- Neural Response Generation via GAN with an Approximate Embedding Layer*
- Commonsense Knowledge Aware Conversation Generation with Graph Attention

- Neural Response Generation via GAN with an Approximate Embedding Layer*

Introduction

- "safe response" problem. This is due to the fundamental nature of statistical models, which fit sufficiently observed examples better than insufficiently observed ones.
- This paper presents a Generative Adversarial Network(GAN) to model single turn short-text conversation.
- The proposed method introduces an approximate embedding layer to solve the non-differentiable problem caused by the sampling-based output decoding procedure in the Seq2Seq generative model.

Model Overview

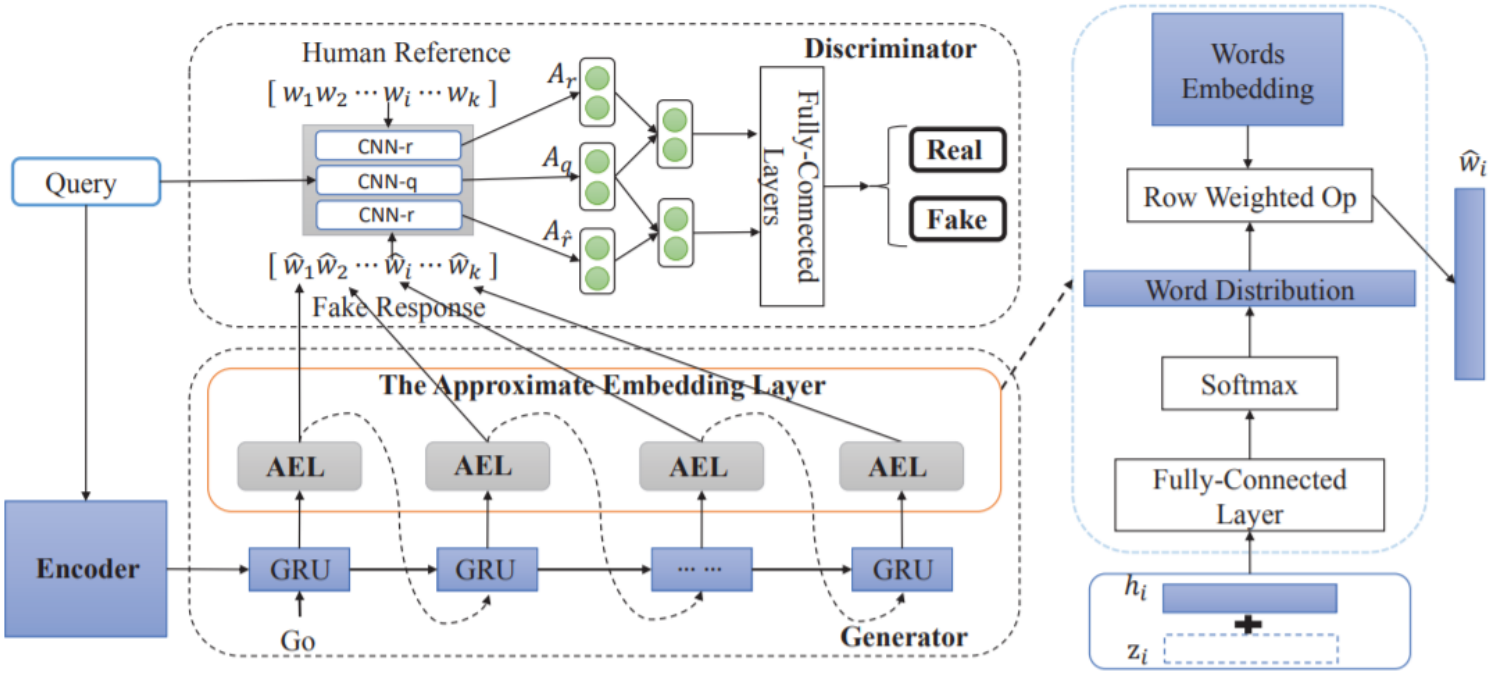


Figure 1: The Framework of GAN for the Response Generator.

The whole framework consists of a response generator G, a discriminator D and an embedding approximation layer that connects the G and the D.

Model Overview

- The generator adopts the Gated Recurrent Unit (GRU) based encoder decoder architecture.
- An approximate embedding layer is designed to guarantee that the response generation procedure is continuous and differentiable, serving as an interface for the discriminator to propagate its loss to the generator.
- The Convolutional Neural Network (CNN) based discriminator is attached on top of the approximation layer, which aims to distinguish the fake responses output by the approximation layer and the corresponding human-generated references.

Pre-training the Generator by MLE

- The generator estimates the probability of each word occurring in r conditioned on q_v.

$$p(r|q) = \prod_{t=1}^{K} p(w_{r,t}|q_v, w_{r,1}, ..., w_{r,t-1})$$

- The generator is trained by optimizing the MLE objective defined as:

$$\frac{1}{|D|} \sum_{(q,r)\in D} \sum_{t=1}^{K} logp(w_{r,t}|q_v, w_{r,1}, ..., w_{r,t-1})$$

- We need to pre-train the generator to guarantee the generator produce grammatical utterances.

Pre-training the CNN-based Discriminator

- V_q: word embedding vector sequence for a query q.
- V_r: word embedding vector sequence for a human-produced response r.
- $V_{\hat{r}}$: word embedding vector sequence for a fake response $\hat{r}$.
- Two CNNs with shared parameters are employed to encode V_r and $V_{\hat{r}}$ into higher-level abstractions, respectively. In addition, a separate CNN is used to abstract V_q.
- We denote such abstraction layers in the above CNNs as A_r,$A_{\hat{r}}$,A_q. Then we concatenate them and feed the resulting vectors to their respective fully-connected layers.
- The Discriminator D is pre-trained by maximising the following objective function:

$$D_{loss} = logD(r|q) + log(1 - D(\hat{r}|q))$$

The Approximate Embedding Layer

- This approximation is based on the assumption that ideally the word distributions should be trained to reasonably approach the one-hot representation of the discrete words.
- The overall word embedding approximation is computed as:

$$\hat{e} = \sum_{j=1}^{V} e_j \cdot Softmax(W_p(h_i + z_i) + b_p)_j$$

- Where W_p and b_p are the weight and bias parameters of the word projection layer, respectively, and h_i is the hidden representation of word w_i.

Adversial Training of the Generator

- Firstly, when training G, we replace the objective function with the l_2-loss between A_r and $A_{\hat{r}}$.
- Secondly, we freeze the parameters of the encoder network and the projection layer of the decoder network, but only tune the parameters of decoder's hidden layers.
- The gradient of the generator can be computed as:

$$\nabla_{gD,G(\theta_G)} = \frac{\partial G_{loss}}{\partial V_{\hat{r}}} \frac{\partial V_{\hat{r}}}{\partial \theta_G} = \frac{\partial G_{loss}}{\partial V_{\hat{r}}} \frac{\partial V_{\hat{r}}}{\partial G} \frac{\partial G}{\partial \theta_G}$$

- Where θ_G denotes the active parameters of the generator G, $G_{loss} = ||A_r - A_{\hat{r}||}$ and $gD, G(*)$ stands for the inference step of the entire GAN.

Automatic Evaluation

Model	Relevance			Diversity		
	Average	**Greedy**	**Extreme**	**Dist-1**	**Dist-2**	**Novelty**
Seq2Seq	0.720	0.614	0.571	0.0037	0.0121	0.0102
MMI-anti	0.713	0.592	0.552	0.0127	0.0495	0.0250
Adver-REGS	0.722	0.660	0.574	0.0153	0.0658	0.0392
GAN-AEL	**0.736**	**0.689**	**0.580**	**0.0214**	**0.0963**	**0.0635**

Table 1: Relevance and diversity evaluation on the Tieba dataset.

Model	Relevance			Diversity		
	Average	**Greedy**	**Extreme**	**Dist-1**	**Dist-2**	**Novelty**
Seq2Seq	0.719	0.578	0.505	0.0054	0.0141	0.0045
MMI-anti	0.710	0.569	0.499	0.0175	0.0586	0.0097
Adver-REGS	0.726	0.590	0.507	0.0223	0.0725	0.0147
GAN-AEL	**0.734**	**0.621**	**0.514**	**0.0296**	**0.0955**	**0.0216**

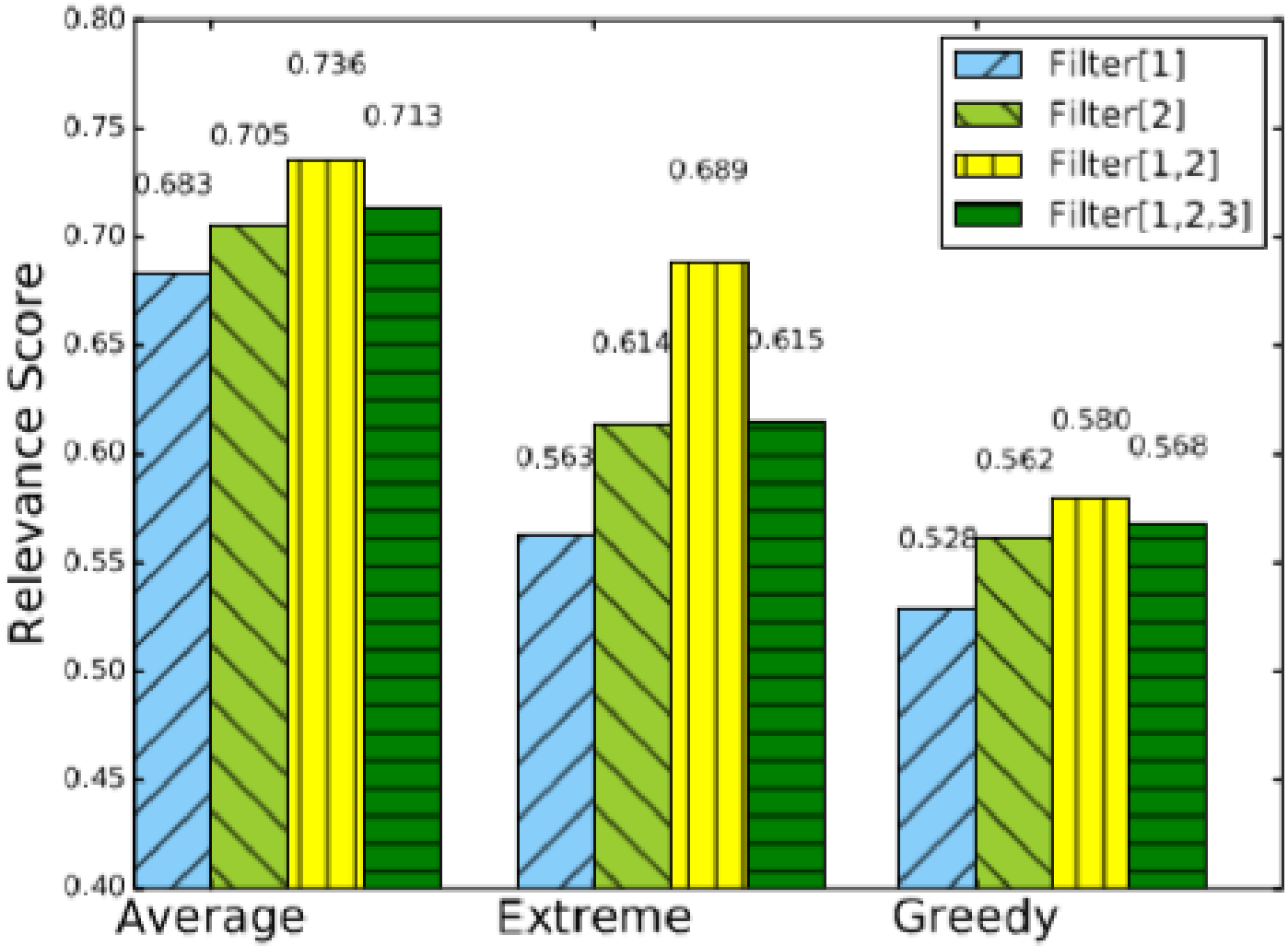
Table 2: Relevance and diversity evaluation on the OpenSubtitles dataset.

Human Evaluation Results

GAN-AEL vs Adver-REGS		
Wins	Losses	Ties
0.61	0.13	0.26

Table 3: Evaluations of GAN-AEL and Adver-REGS based on human subjects,

The Inuence of the Discriminator to Adversarial Training



Example responses

Example responses generated based on the OpenSubtitle data	Example responses generated based on the Baidu Tieba data
Query: You swore an oath when you put that uniform on. **Seq2Seq:** I don't know what to do. **GAN-AEL:** I think I had a good reason.	**Query:** 你喜欢猫还是狗？Do you like cats or dogs? **Seq2Seq:** 喜欢养猫。I Like cats. **GAN-AEL:** 我喜欢金毛。I like golden retriever.
Query: Entire town knows your son is a goon. **Seq2Seq:** What do you mean? **GAN-AEL:** His life is really a sad story.	**Query:** 你像奥巴马的妻子。You look like Obama's wife. **Seq2Seq:** 哈哈哈哈哈哈。Haha… **GAN-AEL:** 我是大众脸。I have a common face.
Query: Turn his face away from me. **Seq2Seq:** I don't know what to do. **GAN-AEL:** Let me explain to you about the mistake.	**Query:** 我不光搭讪，还请吃饭呢。 I wish to invite you to dinner rather than chat up only. **Seq2Seq:** 我也想吃。I want to have dinner too. **GAN-AEL:** 我是胖子，减肥。I am fat, and I am on a diet.

Figure 3: Example responses generated by Seq2Seq and GAN-AEL.

Conclusion

- An approximation embedding layer is introduced to force the entire network differentiable, which signicantly overcomes the drawbacks found in the previous RL-based attempts.
- The superiority of the proposed method has been demonstrated by empirical experiments based on both automatic evaluation metrics and human judgements.

- Commonsense Knowledge Aware Conversation Generation with Graph Attention

Introduction

- Some models are highly dependent on the quality of unstructured texts or limited by the small-scale, domain-specific knowledge.
- They usually make use of knowledge triples (entities) separately and independently, instead of treating knowledge triples as a whole in a graph.
- To address the two issues, this paper propose a commonsense knowledge aware conversational model (CCM) to facilitate language understanding and generation in open-domain conversational. systems.

Task Definition and Overview

- Given a post $X = \{x_1, x_2, ..., x_n\}$ and some commonsense knowledge graphs $G = \{g_1, g_2, ..., g_{H_G}\}$, the goal is to generate a proper response $Y = \{y_1, y_2, ...y_m\}$.
- The graphs are retrieved from a knowledge base using the words in a post as queries, and each word coresponds to a graph in G.
- Each graph consists of a set of triple $g_i = \{\tau_1, \tau_2, ..., \tau_{N_{g_i}}\}$ and each triple (head entity, relation, tail entity) is denoted as $\tau = (h, r, t)$
- A knowledge triple τ is represented by $\mathbf{k} = (\mathbf{h}, \mathbf{r}, \mathbf{t}) = \mathbf{MLP}(\mathit{TransE}(h, r, t))$.

Model Overview

- The knowledge interpreter takes as input a post $X = x_1 x_2 ... x_n$ and retrieved knowledge graphs $G = \{g_1, g_2, ..., g_n\}$ to obtain knowledge aware representations at each word position, by concatenating a word vector and its corresponding knowledge graph vector.
- A knowledge graph vector represents a knowledge graph for the corresponding word in X through a static graph attention mechanism.
- The knowledge aware generator generates a response $Y = y_1 y_2 ... y_m$ with our dynamic graph attention mechanism. At each decoding position, it attentively reads the retrieved graphs and the entities in each graph, and then generates a generic word in the vocabulary or an entity in the knowledge graphs.

Model Overview

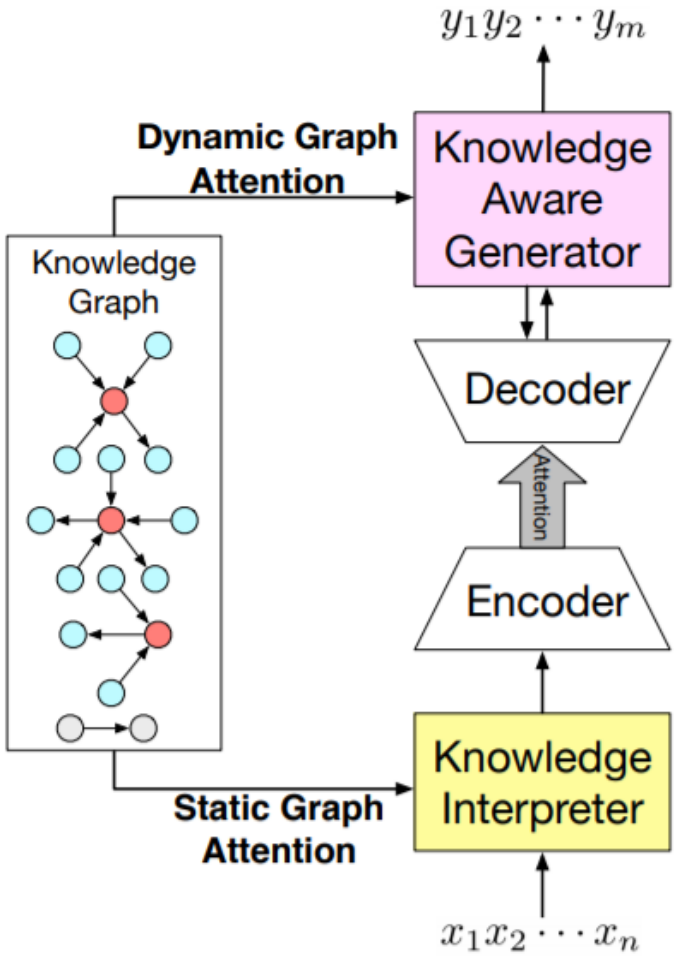


Figure 2: Overview of CCM.

Knowledge Interpreter

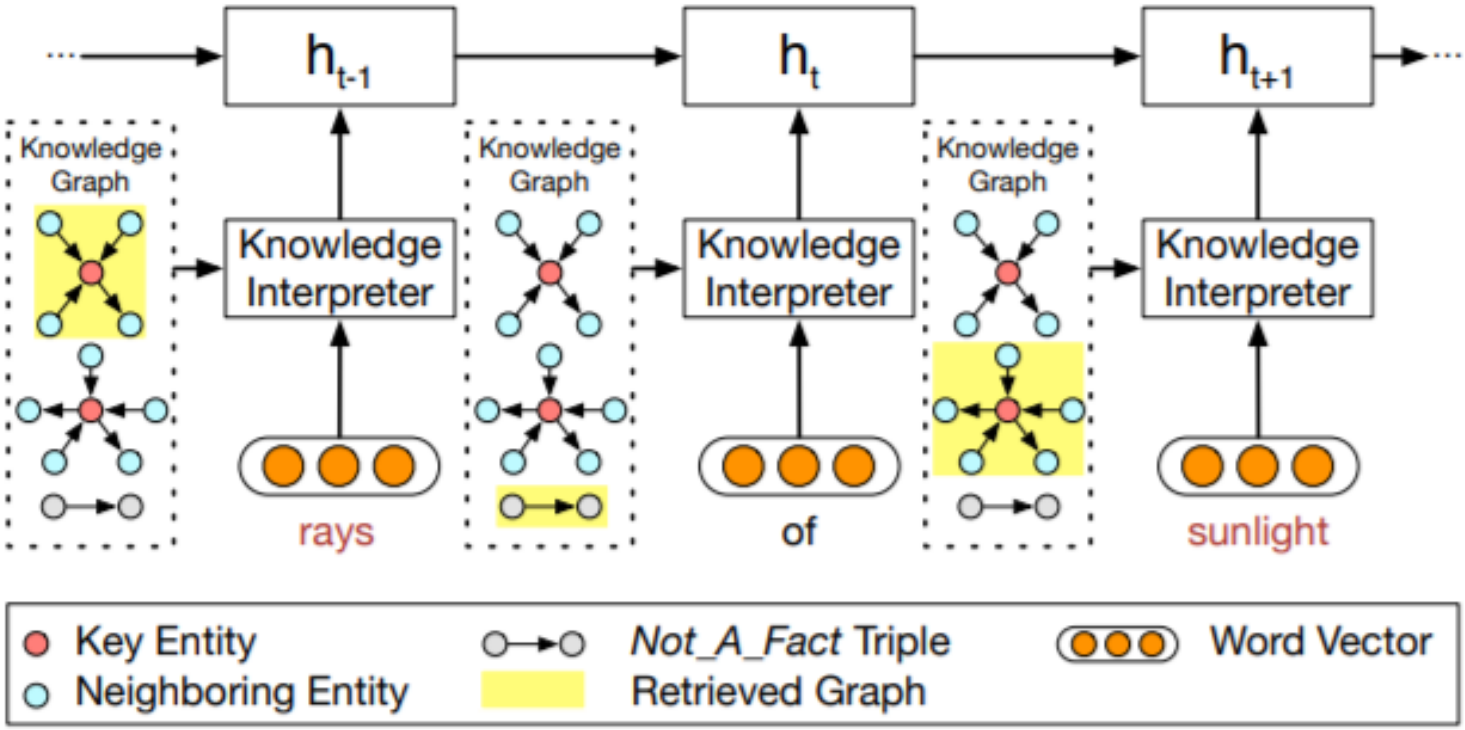


Figure 3: Knowledge interpreter concatenates a word vector and the graph vector of the corresponding retrieved graph. In this example, word *rays* (also key entity) corresponds to the first graph, and *sunlight* to the second one. Each graph is represented by a graph vector. A key entity is an entity which occurs in the post.

Static Graph Attention

- The static graph attention mechanism takes as input the knowledge triple vectors $\mathbf{K}(g_i) = \{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_{Ng_i}\}$ in retrieved knowledge graph g_i, to produce a graph vector $\mathbf{g_i}$ as follows:
-

$$\mathbf{g_i} = \sum_{n=1}^{N_{g_i}} \alpha_n^s [\mathbf{h_n}; \mathbf{t_n}]$$

$$\alpha_n^s = \frac{exp\beta_n^s}{\sum_{j=1}^{N_{g_i}} exp(\beta_j^s)}$$

$$\beta_n^s = (\mathbf{W_r r_n})^T tanh(\mathbf{W_h h}_n + \mathbf{W_t t_n})$$

- where $(\mathbf{h_n}, \mathbf{r_n}, \mathbf{t_n}) = \mathbf{k_n}, \mathbf{W_h}, \mathbf{W_r}, \mathbf{W_t}$ are weight matrices for head entities, relations, and tail entities, respectively.
- Essentially, a graph vector $\mathbf{g}_i$ is a weighted sum of the head and tail vector $[\mathbf{h}_n; \mathbf{t}_n]$ of the triples contained in the graph.

Knowledge Aware Generator

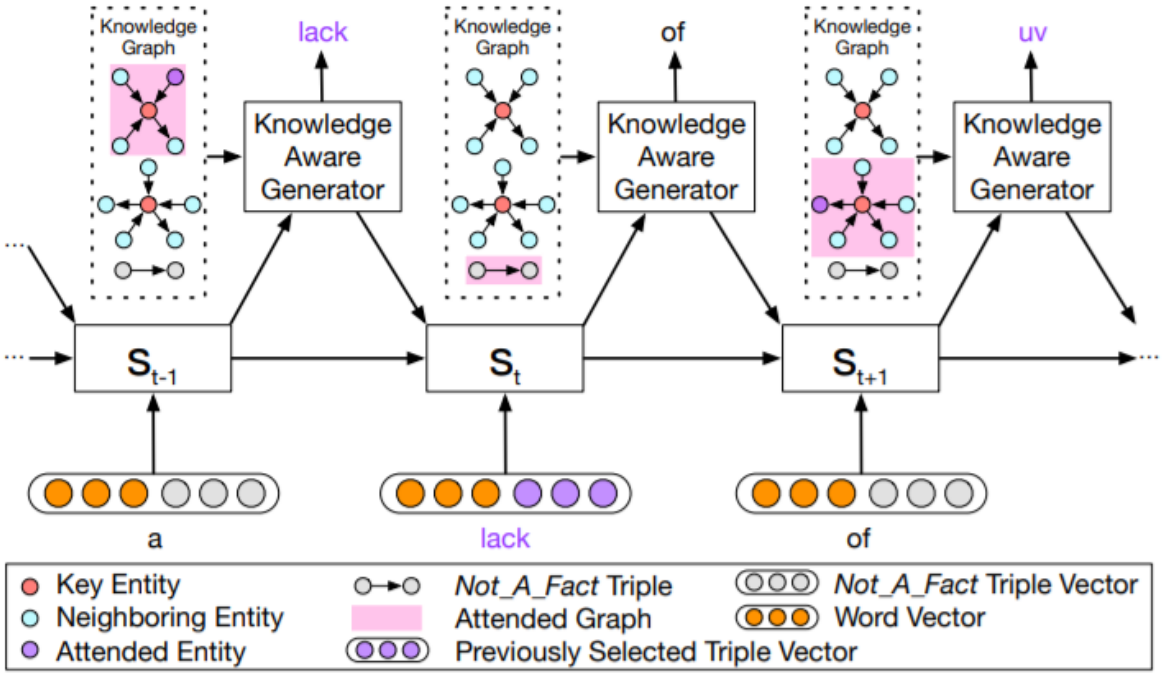


Figure 4: Knowledge aware generator dynamically attends on the graphs (the pink graph is mostly attended) . It then attentively reads the triples in each graph to estimate the probability of selecting a triple, where the triple's neighboring entity (purple dots/words) is used for word generation.

Dynamic Graph Attention

- Given the decoder state $\mathbf{s}_t$, the dynamic graph attention mechanism first attends on the knowledge graph vectors $\{\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_{N_G}\}$ to compute the probability of using of each graph g_i, which is defined as below:
-

$$\mathbf{c}_t^g = \sum_{i=1}^{N_G} \alpha_{ti}^g \mathbf{g}_i$$

$$\alpha_{ti}^g = \frac{exp(\beta_{ti}^g)}{\sum_{j=1}^{N_G} exp(\beta_{ti}^g)}$$

$$\beta_{ti}^g = \mathbf{V}_b^T tanh(\mathbf{W_b s}_t + \mathbf{U_b g}_i)$$

- where $\mathbf{V}_b/\mathbf{W}_b/\mathbf{U}_b$ are parameters, and α_{ti}^g is the probability of choosing knowledge graph g_i at step t. The graph context vector c_t^g is a weighted sum of the graph vectors, and the weight measures the association between the decoder's state $\mathbf{s}_t$ and a graph vector $\mathbf{g}_i$.

Dynamic Graph Attention

- The model then attends on the knowledge triple vectors $\mathbf{K}(g_i) = \{\mathbf{k}_1, \mathbf{k}_2, ...\mathbf{k}_{N_{gi}}\}$ within each graph g_i to calculate the probability of selecting a triple for word generation, formally as follows:
-

$$\mathbf{c}_t^k = \sum_{i=1}^{N_G} \sum_{j=1}^{N_{gi}} \alpha_{ti}^g \alpha_{tj}^k \mathbf{k}_i$$

$$\alpha_{tj}^k = \frac{exp(\beta_{ti}^k)}{\sum_{n=1}^{N_{gi}} exp(\beta_{tn}^k)}$$

$$\beta_{tj}^k = \mathbf{k}_j^T \mathbf{W_c} \mathbf{s}_t$$

- where β_{tj}^k can be viewed as the similarity between each knowledge triple vector $\mathbf{k}_j$ and the decoder state $\mathbf{s}_t$, α_{tj}^k is the probability pf choosing triple τ_j from all triples in graph g_i at step t.

Knowledge Aware Generator

- Finally, the knowledge aware generator selects a generic word or an entity word with the following distributions:
-

$$\mathbf{a}_t = [\mathbf{s}_t; \mathbf{c}_t; \mathbf{c}_t^g; \mathbf{c}_t^k]$$

$$\gamma_t = sigmoid(\mathbf{V}_o^T \alpha_t)$$

$$P_c(y_t = w_c) = softmax(\mathbf{W_o} \mathbf{a}_t)$$

$$P_e(y_t = w_e) = \alpha_{ti}^g \alpha_{tj}^k$$

$$y_t \sim \mathbf{o}_t = P(y_t) = [(1 - \gamma_t) P_g(y_t = w_c); \gamma_t P_e(y_t = w_e)]$$

- where $\gamma_t \in [0, 1]$ is a scalar to balance the choice between an entity word w_e and a generic word w_c, P_c/P_e is the distribution over generic/entity words respectively.The final distribution $P(y_t)$ is a concatenation of two disributions.

Loss Function

- The loss function is cross entropy between the predicted token distribution $\mathbf{o}_t$ and the reference distribution $\mathbf{p}_t$ in the training corpus.The loss on one sample $< X, Y >$ $(X = x_1 x_2 ... x_n, Y = y_1 y_2 ... y_m)$ is defined as:
-

$$L(\theta) = -\sum_{t=1}^{m} \mathbf{p}_t log(\mathbf{o}_t) - \sum_{t=1}^{m} (q_t log(\gamma_t) + (1 - q_t) log(1 - \gamma_t))$$

- where γ_t is the probability of selecting an entity word or a generic word, and $q_t \in \{0, 1\}$ is the true choice of an entity word or a generic word in $\mathbf{Y}$. The second term is used to supervise the probability of selecting an entity word or a generic word.

Manual evaluation

Model	Overall		High Freq.		Medium Freq.		Low Freq.		OOV	
	ppx.	ent.	ppx.	ent.	ppx.	ent.	ppx.	ent.	ppx.	ent.
Seq2Seq	47.02	0.717	42.41	0.713	47.25	0.740	48.61	0.721	49.96	0.669
MemNet	46.85	0.761	41.93	0.764	47.32	0.788	48.86	0.760	49.52	0.706
CopyNet	40.27	0.96	36.26	0.91	40.99	0.97	42.09	0.96	42.24	0.96
CCM	**39.18**	**1.180**	**35.36**	**1.156**	**39.64**	**1.191**	**40.67**	**1.196**	**40.87**	**1.162**

Table 2: Automatic evaluation with *perplexity* (ppx.), and *entity score* (ent.).

Model	Overall		High Freq.		Medium Freq.		Low Freq.		OOV	
	app.	inf.	app.	inf.	app.	inf.	app.	inf.	app.	inf.
CCM vs. Seq2Seq	0.616	0.662	0.605	0.656	0.549	0.624	0.636	0.650	0.673	0.716
CCM vs. MemNet	0.602	0.647	0.593	0.656	0.566	0.640	0.622	0.635	0.626	0.657
CCM vs. CopyNet	0.600	0.640	0.606	0.669	0.586	0.619	0.610	0.633	0.596	0.640

Table 3: Manual evaluation with *appropriateness* (app.), and *informativeness* (inf.). The score is the percentage that CCM wins its competitor after removing "Tie" pairs. CCM is significantly better (sign test, p-value < 0.005) than all the baselines on all the test sets.

Sample responses

Post	Why are you so **breakable**?
Knowledge	(**glass**, RelatedTo, **breakable**), (**brittle**, RelatedTo, **breakable**), (**rule**, RelatedTo, **breakable**)
Seq2Seq	I'm not a **OOV** , I'm just a **OOV**.
MemNet	I'm not **OOV**. I'm just a really nice person.
CopyNet	I'm not. I'm just a lurker.
CCM	Because I'm a **brittle** man .

Table 4: Sample responses generated by all the models.

Conclusion

- This paper presents a commonsense knowledge aware conversational model (CCM) to demonstrate how commonsense knowledge can facilitate language understanding and generation in open-domain conversational systems.
- Instead of treating knowledge triples (or entities) separately and independently, we devise static and dynamic graph attention mechanisms to treat the knowledge triples as a graph, from which we can better interpret the semantics of an entity from its neighboring entities and relations.