Reference-Aware Language Models

Zichao Yang[1*], **Phil Blunsom**[2,3], **Chris Dyer**[1,2], **and Wang Ling**[2]
[1]Carnegie Mellon University, [2]DeepMind, and [3]University 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 referents (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.

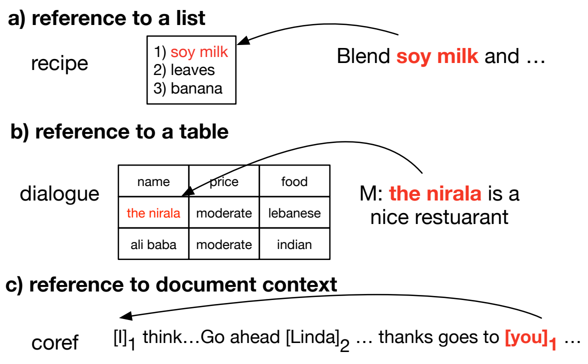

Figure 1: Reference-aware language models.

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

[*]Work completed at DeepMind.

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 $x_1, \dots, x_L$, 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 $c_i = x_1, \dots, x_{i-1}$. 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:

$$p(x_i, z_i \mid c_i) = p(x_i \mid z_i, c_i)p(z_i \mid c_i), \quad (1)$$

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., $p(x_i|c_i) = \sum_{z_i} p(x_i|z_i, c_i)p(z_i|c_i)$.

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.

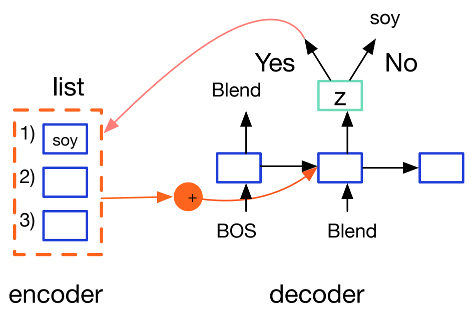

Figure 2: Recipe pointer

Let the ingredients of a recipe be $X = \{x_i\}_{i=1}^T$ and each ingredient contains L tokens $x_i =$

Ingredients	Recipe
1 cup plain `soy milk`	Blend `soy milk` and `spinach leaves`
3/4 cup packed fresh `spinach leaves`	together in a blender until smooth. Add
1 large `banana`, sliced	`banana` and pulse until thoroughly blended.

Table 1: Ingredients and recipe for *Spinach and Banana Power Smoothie*.

$\{x_{ij}\}_{j=1}^{L}$. The corresponding recipe is $y = \{y_v\}_{v=1}^{K}$. We would like to model $p(y|X) = \Pi_v p(y_v|X, y_{<v})$.

We first use a LSTM (Hochreiter and Schmidhuber, 1997) to encode each ingredient: $h_{i,j} = \text{LSTM}_\text{E}(W_E x_{ij}, h_{i,j-1})\ \ \forall i$. Then, we sum the resulting final state of each ingredient to obtain the starting LSTM state of the decoder. We use an attention based decoder:

$$\begin{aligned}
s_v &= \text{LSTM}_\text{D}([W_\text{E} y_{v-1}, d_{v-1}], s_{v-1}),\\
p_v^{\text{copy}} &= \text{ATTN}(\{\{h_{i,j}\}_{i=1}^{T}\}_{j=1}^{L}, s_v),\\
d_v &= \sum_{ij} p_{v,i,j} h_{i,j},\\
p(z_v|s_v) &= \text{sigmoid}(W[s_v, d_v]),\\
p_v^{\text{vocab}} &= \text{softmax}(W[s_v, d_v]),
\end{aligned}$$

where $\text{ATTN}(h, q)$ is the attention function that returns the probability distribution over the set of vectors h, conditioned on any input representation q. A full description of this operation is described in (Bahdanau et al., 2014). The decision to copy from the ingredient list or generate a new word from the softmax is performed using a switch, denoted as $p(z_v|s_v)$. We can obtain a probability distribution of copying each of the words in the ingredients by computing $p_v^{\text{copy}} = \text{ATTN}(\{\{h_{i,j}\}_{i=1}^{T}\}_{j=1}^{L}, s_v)$ in the attention mechanism.

Objective: We can obtain the value of z_v through a string match of tokens in recipes with tokens in ingredients. If a token appears in the ingredients, we set $z_v = 1$ and $z_v = 0$ otherwise. We can train the model in a fully supervised fashion, i.e., we can obtain the probability of y_v as $p(y_v, z_v|s_v) = p_v^{\text{copy}}(y_v)p(1|s_v)$ if $z_v = 1$ and $p_v^{\text{vocab}}(y_v)(1 - p(1|s_{i,v}))$ otherwise.

However, it may be not be accurate. In many cases, the tokens that appear in the ingredients do not specifically refer to ingredients tokens. For examples, the recipe may start with “Prepare a cup of water”. The token “cup” does not refer to the “cup” in the ingredient list “1 cup plain soy milk”. 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:

$$\begin{aligned}
p(y_v|s_v) &= p_v^{\text{vocab}}(y_v)p(0|s_v) + p_v^{\text{copy}}(y_v)p(1|s_v)\\
&= p_v^{\text{vocab}}(y_v)(1 - p(1|s_v)) + p_v^{\text{copy}}(y_v)p(1|s_v).
\end{aligned}$$

If no string match is found for y_v, we simply set $p_v^{\text{copy}}(y_v) = 0$ 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”

NAME	PRICE RANGE	FOOD	AREA	ADDRESS	POST CODE	PHONE
ali baba	moderate	lebanese	centre	59 Hills Road City Centre	CB 2, 1 NT	01462 432565
the nirala	moderate	indian	north	7 Milton Road Chesterton	CB 4, 1 UY	01223 360966

Table 2: Fragment of database for dialogue system.

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

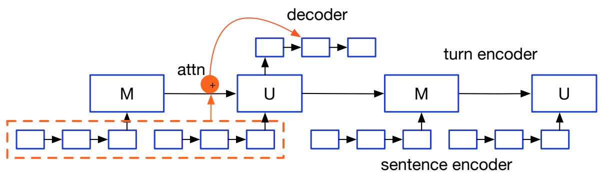

Figure 3: Hierarchical RNN Seq2Seq model. The red box denotes attention mechanism over the utterances in the previous turn.

column contains "indian". Now, we describe how we deploy a model that fulfills these requirements. We first introduce the basic dialogue framework in which we incorporates the table reference module.

Basic Dialogue Framework: We build a basic dialogue model based on the hierarchical RNN model described in (Serban et al., 2016), as in dialogues, the generation of the response is not only dependent on the previous sentence, but on all sentences leading to the response. We assume that a dialogue is alternated between a machine and a user. An illustration of the model is shown in Figure 3.

Consider a dialogue with T turns, the utterances from a user and a machines 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 $\{y_1, x_1, y_2, x_2, \dots, y_T, x_T\}$. We would like to model the utterances from the machine

$$p(y_1, y_2, \dots, y_T | x_1, x_2, \dots, x_T) = \prod_i p(y_i | y_{<i}, x_{<i}) = \prod_{i,v} p(y_{i,v} | y_{i,<v}, y_{<i}, x_{<i}).$$

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 $h^x_{i,j} = \text{LSTM}_\text{E}(W_E x_{i,j}, h^x_{i,j-1})$. The representation of x_i is given by the $h^x_i = h^x_{i,|x_i|}$. The same process is applied to obtain the machine utterance representation $h^y_i = h^y_{i,|y_i|}$. **Turn Encoder**: We further encode the sequence $\{h^y_1, h^x_1, ..., h^y_i, h^x_i\}$ 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:

$$s^y_{i,v} = \text{LSTM}_\text{D}(W_E y_{i,v-1}, s_{i,v-1}),$$
$$p^y_{i,v} = \text{softmax}(W s^y_{i,v}).$$

We can also incoroprate the attetionn 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 mathmatically 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.

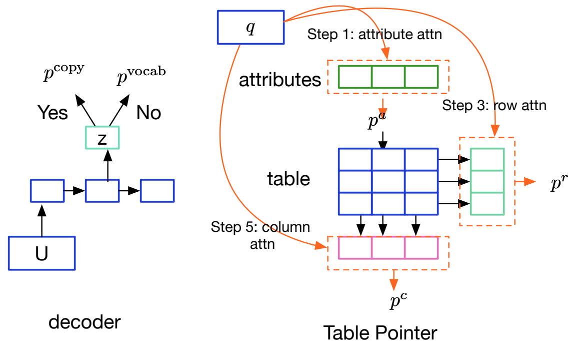

Figure 4: Decoder with table pointer.

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 $e_{r,c} = \tanh(W[W_E t_{r,c}, g_c])$.

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, $e_r = \sum_c p^a_c e_{r,c} \quad \forall r$. 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 $p^c = \text{ATTN}(\{e_c\}, q)$.

Step 6: To get the probability matrix of copying each cell, we simply compute the outer product $p^{\text{copy}} = p^r \otimes p^c$.
The overall process is as follows:

$$\begin{aligned} p^a &= \text{ATTN}(\{g_c\}, q), \\ e_r &= \sum_c p^a_c e_{r,c} \quad \forall r, \\ p^r &= \text{ATTN}(\{e_r\}, q), \\ e_c &= \sum_r p^r_r e_{r,c} \quad \forall c, \\ p^c &= \text{ATTN}(\{e_c\}, q), \\ p^{\text{copy}} &= p^r \otimes p^c. \end{aligned}$$

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 $s^y_{i,v}$.
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 $h^e = \{h^e_0, h^e_1, ..., h^e_M\}$ 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:

$$\begin{aligned} p^{\text{coref}}(v_i|h_{i-1}, h^e) &= \text{ATTN}(h^e, h_{i-1}), \\ d_i &= \sum_{v_i} p(v_i) h^e_{v_i}, \\ p(z_i|h_{i-1}) &= \text{sigmoid}(W[h_{i-1}, d_i]), \end{aligned}$$

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]$_1$ think that is whats - 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]$_5$...

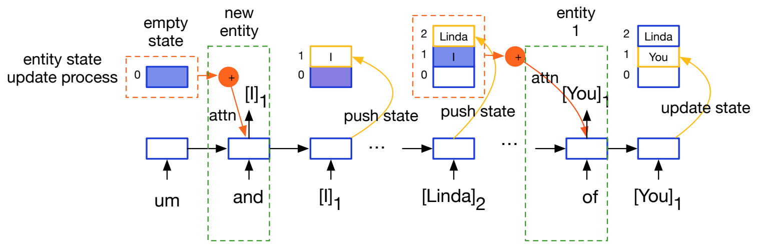

Figure 5: Coreference based language model, example taken from Wiseman et al. (2016).

$\text{softmax}(W_1 \tanh(W_2[h_{i-1}, h^e_{v_i}]))$ else $p(x_i|h_{i-1}) = \text{softmax}(W_1 h_{i-1})$. Hence,

$$p(x_i|x_{<i}) = \begin{cases} p(x_i|h_{i-1})p(z_i|h_{i-1}, h^e) & \text{if } z_i = 0. \\ p(x_i|v_i, h_{i-1}, h^e) \times \\ \quad p^{\text{coref}}(v_i|h_{i-1}, h^e)p(z_i|h_{i-1}, h^e) & \text{if } z_i = 1. \end{cases}$$

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^e at each time step. In the beginning, $h^e = \{h^e_0\}$, h^e_0 denotes the state of an virtual 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^e, i.e., $h^e = \{h^e, h_i\}$, if $z_i = 1$ and $v_i > 0$, then we update the corresponding entity state with the new hidden state, $h^e[v_i] = h_i$. 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 of the j-th row, we replace the tokens in each cell with _NAME_j, _ADDR_j, _POSTCODE_j, _PHONE_j. If a table cell is empty, we replace it with an empty token _EMPTY. We do a string match in the transcript and replace the corresponding tokens in transcripts from the table with the special tokens. Each dialogue on average has 8 turns (16 sentences). We use a vocabulary size of 900, including about 400 table tokens and 500 words.

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, v_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. Hyper-parameters 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),

Model	val				test			
	PPL			BLEU	PPL			BLEU
	All	Ing	Word		All	Ing	Word	
Seq2Seq	5.60	11.26	**5.00**	14.07	5.52	11.26	**4.91**	14.39
Attn	5.25	6.86	5.03	14.84	5.19	6.92	4.95	15.15
Pointer	5.15	5.86	5.04	**15.06**	5.11	6.04	4.98	15.29
Latent	**5.02**	**5.10**	5.01	14.87	**4.97**	**5.19**	4.94	**15.41**

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.

Model	All	Table	Table OOV	Word
Seq2Seq	1.35±0.01	4.98±0.38	1.99E7±7.75E6	1.23±0.01
Table Attn	1.37±0.01	5.09±0.64	7.91E7±1.39E8	1.24±0.01
Table Pointer	**1.33±0.01**	**3.99±0.36**	**1360 ± 2600**	**1.23±0.01**
Table Latent	1.36±0.01	4.99±0.20	3.78E7±6.08E7	1.24±0.01
+ Sentence Attn				
Seq2Seq	1.28±0.01	3.31±0.21	2.83E9 ± 4.69E9	**1.19±0.01**
Table Attn	1.28±0.01	3.17±0.21	1.67E7±9.5E6	1.20±0.01
Table Pointer	**1.27±0.01**	**2.99±0.19**	**82.86±110**	1.20±0.01
Table Latent	1.28±0.01	3.26±0.25	1.27E7±1.41E7	1.20±0.01

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.

Model	val			test		
	All	Entity	Word	All	Entity	Word
LM	33.08	44.52	32.04	33.08	43.86	32.10
Pointer	32.57	32.07	32.62	32.62	32.07	32.69
Pointer + init	**30.43**	**28.56**	**30.63**	**30.42**	**28.56**	**30.66**

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

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

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *CoRR*, abs/1608.00318.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *CoRR*, abs/1603.08148.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Jason Williams. 2014. Dialog state tracking challenge 2 & 3.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2015. Document context language models. *arXiv preprint arXiv:1511.03962*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proc. EMNLP*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. In *Proc. EMNLP*.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. In *Proc. ACL*.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 11–19.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proc. NAACL*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proc. NIPS*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *Proc. ICML Deep Learning Workshop*.

Tian Wang and Kyunghyun Cho. 2015. Larger-context language modelling. *arXiv preprint arXiv:1511.03729*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-hao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proc. EMNLP*.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning global features for coreference resolution. *arXiv preprint arXiv:1604.03035*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.