improves over basic Seq2Seq model, but still not as good as directly pointing to cells in the table, which shows the advantage of modeling reference explicitly. As expected, using sentence attention improves significantly over models without sentence attention. Surprisingly, Table Latent performs much worse than Table Pointer. We also measure the perplexity of table tokens that appear only in test set. For models other than Table Pointer, because the tokens never appear in the training set, the perplexity is quite high, while Table Pointer can predict these tokens much more accurately. This verifies our conjecture that our model can learn reasoning over databases.

The coref based LM results are shown in Table 6. We find that coref based LM performs much better on the entity perplexity, but is a little bit worse for non-entity words. We found it was an optimization problem and the model was stuck in a local optimum. So we initialize the Pointer model with the weights learned from LM, the Pointer model performs better than LM both for entity perplexity and non-entity words perplexity.

In Appendix A, we also visualize the heat map of table reference and list items reference. The visualization shows that our model can correctly predict when to refer to which entries according to context.

4 Related Work

In terms of methodology, our work is closely related to previous works that incorporate copying mechanism with neural models (Gülçehre et al., 2016; Gu et al., 2016; Ling et al., 2016; Vinyals et al., 2015). Our models are similar to models proposed in (Ahn et al., 2016; Merity et al., 2016),
Table 4: Recipe results, evaluated in perplexity and BLEU score. All means all tokens, Ing denotes tokens from recipes that appear in ingredients. Word means non-table tokens. Pointer and Latent differs in that for Pointer, we provide supervised signal on when to generate a reference token, while in Latent it is a latent decision.

<table>
<thead>
<tr>
<th>Model</th>
<th>All Ing Word</th>
<th>All Ing Word</th>
<th>All Ing Word</th>
<th>All Ing Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq</td>
<td>5.60 11.26</td>
<td>5.00 14.07</td>
<td>5.52 11.26</td>
<td>4.91 14.39</td>
</tr>
<tr>
<td>Attn</td>
<td>5.25 6.86</td>
<td>5.03 14.84</td>
<td>5.19 6.92</td>
<td>4.95 15.15</td>
</tr>
<tr>
<td>Pointer</td>
<td>5.15 5.86</td>
<td>5.04 15.06</td>
<td>5.11 6.04</td>
<td>4.98 15.29</td>
</tr>
<tr>
<td>Latent</td>
<td>5.02 5.10</td>
<td>5.01 14.87</td>
<td>4.97 5.19</td>
<td>4.94 15.41</td>
</tr>
</tbody>
</table>

Table 5: Dialogue perplexity results. Table means tokens from table, Table OOV denotes table tokens that do not appear in the training set. **Sentence Attn** denotes we use attention mechanism over tokens in utterances from the previous turn.

<table>
<thead>
<tr>
<th>Model</th>
<th>All Table</th>
<th>Table OOV</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq</td>
<td>1.35 ± 0.01</td>
<td>4.98 ± 0.38</td>
<td>1.99E7 ± 7.75E6</td>
</tr>
<tr>
<td>Table Attn</td>
<td>1.37 ± 0.01</td>
<td>5.09 ± 0.64</td>
<td>7.91E7 ± 1.39E8</td>
</tr>
<tr>
<td>Table Pointer</td>
<td>1.33 ± 0.01</td>
<td>3.99 ± 0.36</td>
<td>1360 ± 2600</td>
</tr>
<tr>
<td>Table Latent</td>
<td>1.36 ± 0.01</td>
<td>4.99 ± 0.20</td>
<td>3.78E7 ± 6.08E7</td>
</tr>
</tbody>
</table>

+ **Sentence Attn**

| Seq2Seq   | 1.28 ± 0.01 | 3.31 ± 0.21 | 2.83E9 ± 4.69E9 |
| Table Attn| 1.28 ± 0.01 | 3.17 ± 0.21 | 1.67E7 ± 9.5E6 |
| Table Pointer | 1.27 ± 0.01 | 2.99 ± 0.19 | 82.86 ± 110 |
| Table Latent | 1.28 ± 0.01 | 3.26 ± 0.25 | 1.27E7 ± 1.41E7 |

Table 6: Coreference based LM. Pointer + init means we initialize the model with the LM weights.

<table>
<thead>
<tr>
<th>Model</th>
<th>All</th>
<th>val</th>
<th>Entity</th>
<th>Word</th>
<th>All</th>
<th>test</th>
<th>Entity</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM</td>
<td>33.08</td>
<td>44.52</td>
<td>32.04</td>
<td>32.04</td>
<td>33.08</td>
<td>43.86</td>
<td>32.10</td>
<td></td>
</tr>
<tr>
<td>Pointer</td>
<td>32.57</td>
<td>32.07</td>
<td>32.62</td>
<td>32.62</td>
<td>32.57</td>
<td>32.07</td>
<td>32.69</td>
<td></td>
</tr>
<tr>
<td>Pointer + init</td>
<td><strong>30.43</strong></td>
<td><strong>28.56</strong></td>
<td><strong>30.63</strong></td>
<td><strong>30.42</strong></td>
<td><strong>28.56</strong></td>
<td><strong>30.66</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where the generation of each word can be conditioned on a particular entry in knowledge lists and previous words. In our work, we describe a model with broader applications, allowing us to condition, on databases, lists and dynamic lists.

In terms of applications, our work is related to chit-chat dialogue (Li et al., 2016; Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016; Shang et al., 2015) and task oriented dialogue (Wen et al., 2015; Bordes and Weston, 2016; Williams and Zweig, 2016; Wen et al., 2016). Most of previous works on task oriented dialogues embed the seq2seq model in traditional dialogue systems, in which the table query part is not differentiable, while our model queries the database directly. Recipe generation was proposed in (Kiddon et al., 2016). They use attention mechanism over the checklists, whereas our work models explicit references to them. Context dependent language models (Mikolov et al., 2010; Jozefowicz et al., 2016; Mikolov et al., 2010; Ji et al., 2015; Wang and Cho, 2015) are proposed to capture long term dependency of text. There are also lots of works on coreference resolution (Haghighi and Klein, 2010; Wiseman et al., 2016). We are the first to combine coreference with language modeling, to the best of our knowledge.

5 Conclusion

We introduce reference-aware language models which explicitly model the decision of from where to generate the token at each step. Our model can also learns the decision by treating it as a latent variable. We demonstrate on three applications, table based dialogue modeling, recipe generation and coref based LM, that our model performs better than attention based model, which does not incorporate this decision explicitly. There are several directions to explore further based on our framework. The current evaluation method is based on perplexity and BLEU. In task oriented dialogues, we can also try human evaluation to see if the model can reply users’ query accurately. It is also interesting to use reinforcement learning to learn the actions in each step in coref based LM.
References


