Barack's Wife Hillary: Using Knowledge Graphs for Fact-Aware Language Modeling

Robert L. Logan IV* **Nelson F. Liu†§** **Matthew E. Peters§**
Matt Gardner§ **Sameer Singh***

* University of California, Irvine, CA, USA
† University of Washington, Seattle, WA, USA
§ Allen Institute for Artificial Intelligence, Seattle, WA, USA
{rlogan, sameer}@uci.edu, {mattg, matthewp}@allenai.org, nfliu@cs.washington.edu

Abstract

Modeling human language requires the ability to not only generate fluent text but also encode factual knowledge. However, traditional language models are only capable of remembering facts seen at training time, and often have difficulty recalling them. To address this, we introduce the knowledge graph language model (KGLM), a neural language model with mechanisms for selecting and copying facts from a knowledge graph that are relevant to the context. These mechanisms enable the model to render information it has never seen before, as well as generate out-of-vocabulary tokens. We also introduce the *Linked WikiText-2* dataset,[1] a corpus of annotated text aligned to the Wikidata knowledge graph whose contents (roughly) match the popular *WikiText-2* benchmark (Merity et al., 2017). In experiments, we demonstrate that the KGLM achieves significantly better performance than a strong baseline language model. We additionally compare different language models' ability to complete sentences requiring factual knowledge, and show that the KGLM outperforms even very large language models in generating facts.

1 Introduction

For language models to generate plausible sentences, they must be both syntactically coherent as well as consistent with the world they describe. Although language models are quite skilled at generating grammatical sentences, and previous work has shown that language models also possess some degree of common-sense reasoning and basic knowledge (Vinyals and Le, 2015; Serban et al., 2016; Trinh and Le, 2019), their ability to generate *factually correct* text is quite limited. The clearest limitation of existing language models is that they, at best, can only memorize facts observed during training. For instance, when conditioned on the text at the top of Figure 1, an AWD-LSTM language model (Merity et al., 2018) trained on *Wikitext-2* assigns higher probability to the word "*PlayStation*" than "*Game Boy*", even though this sentence appears verbatim in the training data. This is not surprising—existing models represent the distribution over the entire vocabulary directly, whether they are common words, references to real world entities, or factual information like dates and numbers. As a result, language models are unable to generate factually correct sentences, do not generalize to rare/unseen entities, and often omit rare tokens from the vocabulary (instead generating *UNKNOWN* tokens).

[*Super Mario Land*] is a [*1989*] [*side-scrolling*] [*platform video game*] developed and published by [*Nintendo*] as a [*launch title*] for their [*Game Boy*] [*handheld game console*].

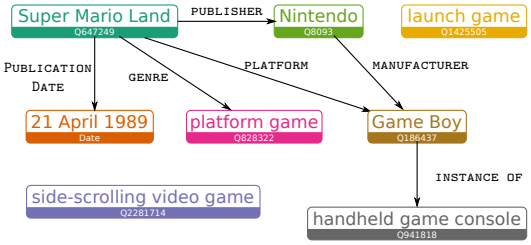

Figure 1: ***Linked WikiText-2* Example**. A localized knowledge graph containing facts that are (possibly) conveyed in the sentence above. The graph is built by iteratively linking each detected entity to Wikidata, then adding any relations to previously mentioned entities. Note that not all entities are connected, potentially due to missing relations in Wikidata.

We introduce the *knowledge graph language model* (KGLM), a neural language model with mechanisms for selecting and copying information from an external knowledge graph. The KGLM maintains a dynamically growing *local knowledge*

[1] https://rloganiv.github.io/linked-wikitext-2

graph, a subset of the knowledge graph that contains entities that have already been mentioned in the text, and their related entities. When generating entity tokens, the model either decides to render a new entity that is absent from the local graph, thereby growing the local knowledge graph, or to render a fact from the local graph. When rendering, the model combines the standard vocabulary with tokens available in the knowledge graph, thus supporting numbers, dates, and other rare tokens.

Figure 1 illustrates how the KGLM works. Initially, the graph is empty and the model uses the entity Super Mario Land to render the first three tokens, thus adding it and its relations to the local knowledge graph. After generating the next two tokens ("*is*", "*a*") using the standard language model, the model selects Super Mario Land as the parent entity, *Publication Date* as the relation to render, and copies one of the tokens of the date entity as the token ("*1989*" in this case).

To facilitate research on knowledge graph-based language modeling, we collect the distantly supervised *Linked WikiText-2* dataset. The underlying text closely matches *WikiText-2* (Merity et al., 2017), a popular benchmark for language modeling, allowing comparisons against existing models. The tokens in the text are linked to entities in Wikidata (Vrandečić and Krötzsch, 2014) using a combination of human-provided links and off-the-shelf linking and coreference models. We also use relations between these entities in Wikidata to construct plausible reasons for why an entity may have been mentioned: it could either be related to an entity that is already mentioned (including itself) or a brand new, unrelated entity for the document.

We train and evaluate the KGLM on *Linked WikiText-2*. When compared against AWD-LSTM, a recent and performant language model, KGLM obtains not only a lower overall perplexity, but also a substantially lower *unknown-penalized* perplexity (Ueberla, 1994; Ahn et al., 2016), a metric that allows fair comparisons between models that accurately model rare tokens and ones that predict them to be *unknown*. We also compare *factual completion* capabilities of these models, where they predict the next word after a factual sentence (e.g., "*Barack is married to* ___") and show that KGLM is significantly more accurate. Lastly, we show that the model is able to generate accurate facts for rare entities, and can be *controlled* via modifications the knowledge graph.

2 Knowledge Graph Language Model

In this section we introduce a language model that is conditioned on an external, structured knowledge source, which it uses to generate factual text.

2.1 Problem Setup and Notation

A *language model* defines a probability distribution over each token within a sequence, conditioned on the sequence of tokens observed so far. We denote the random variable representing the next token as x_t and the sequence of the tokens before t as $x_{<t}$, i.e. language models compute $p(x_t|x_{<t})$. RNN language models (Mikolov et al., 2010) parameterize this distribution using a recurrent structure:

$$p(x_t|x_{<t}) = \text{softmax}(\mathbf{W}_h\mathbf{h}_t + \mathbf{b}),$$
$$\mathbf{h}_t = \text{RNN}(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}). \quad (1)$$

We use LSTMs (Hochreiter and Schmidhuber, 1997) as the recurrent module in this paper.

A *knowledge graph* (KG) is a directed, labeled graph consisting of entities $\mathcal{E}$ as nodes, with edges defined over a set of relations $\mathcal{R}$, i.e. $\mathcal{KG} = \{(p, r, e) \mid p \in \mathcal{E}, r \in \mathcal{R}, e \in \mathcal{E}\}$, where p is a parent entity with relation r to another entity e. Practical KGs have other aspects that make this formulation somewhat inexact: some relations are to *literal values*, such as numbers and dates, facts may be expressed as *properties* on relations, and entities have *aliases* as the set of strings that can refer to the entity. We also define a *local knowledge graph* for a subset of entities $\mathcal{E}_{<t}$ as $\mathcal{KG}_{<t} = \{(p, r, e) \mid p \in \mathcal{E}_{<t}, r \in \mathcal{R}, e \in \mathcal{E}\}$, i.e. contains entities $\mathcal{E}_{<t}$ and all facts they participate in.

2.2 Generative KG Language Model

The primary goal of the knowledge graph language model (KGLM) is to enable a neural language model to generate entities and facts from a knowledge graph. To encourage the model to generate facts that have appeared in the context already, KGLM will maintain a local knowledge graph containing all facts involving entities that have appeared in the context. As the model decides to refer to entities that have not been referred to yet, it will grow the local knowledge graph with additional entities and facts to reflect the new entity.

Formally, we will compute $p(x_t, \mathcal{E}_t|x_{<t}, \mathcal{E}_{<t})$ where $x_{<t}$ is the sequence of observed tokens, $\mathcal{E}_{<t}$ is the set of entities mentioned in $x_{<t}$, and $\mathcal{KG}_{<t}$ is the local knowledge graph determined by $\mathcal{E}_{<t}$, as described above. The generative process is:

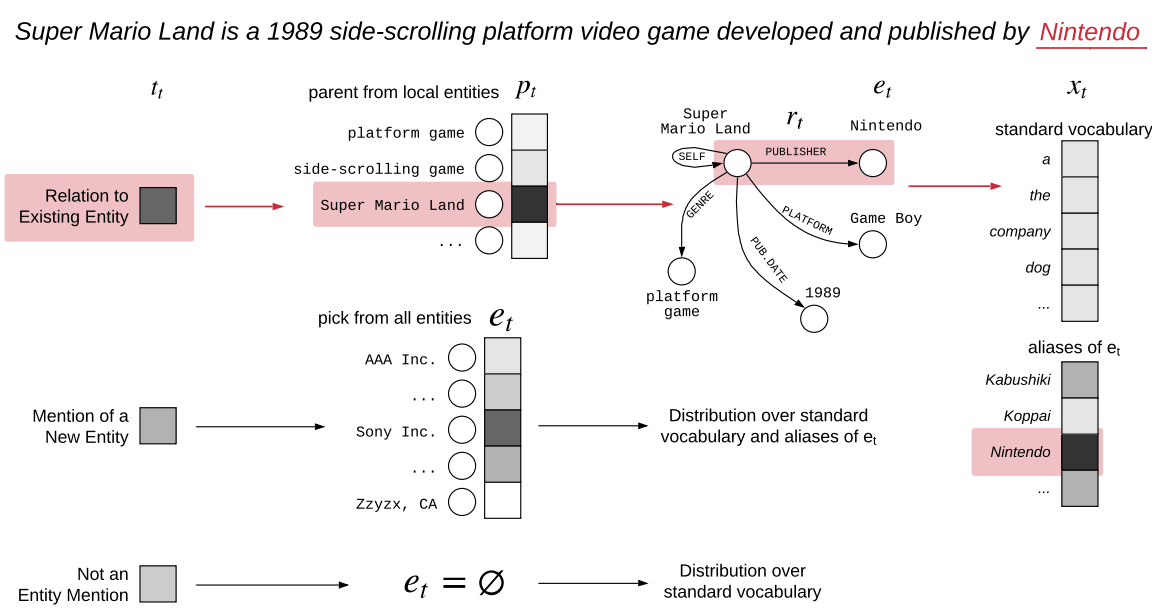

Figure 2: **KGLM Illustration.** When trying to generate the token following "*published by*", the model first decides the type of the mention (t_t) to be a related entity (darker indicates higher probability), followed by identifying the parent (p_t), relation (r_t), and entity to render (e_t) from the local knowledge graph as (Super Mario Land, *Publisher*, Nintendo). The final distribution over the words includes the standard vocabulary along with aliases of Nintendo, and the model selects "*Nintendo*" as the token x_t. Facts related to Nintendo will be added to the local graph.

- Decide the *type* of x_t, which we denote by t_t: whether it is a reference to an entity in $\mathcal{KG}_{<t}$ (related), a reference to an entity not in $\mathcal{KG}_{<t}$ (new), or not an entity mention ($\emptyset$).
- If $t_t = \text{new}$ then choose the upcoming entity e_t from the set of all entities $\mathcal{E}$.
- If $t_t = \text{related}$ then:
  - Choose a parent entity p_t from $\mathcal{E}_{<t}$.
  - Choose a factual relation r_t to render, $r_t \in \{(p, r, e) \in \mathcal{KG}_{<t} | p = p_t\}$.
  - Choose e_t as one of the tail entities, $e_t \in \{e | (p_t, r_t, e) \in \mathcal{KG}_{<t}\}$.
- If $t_t = \emptyset$ then $e_t = \emptyset$.
- Generate x_t conditioned on e_t, potentially copying one of e_t's aliases.
- If $e_t \notin \mathcal{E}_{<t}$, then $\mathcal{E}_{<(t+1)} \leftarrow \mathcal{E}_{<t} \cup \{e_t\}$, else $\mathcal{E}_{<(t+1)} \leftarrow \mathcal{E}_{<t}$.

For the model to refer to an entity it has already mentioned, we introduce a *Reflexive* relation that self-relates, i.e. $p = e$ for $(p, \textit{Reflexive}, e)$.

An illustration of this process and the variables is provided in Figure 2, for generating a token in the middle of the same sentence as in Figure 1. Amongst the three mention types (t_t), the model chooses a reference to existing entity, which requires picking a fact to render. As the parent entity of this fact (p_t), the model picks Super Mario Land, and then follows the *Publisher* relation (r_t) to select Nintendo as the entity to render (e_t). When rendering Nintendo as a token x_t, the model has an *expanded* vocabulary available to it, containing the standard vocabulary along with all word types in any of the aliases of e_t.

Marginalizing out the KG There is a mismatch between our initial task requirement, $p(x_t|x_{<t})$, and the model we describe so far, which computes $p(x_t, \mathcal{E}_t|x_{<t}, \mathcal{E}_{<t})$. We will essentially *marginalize* out the local knowledge graph to compute the probability of the tokens, i.e. $p(\mathbf{x}) = \sum_{\boldsymbol{\mathcal{E}}} p(\mathbf{x}, \boldsymbol{\mathcal{E}})$. We will clarify this, along with describing the training and the inference/decoding algorithms for this model and other details of the setup, in Section 4.

2.3 Parameterizing the Distributions

The parametric distributions used in the generative process above are defined as follows. We begin by computing the hidden state $\mathbf{h}_t$ using the formula in Eqn (1). We then split the vector into three components: $\mathbf{h}_t = [\mathbf{h}_{t,x}; \mathbf{h}_{t,p}; \mathbf{h}_{t,r}]$, which are respectively used to predict words, parents, and relations. The type of the token, t_t, is computed using a single-layer softmax over $\mathbf{h}_{t,x}$ to predict one of $\{\text{new}, \text{related}, \emptyset\}$.

Picking an Entity We also introduce pretrained embeddings for all entities and relations in the

knowledge graph, denoted by $\mathbf{v}_e$ for entity e and $\mathbf{v}_r$ for relation r. To select e_t from all entities in case $t_t = \text{new}$, we use:

$$p(e_t) = \text{softmax}(\mathbf{v}_e \cdot (\mathbf{h}_{t,p} + \mathbf{h}_{t,r}))$$

over all $e \in \mathcal{E}$. The reason we add $\mathbf{h}_{t,p}$ and $\mathbf{h}_{t,r}$ is to mimic the structure of TransE, which we use to obtain entity and relation embeddings. Details on TransE will be provided in Section 4. For mention of a related entity, $t_t = \text{related}$, we pick a parent entity p_t using

$$p(p_t) = \text{softmax}(\mathbf{v}_p \cdot \mathbf{h}_{t,p})$$

over all $p \in \mathcal{E}_t$, then pick the relation r_t using

$$p(r_t) = \text{softmax}(\mathbf{v}_r \cdot \mathbf{h}_{t,r})$$

over all $r \in \{r|(p_t, r, e) \in \mathcal{KG}_t\}$. The combination of p_t and r_t determine the entity e_t (which must satisfy $(p_t, r_t, e_t) \in \mathcal{KG}_t$; if there are multiple options one is chosen at random).

Rendering the Entity If $e_t = \emptyset$, i.e. there is no entity to render, we use the same distribution over the vocabulary as in Eqn (1) - a softmax using $\mathbf{h}_{t,x}$. If there is an entity to render, we construct the distribution over the original vocabulary *and* a vocabulary containing all the tokens that appear in aliases of e_t. This distribution is conditioned on e_t in addition to x_t. To compute the scores over the original vocabulary, $\mathbf{h}_{t,x}$ is replaced by $\mathbf{h}'_{t,x} = \mathbf{W}_{\text{proj}}[\mathbf{h}_{t,x}; \mathbf{v}_{e_t}]$ where $\mathbf{W}_{\text{proj}}$ is a learned weight matrix that projects the concatenated vector into the same vector space as $\mathbf{h}_{t,x}$.

To obtain probabilities for words in the alias vocabulary, we use a copy mechanism Gu et al. (2016). The token sequences comprising each alias $\{a_j\}$ are embedded then encoded using an LSTM to form vectors $\mathbf{a}_j$. Copy scores are computed as:

$$p(x_t = a_j) \propto \exp\left[\sigma\left(\left(\mathbf{h}'_{t,x}\right)^T \mathbf{W}_{\text{copy}}\right)\mathbf{a}_j\right]$$

3 *Linked WikiText-2*

Modeling aside, one of the primary barriers to incorporating factual knowledge into language models is that training data is hard to obtain. Standard language modeling corpora consist only of text, and thus are unable to describe which entities or facts each token is referring to. In contrast, while relation extraction datasets link text to a knowledge graph, the text is made up of disjoint sentences that do not provide sufficient context to train a powerful language model. Our goals are much more aligned to the *data-to-text* task (Ahn et al., 2016; Lebret et al., 2016; Wiseman et al., 2017; Yang et al., 2017; Gardent et al., 2017; Ferreira et al., 2018), where a small table-sized KB is provided to generate a short piece of text; we are interested in language models that dynamically decide the facts to incorporate from the knowledge graph, guided by the discourse.

For these reasons we introduce the *Linked WikiText-2* dataset, consisting of (approximately) the same articles appearing in the *WikiText-2* language modeling corpus, but linked to the Wikidata (Vrandečić and Krötzsch, 2014) knowledge graph. Because the text closely matches, models trained on *Linked WikiText-2* can be compared to models trained on *WikiText-2*. Furthermore, because many of the facts in Wikidata are derived from Wikipedia articles, the knowledge graph has a good coverage of facts expressed in the text. The dataset is available for download at: https://rloganiv.github.io/linked-wikitext-2. Our system annotates one document at a time, and consists of entity linking, relation annotations, and post-processing. The following paragraphs describe each step in detail.

Initial entity annotations We begin by identifying an initial set of entity mentions within the text. The primary source of these mentions is the human-provided links between Wikipedia articles. Whenever a span of text is linked to another Wikipedia article, we associate its corresponding Wikidata entity with the span. While article links provide a large number of gold entity annotations, they are insufficient for capturing all of the mentions in the article since entities are only linked the first time they occur. Accordingly, we use the neural-el (Gupta et al., 2017) entity linker to identify additional links to Wikidata, and identify coreferences using Stanford CoreNLP[2] to cover pronouns, nominals, and other tokens missed by the linker.

Local knowledge graph The next step iteratively creates a generative story for the entities using relations in the knowledge graph as well as identifies new entities. To do this, we process the text token by token. Each time an entity is encountered, we add all of the related entities in Wikidata as candi-

[2] https://stanfordnlp.github.io/CoreNLP/

Tokens x_t	Super Mario Land	is	a	1989	side - scrolling	platform video game	developed
Mention type t_t	new	∅	∅	related	new	related	∅
Entity Mentioned e_t	SML	∅	∅	04-21-1989	SIDE_SCROLL	PVG	∅
Relation r_t	∅	∅	∅	pub date	∅	genre	∅
Parent Entity p_t	∅	∅	∅	SML	∅	SML	∅

x_t	and	published	by	Nintendo	as	a	launch title	for	their	Game Boy	handheld game console	.
t_t	∅	∅	∅	related	∅	∅	new	∅	∅	related	related	∅
e_t	∅	∅	∅	NIN	∅	∅	LT	∅	∅	GAME_BOY	HGC	∅
r_t	∅	∅	∅	pub	∅	∅	∅	∅	∅	R:manu / platform	instance of	∅
p_t	∅	∅	∅	SML	∅	∅	∅	∅	∅	NIN / SML	GAME_BOY	∅

Table 1: **Example Annotation** of the sentence from Figure 1, including corresponding variables from Figure 2. Note that *Game Boy* has multiple parent and relation annotations, as the platform for Super Mario Land and as manufactured by Nintendo. Wikidata identifiers are made human-readable (e.g., SML is Q647249) for clarity.

dates for matching. If one of these related entities is seen later in the document, we identify the entity as a parent for the later entity. Since multiple relations may appear as *explanations* for each token, we allow a token to have multiple facts.

Expanding the annotations Since there may be entities that were missed in the initial set, as well as non-entity tokens of interest such as dates and quantities we further expand the entity annotations using string matching. For entities, we match the set of aliases provided in Wikidata. For dates, we create an exhaustive list of all of the possible ways of expressing the date (e.g. "*December 7, 1941*", "*7-12-1941*", "*1941*", ...). We perform a similar approach for quantities, using the pint library in Python to handle the different ways of expressing units (e.g. "*g*", "*gram*", ...). Since there are many ways to express a numerical quantity, we only render the quantity at the level of precision supplied by Wikidata, and do not perform unit conversions.

Example Annotation An example annotation is provided in Table 1 corresponding to the instance in Figure 1, along with the variables that correspond to the generative process of the knowledge graph language model (KGLM). The entity mentioned for most tokens here are human-provided links, apart from "*1989*" that is linked to 04-21-1989 by the string matching process. The annotations indicate which of the entities are *new* and *related* based on whether they are reachable by entities linked so far, clearly making a mistake for side-scrolling game and platform video game due to missing links in Wikidata. Finally, multiple plausible reasons for Game Boy are included: it's the platform for Super Mario Land and it is manufactured by Nintendo, even though only the former is more relevant here.

	Train	**Dev**	**Test**
Documents	600	60	60
Tokens	2,019,195	207,982	236,062
Vocab. Size	33,558	-	-
Mention Tokens	207,803	21,226	24,441
Mention Spans	122,983	12,214	15,007
Unique Entities	41,058	5,415	5,625
Unique Relations	1,291	484	504

Table 2: ***Linked WikiText-2* Corpus Statistics.**

Even with these omissions and mistakes, it is clear that the annotations are rich and detailed, with a high coverage, and thus should prove beneficial for training knowledge graph language models.

Dataset Statistics Statistics for *Linked WikiText-2* are provided in Table 2. In this corpus, more than 10% of the tokens are considered entity tokens, i.e. they are generated as factual references to information in the knowledge graph. Each entity is only mentioned a few times (less than 5 on average, with a long tail), and with more than thousand different relations. Thus it is clear that regular language models would not be able to generate factual text, and there is a need for language models to be able to refer to external sources of information.

Differences from *WikiText-2* Although our dataset is designed to closely replicate *WikiText-2*, there are some differences that prevent direct comparison. Firstly, there are minor variations in text across articles due to edits between download dates. Secondly, according to correspondence with Merity et al. (2017), *WikiText-2* was collected by querying the Wikipedia Text API. Because this API discards useful annotation information (e.g. article links), *Linked WikiText-2* instead was created by directly from the article HTML.

4 Training and Inference for KGLM

In this section, we describe the training and inference algorithm for KGLM.

Pretrained KG Embeddings During evaluation, we may need to make predictions on entities and relations that have not been seen during training. Accordingly, we use fixed entity and relations embeddings pre-trained using TransE (Bordes et al., 2013) on Wikidata. Given (p, r, e), we learn embeddings $\mathbf{v}_p$, $\mathbf{v}_r$ and $\mathbf{v}_e$ to minimize the distance:

$$\delta(\mathbf{v}_p, \mathbf{v}_r, \mathbf{v}_e) = \|\mathbf{v}_p + \mathbf{v}_r - \mathbf{v}_e\|^2 .$$

We use a max-margin loss to learn the embeddings:

$$\mathcal{L} = \max\left(0, \gamma + \delta\left(\mathbf{v}_p, \mathbf{v}_r, \mathbf{v}_e\right) - \delta\left(\mathbf{v}'_p, \mathbf{v}_r, \mathbf{v}'_e\right)\right)$$

where γ is the margin, and either p' or e' is a randomly chosen entity embedding.

Training with *Linked WikiText-2* Although the generative process in KGLM involves many steps, training the model on *Linked WikiText-2* is straightforward. Our loss objective is the negative log-likelihood of the training data:

$$\ell(\Theta) = \sum_t \log p(x_t, \mathcal{E}_t | x_{<t}, \mathcal{E}_{<t}; \Theta),$$

where Θ is the set of model parameters. Note that if an annotation has multiple viable parents such as Game Boy in 1, then we marginalize over all of the parents. Since all random variables are observed, training can performed using off-the-shelf gradient-based optimizers.

Inference While observing annotations makes the model easy to train, we do not assume that the model has access to annotations during evaluation. Furthermore, as discussed in Section 2.2, the goal in language modelling is to measure the marginal probability $p(\mathbf{x}) = \sum_{\boldsymbol{\mathcal{E}}} p(\mathbf{x}, \boldsymbol{\mathcal{E}})$ not the joint probability. However, this sum is intractable to compute due to the large combinatorial space of possible annotations. We address this problem by approximating the marginal distribution using importance sampling. Given samples from a proposal distribution $q(\boldsymbol{\mathcal{E}}|\mathbf{x})$ the marginal distribution is:

$$\begin{aligned} p(\mathbf{x}) &= \sum_{\boldsymbol{\mathcal{E}}} p\left(\mathbf{x}, \boldsymbol{\mathcal{E}}\right) = \sum_{\boldsymbol{\mathcal{E}}} \frac{p\left(\mathbf{x}, \boldsymbol{\mathcal{E}}\right)}{q\left(\boldsymbol{\mathcal{E}}|\mathbf{x}\right)} q\left(\boldsymbol{\mathcal{E}}|\mathbf{x}\right) \\ &\approx \frac{1}{N} \sum_{\boldsymbol{\mathcal{E}} \sim q} \frac{p\left(\mathbf{x}, \boldsymbol{\mathcal{E}}\right)}{q\left(\boldsymbol{\mathcal{E}}|\mathbf{x}\right)} \end{aligned}$$

This approach is used to evaluate models in Ji et al. (2017) and Dyer et al. (2016). Following Ji et al. (2017), we compute $q\left(\boldsymbol{\mathcal{E}}|\mathbf{x}\right)$ using a discriminative version of our model that predicts annotations for the current token instead of for the next token.

5 Experiments

To evaluate the proposed language model, we first introduce the baselines, followed by an evaluation using perplexity of held-out corpus, accuracy on fact completion, and an illustration of how the model uses the knowledge graph.

5.1 Evaluation Setup

Baseline Models We compare KGLM to the following baseline models:

- **AWD-LSTM** (Merity et al., 2018): strong LSTM-based model used as the foundation of most state-of-the-art models on *WikiText-2*.
- **EntityNLM** (Ji et al., 2017): an LSTM-based language model with the ability to track entity mentions. Embeddings for entities are created dynamically, and are not informed by any external sources of information.
- **EntityCopyNet:** a variant of the KGLM where t_t = new for all mentions, i.e. entities are selected from $\mathcal{E}$ and entity aliases are copied, but relations in the knowledge graph are unused.

Hyperparameters We pre-train 256 dimensional entity and relation embeddings for all entities within two hops of the set of entities that occur in *Linked WikiText-2* using TransE with margin $\gamma = 1$. Weights are tied between all date embeddings and between all quantity embeddings to save memory. Following Merity et al. (2018) we use 400 dimensional word embeddings and a 3 layer LSTM with hidden dimension 1150 to encode tokens. We also employ the same regularization strategy (DropConnect (Wan et al., 2013) + Dropout(Srivastava et al., 2014)) and weight tying approach. However, we perform optimization using Adam (Kingma and Ba, 2015) with learning rate 1e-3 instead of NT-ASGD, having found that it is more stable.

5.2 Results

Perplexity We evaluate our model using the standard *perplexity* metric: $\exp\left(\frac{1}{T}\sum_{t=1}^{T} \log p(x_t)\right)$. However, perplexity suffers from the issue that it

	PPL	UPP
ENTITYNLM* (Ji et al., 2017)	85.4	189.2
EntityCopyNet*	76.1	144.0
AWD-LSTM (Merity et al., 2018)	[illegible]	165.8
KGLM*	[illegible]	**88.5**

Table 3: **Perplexity Results** on *Linked WikiText-2*. Results for models marked with * are obtained using importance sampling.

overestimates the probability of out-of-vocabulary tokens when they are mapped to a single UNK token. This is problematic for comparing the performance of the KGLM to traditional language models on *Linked WikiText-2* since there are a large number of rare entities whose alias tokens are out-of-vocabulary. That is, even if the KGLM identifies the correct entity and copies the correct alias token with high probability, other models can attain better perplexity by assigning a higher probability to UNK. Accordingly, we also measure *unknown penalized perplexity* (UPP) (a.k.a *adjusted perplexity*) introduced by Ueberla (1994), and used recently by Ahn et al. (2016) and Spithourakis and Riedel (2018). This metric penalizes the probability of UNK tokens by evenly dividing their probability mass over $\mathcal{U}$, the set of tokens that get mapped to UNK . We can be compute UPP by replacing $p(\text{UNK})$ in the perplexity above by $\frac{1}{|\mathcal{U}|}p(\text{UNK})$, where $|\mathcal{U}|$ is estimated from the data.

We present the model perplexities in Table 3. To marginalize over annotations, perplexities for the ENTITYNLM, EntityCopyNet, and KGLM are estimated using the importance sampling approach described in Section 4. We observe that the KGLM attains substantially lower perplexity than the other entity-based language models (44.1 vs. 76.1/85.4), providing strong evidence that leveraging knowledge graphs is crucial for accurate language modeling. Furthermore, KGLM significantly outperforms all models in unknown penalized perplexity, demonstrating its ability to generate rare tokens.

Fact Completion Since factual text generation is our primary objective, we evaluate the ability of language models to complete sentences with factual information. We additionally compare with the *small* GPT-2 (Radford et al., 2019), a language model trained on a much larger corpus of text. We select 6 popular relations from Freebase, and write a simple *completion* template for each, such as "*X was born in* ___" for the *birthplace* relation. We generate sentences for these templates for a number of (X, Y) pairs for which the relation holds, and manually examine the first token generated by each language model to determine whether it is correct.

	AWD-LSTM	GPT-2	KGLM	
			Oracle	NEL
nation-capital	0 / 0	**6 / 7**	0 / 0	0 / 4
birthloc	0 / 9	14 / 14	**94 / 95**	85 / 92
birthdate	0 / 25	8 / 9	**65 / 68**	61 / 67
spouse	0 / 0	**2** / 3	**2** / 2	1 / **19**
city-state	0 / 13	**62 / 62**	9 / 59	4 / 59
book-author	0 / 2	0 / 0	**61 / 62**	25 / 28
Average	0.0/8.2	15.3/15.8	**38.5/47.7**	29.3/44.8

Table 4: **Fact Completion**. Top-k accuracy (@1/@5,%) for predicting the next token for an incomplete factual sentence. See examples in Table 5.

Table 4 presents performance of each language model on the relations. The *oracle* KGLM is given the correct entity annotation for X, while the *NEL* KGLM uses the discriminative model used for importance sampling combined with the NEL entity linker to produce an entity annotation for X.

Amongst models trained on the same data, both KGLM variants significantly outperform AWD-LSTM; they produce accurate facts, while AWD-LSTM produced generic, common words. KGLMs are also competitive with models trained on orders of magnitude more data, producing factual completions that require specific knowledge, such as birthplaces, dates, and authors. However, they do not capture facts or relations that frequently appear in large corpora, like the cities within states.[3] It is encouraging to see that the KGLM with automatic linking performs comparably to oracle linking.

We provide examples in Table 5 to highlight qualitative differences between KGLM, trained on 600 documents, and the recent state-of-the-art language model, GPT-2, trained on the WebText corpus with over 8 million documents (Radford et al., 2019). For examples that both models get factually correct or incorrect, the generated tokens by KGLM are often much more specific, as opposed to selection of more popular/generic tokens (GPT-2 often predicts "New York" as the birthplace, even for popular entities). KGLM, in particular, gets factual statements correct when the head or tail entities are rare, while GPT-2 can only complete facts for more-popular entities while using more-generic tokens (such as "*January*" instead of "*20*").

[3]This is not a failure of the KG, but of the model's ability to pick the correct relation from the KG given the prompt.

	Input Sentence	Gold	GPT-2	KGLM
Both correct	Paris Hilton was born in ___	New York City	New	1981
	Arnold Schwarzenegger was born on ___	1947-07-30	July	30
KGLM correct	Bob Dylan was born in ___	Duluth	New	Duluth
	Barack Obama was born on ___	1961-08-04	January	August
	Ulysses is a book that was written by ___	James Joyce	a	James
GPTv2 correct	St. Louis is a city in the state of ___	Missouri	Missouri	Oldham
	Richard Nixon was born on ___	1913-01-09	January	20
	Kanye West is married to ___	Kim Kardashian	Kim	the
Both incorrect	The capital of India is ___	New Delhi	the	a
	Madonna is married to ___	Carlos Leon	a	Alex

Table 5: **Completion Examples**. Examples of fact completion by KGLM and GPT-2, which has been trained on a much larger corpus. GPT-2 tends to produce very common and general tokens, such as one of a few popular cities to follow "*born in*". KGLM sometimes makes mistakes in linking to the appropriate fact in the KG, however, the generated facts are more specific and contain rare tokens. We omit AWD-LSTM from this figure as it rarely produced tokens apart from the generic "*the*" or "*a*", or "⟨*UNK*⟩".

Effect of changing the KG For most language models, it is difficult to control their generation since *factual* knowledge is entangled with generation capabilities of the model. For KGLM, an additional benefit of its use of an external source of knowledge is that KGLM is directly controllable via modifications to the KG. To illustrate this capability with a simple example, we create completion of "*Barack Obama was born on* _" with the original fact (Barack Obama, *birthDate*, 1961-08-04), resulting in the top three decoded tokens as "*August*", "*4*", "*1961*". After changing the birth date to 2013-03-21, the top three decoded tokens become "*March*", "*21*", "*2013*". Thus, changing the fact in the knowledge graph directly leads to a corresponding change in the model's prediction.

6 Related Work

Knowledge-based language models Our work draws inspiration from two existing knowledge-based language models:

(i) EntityNLM (Ji et al., 2017) which improves a language model's ability to track entities by jointly modeling named entity recognition and coreference. Our model similarly tracks entities through a document, improving its ability to generate factual information by modeling entity linking and relation extraction.

(ii) The neural knowledge language model (NKLM) (Ahn et al., 2016) which established the idea of leveraging knowledge graphs in neural language models. The main differentiating factor between the KGLM and NKLM is that the KGLM operates on an entire knowledge graph and can be evaluated on text without additional conditioning information, whereas the NKLM operates on a relatively smaller set of predefined edges emanating from a single entity, and requires that entity be provided as conditioning information ahead of time. This requirement precludes direct comparison between NKLM and the baselines in Section 5.

Data-to-text generation Our work is also related to the task of neural data-to-text generation. For a survey of early non-neural text generation methods we refer the reader to Reiter and Dale (1997). Recent neural methods have been applied to generating text from tables of sports statistics (Wiseman et al., 2017), lists and tables (Yang et al., 2017), and Wikipedia info-boxes (Lebret et al., 2016). The primary difference between these works and ours is our motivation. These works focus on generating coherent text within a narrow domain (e.g. sports, recipes, introductory sentences), and optimize metrics such as BLEU and METEOR score. Our focus instead is to use a large source of structured knowledge to improve language model's ability to handle rare tokens and facts on a broad domain of topics, and our emphasis is on improving perplexity.

General language modeling Also related are the recent papers proposing modifications to the AWD-LSTM that improve performance on *Wikitext-2* (Gong et al., 2018; Yang et al., 2018; Krause et al., 2018). We chose to benchmark against AWD-LSTM since these contributions are orthogonal, and many of the techniques are compatible with the KGLM. KGLM improves upon AWD-LSTM, and we expect using KGLM in conjunction with these methods will yield further improvement.

7 Conclusions and Future Work

By relying on memorization, existing language models are unable to generate factually correct text about real-world entities. In particular, they are unable to capture the long tail of rare entities and word types like numbers and dates. In this work, we proposed the *knowledge graph language model* (KGLM), a neural language model that can access an external source of facts, encoded as a knowledge graph, in order to generate text. Our implementation is available at: https://github.com/rloganiv/kglm-model. We also introduced *Linked WikiText-2* containing text that has been aligned to facts in the knowledge graph, allowing efficient training of the model. *Linked WikiText-2* is freely available for download at: https://rloganiv.github.io/linked-wikitext-2. In our evaluation, we showed that by utilizing this graph, the proposed KGLM is able to generate higher-quality, factually correct text that includes mentions of rare entities and specific tokens like numbers and dates.

This work lays the groundwork for future research into knowledge-aware language modeling. The limitations of the KGLM model, such as the need for marginalization during inference and reliance on annotated tokens, raise new research problems for advancing neural NLP models. Our distantly supervised approach to dataset creation can be used with other knowledge graphs and other kinds of text as well, providing opportunities for accurate language modeling in new domains.

Acknowledgements

First and foremost, we would like to thank Stephen Merity for sharing the materials used to collect the *WikiText-2* dataset, and Nitish Gupta for modifying his entity linker to assist our work. We would also like to thank Dheeru Dua and Anthony Chen for their thoughtful feedback. This work was supported in part by Allen Institute of Artificial Intelligence (AI2), and in part by NSF award #IIS-1817183. The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. ArXiv:1608.00318.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proc. of NeurIPS*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.

Thiago Castro Ferreira, Diego Moussallem, Emiel Krahmer, and Sander Wubben. 2018. Enriching the WebNLG corpus. In *Proc. of INLG*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proc. of INLG*.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Frage: frequency-agnostic word representation. In *Proc. of NeurIPS*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proc. of ACL*.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proc. of EMNLP*.

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

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *Proc. of EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2018. Dynamic evaluation of neural sequence models. In *Proc. of ICML*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proc. of EMNLP*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *Proc. of ICLR*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proc. of ICLR*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. of INTERSPEECH*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

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 *Proc. of AAAI*.

Georgios P. Spithourakis and Sebastian Riedel. 2018. Numeracy for language models: Evaluating and improving their ability to predict numbers. In *Proc. of ACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Trieu H. Trinh and Quoc V. Le. 2019. Do language models have common sense? In *Proc. of ICLR*.

Joerg Ueberla. 1994. Analysing a simple language modelÂ·some general conclusions for language models for speech recognition. *Computer Speech & Language*, 8(2):153 – 176.

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

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proc. of ICML*.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proc. of EMNLP*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *Proc. of ICLR*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proc. of EMNLP*.