Latent Relation Language Models

Hiroaki Hayashi,[1*] **Zecong Hu,**[1*] **Chenyan Xiong,**[2] **Graham Neubig**[1]
[1]Carnegie Mellon University, [2]Microsoft Research AI
{hiroakih, zeconghu, gneubig}@cs.cmu.edu, Chenyan.Xiong@microsoft.com

Abstract

In this paper, we propose Latent Relation Language Models (LRLMs), a class of language models that parameterizes the joint distribution over the words in a document and the entities that occur therein via knowledge graph relations. This model has a number of attractive properties: it not only improves language modeling performance, but is also able to annotate the posterior probability of entity spans for a given text through relations. Experiments demonstrate empirical improvements over both word-based language models and a previous approach that incorporates knowledge graph information. Qualitative analysis further demonstrates the proposed model's ability to learn to predict appropriate relations in context.[†]

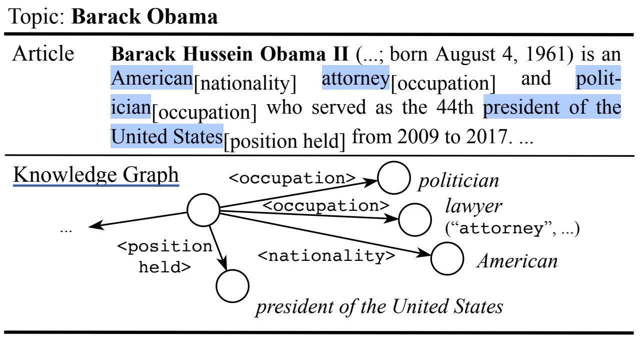
1 Introduction

Language models (LMs) calculate the probability $P(X)$ of textual data X, and are a core model class of interest to NLP. LMs are used as testbeds for evaluation of generative models of text, and have applications such as rescoring of upstream language generation inputs (Sundermeyer, Schlüter, and Ney 2012), grammatical error correction (Felice et al. 2014), or pre-training of sentence representations (Peters et al. 2018). Neural networks are used to model this probability in state-of-the-art LMs (Bengio et al. 2003; Mikolov et al. 2010; Merity et al. 2017).

Textual data X comprise a wide variety of words to be modeled, from closed-class function words, to common nouns or verbs, to named entities and numbers (Zipf 1949). Notably, words on the rarer end of this spectrum are often more semantically or topically important, as evidenced by the success of heuristics such as TF-IDF (Salton and McGill 1986), which up-weight words with low frequency. Previous work has noted that while neural LMs greatly outperform alternatives such as n-gram models on frequent words, they often under-perform on these rare words due to their limited parameter budget, which puts them at a disadvantage compared to non-parametric models like count-based n-grams (Neubig and Dyer 2016).

[*]Equal Contribution.

[†]Code & Data: https://github.com/neulab/lrlm.



Topic: **Barack Obama**

Article: **Barack Hussein Obama II** (...; born August 4, 1961) is an American[nationality] attorney[occupation] and politician[occupation] who served as the 44th president of the United States[position held] from 2009 to 2017. ...

Figure 1: Overview of our task of language modeling conditioned on a knowledge graph. For a given topic, we want to learn a language model that leverages the knowledge graph through relations when modeling the text.

Methods to mitigate this bottleneck have been proposed in the context of *conditional LMs*, which instead model the conditional probability $P(X \mid C)$, where C is some context given to the model. For instance, in sequence transduction tasks, there are mechanisms to copy from the source sequence (Gu et al. 2016) or use word or phrase dictionaries (Arthur, Neubig, and Nakamura 2016) to improve modeling of low-frequency words. Perhaps more interesting from an LM perspective are methods conditioned on information from structured knowledge sources such as knowledge graphs (Ahn et al. 2016; Parvez et al. 2018; Logan et al. 2019), tables (Lebret, Grangier, and Auli 2016), or grammars (Konstas and Lapata 2013). These methods are analogous to human language production, where the underlying knowledge is converted into linguistic realizations.

In this work, we propose Latent Relation Language Models (LRLMs), a class of conditional LMs that take *relational* information between entities in a knowledge graph as context. Specifically, our model is able to generate either words from a fixed word vocabulary, or a span of words defined according to their relations with a topic entity of interest, as shown in Figure 1. The choices of which method of generation to use is defined as a latent variable sequence Z. We

Span level

Alignment knowledge to raw text.

use Latent Predictor Networks (LPNs; Ling et al. (2016)) to jointly learn $P(X, Z \mid C)$, thus tractably marginalizing over all the possible spans. Compared to other word-by-word generation methods that condition LMs on knowledge graphs (KGs; Ahn et al. (2016); Wang et al. (2018)), the span-based generation from the KGs alleviates problems of malformed or incomplete mentions. Moreover, the posterior probabilities of Z can be considered as entity links, which are of interest in their own right in the information extraction field (Ceccarelli et al. 2013; Ganea and Hofmann 2017).

We apply the model on articles from Wikipedia (X), with the help of relational information (C) such as Wikidata (Vrandečić and Krötzsch 2014) or Freebase (Bollacker et al. 2008) regarding each article topic. Empirical results on open vocabulary language modeling show that the proposed model outperforms previous approaches on the same task, demonstrating that LRLMs provide an effective way to condition on this context. We also demonstrate the merit of explicitly modeling latent relations by examining the posterior probabilities over the chosen relations Z, which are in concert with human intuitions about how relations are being expressed in the text.

2 Language Modeling Conditioned on Structured Knowledge

In this section, we define the task of open-vocabulary language modeling conditioned on structured data.

Task Definition

Knowledge graphs (KGs) can be represented as a directed labeled graph $G = (V, E)$ consisting of a set of nodes $V = \{v_1, \ldots, v_{|V|}\}$ and a set of relation edges $E = \{e_i{:}\langle s_i, \omega_i, o_i\rangle \mid s_i, o_i \in V,\ \omega_i \in R\}$. Relation e_i contains s_i, ω_i, and o_i as the subject, relation type, and object. R is the set of all relation types. Each node $v_i \in V$ represents either an entity or an attribute[1], and is associated with a set of surface forms (also called aliases) $\mathcal{A}(v_i) = \{a_{i,1}, \ldots, a_{i,|\mathcal{A}(v_i)|}\}$ that can be used to refer to v_i. For instance in Figure 1, the subject "*Barack Obama*" is connected to both "*politician*" and "*lawyer*" with the relation `<occupation>`, and the object entity "*politician*" has "`political figure`" and "`polit.`" as additional aliases. Notably surface forms of many objects in the KG can be multiple words, and thus it is necessary to have machinery to deal with this fact.

Given this KG, we further define a topic entity s about which we would like to generate a piece of text. Our conditional language modeling problem is then defined as the problem of modeling the conditional probability of text X: $P(X \mid G, s)$. In particular, we consider a subgraph $G' = (V', E')$ of the original KG G by extracting nodes and edges directly related to the topic entity s:

node set $V' : \{s\} \cup \{o_i \mid \langle s, *, o_i\rangle \in E\}$,

relation set $E' : \{e_i{:}\langle s, \omega_i, o_i\rangle \mid \langle s, \omega_i, o_i\rangle \in E \wedge o_i \in V'\}$.

[1]A value specified with a relation from an entity (e.g., dates).

We consider an *open-vocabulary* setting where all word types within X are incorporated. Perplexity under this setting provides a more realistic measure than under closed-vocabulary setting by taking into account words that rarely or never appear in the training set, which, as previously noted, are particularly important for conveying the main content of the text.

Why Condition on Knowledge Graphs?

KGs provide two important benefits for neural LMs. First, the high coverage of rarer words due to entities being often infrequent addresses lack of textual supervision for predicting these words. More importantly, KGs have the potential to help LMs generate *factually consistent* text by providing consistent associations between entities. Normal LMs would have to rely on supervision purely from textual data, which may not provide a learning signal strong enough to accurately generate these facts. For instance, results from Radford et al. (2019) show that even with a very large model trained on massive amounts of data, samples can be factually incorrect, although being fluent and coherent.

3 Latent Relation Language Models

In this setion, we describe our proposed framework of Latent Relation Language Models (LRLMs).

Definition

Knowledge from the KG subgraph G' can be incorporated into generation by copying aliases from related entities into the generated text. For instance in Figure 2, to generate Obama's birth date, the model can of course pick words from its vocabulary. But it is more straightforward to copy from the `<birth date>` relation of the topic entity "*Barack Obama*", which gives the correct birth date.

However, it is insufficient to model probabilities for such choices conditioning only on G' and s, because it is unknown to us which text spans are matched to which relations. Naïve solutions like simple text matching algorithms would yield many false positives. For example, "*New York City*" has an alias "`New York`", which matches "*New York*" (state) and parts of "*New York City Council*".

To circumvent this lack of relation annotation, we treat relations corresponding to such text spans as latent variables. Formally, let $X = \{x_i\}_{i=1}^{N}$ be the sequence of N tokens, and $Z = \{(\sigma_t, \pi_t, \rho_t)\}_{t=1}^{T}$ a sequence of latent variable triplets describing text span matches:

- The *span* variable $\sigma_t := (\ell_t, r_t)$ specifies a token subsequence $x_{\sigma_t} = \{x_i\}_{i=\ell_t}^{r_t}$.
- The *source* variable $\pi_t \in \{\text{REL}, \text{WORD}\}$ denotes the generation source of the span x_{σ_t}.
- The *relation* variable $\rho_t := (e_t, a_t)$ describes the matching relation and surface form of the span x_{σ_t}, and is only used when $\pi_t = \text{REL}$.

For Z to be a valid sequence of latent variables, the following conditions must be satisfied:

- Span variables $\{\sigma_t\}_{t=1}^{T}$ form a *segmentation* of X, *i.e.*, $\ell_t = r_{t-1} + 1$ for $t = 2, \ldots, T$. This also implies $T \le N$.

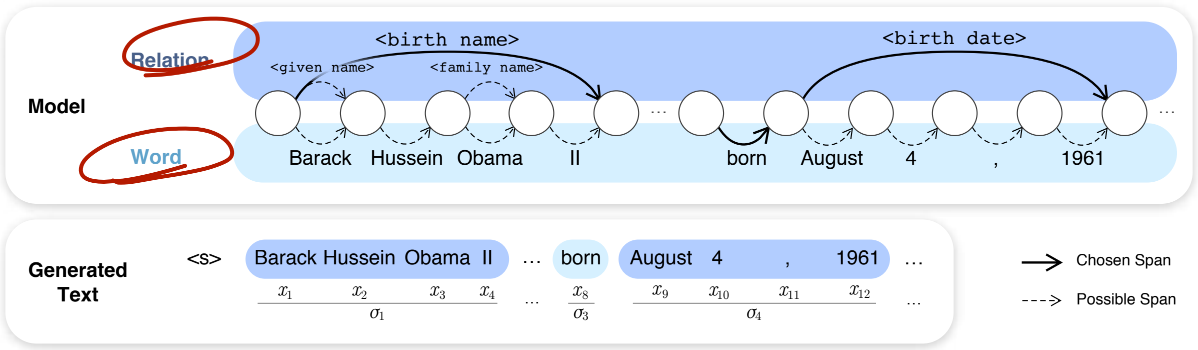


Figure 2: While generating, our model switches between the two *sources*: "Relation" and "Word". Circles represent hidden states up to each token, and edges represent possible span matches. Here we show one valid derivation with solid lines, and other options as dashed lines. We also show an "annotation" of the generated tokens by the spans and sources we choose.

- If $\pi_t = $ WORD, then $\ell_t = r_t$.
- If $\pi_t = $ REL, then $\rho_t = (e_t, a_t)$ where $e_t = \langle s, \omega_t, o_t \rangle$ should satisfy $e_t \in E'$, $a_t \in \mathcal{A}(o_t)$, and $x_{\sigma_t} = a_t$, *i.e.*, ρ_t must correspond to a valid surface form of an object that is related to the topic entity s and matches the text span.

Let $\mathcal{Z}$ be the set of all valid latent variable sequences. We can now model the probability by marginalizing over $\mathcal{Z}$:

$$P(X \mid G', s) = \sum_{Z \in \mathcal{Z}} P(X, Z \mid G', s). \quad (1)$$

For sake of brevity, unless noted otherwise, we drop G' and s from the conditions in the following sections.

Training

Given the latent variable sequence Z, we follow Ling et al. (2016) in factoring the joint probability:

$$P(X, Z) = \prod_{t=1}^{T} P(\sigma_t, \pi_t, \rho_t, x_{\sigma_t} \mid x_{<\ell_t})$$
$$= \prod_{t=1}^{T} P(\pi_t \mid x_{<\ell_t}) P(\sigma_t, x_{\sigma_t}, \rho_t \mid \pi_t, x_{<\ell_t}),$$

here $x_{<i}$ is the sequence of first $i-1$ tokens in X. Figure 2 shows an example of generation according to this factorization, and Algorithm 1 precisely defines the process of generating at time step t.

We marginalize over $\mathcal{Z}$ according to Eq 1 and optimize for the marginal likelihood. Since the probability at time step t is independent of previous latent variables, the marginalization is tractable using the forward-backward algorithm (Baum et al. 1970). The forward probability α_i is defined as the marginal probability of the sequence up to the i-th token (specifically, $\alpha_0 = 1$), computed as follows:

$$\alpha_i = \sum_{(\sigma:(\ell,r),\pi,\rho) \in \tau_i} \alpha_{\ell-1} P(\sigma, \pi, \rho, x_\sigma \mid x_{<\ell}),$$

自底向上

where τ_i is defined as the set of valid latent variable tuples $(\sigma : (\ell, r), \pi, \rho)$ such that $r = i$, *i.e.*, all valid spans ending at the i-th token. The marginal probability we optimize for is then α_N. The backward probability β_i which is required for gradient computation can be similarly calculated.

Parameterization

We use neural networks to parameterize all probability distributions mentioned above. Decisions for time step t are based on a D-dimensional hidden state $\mathbf{h}_{\ell_t}$. This hidden state can be generated by any neural sequence model, and we experiment with multiple models to demonstrate the generality of our approach.

Source Selection Source selection is done using a simple linear model followed by a softmax function applied to the latest word-level hidden state $\mathbf{h}_{\ell_t}$:

$$P(\pi_t \mid x_{<\ell_t}) = \text{softmax}(\mathbf{W}_\pi \mathbf{h}_{\ell_t} + \mathbf{b}_\pi),$$

where $\mathbf{W}_\pi \in \mathbb{R}^{2 \times D}$, $\mathbf{b}_\pi \in \mathbb{R}^2$ are trainable parameters.

Word Generation Like conventional word-level neural language models, we have the option to generate the next token from a fixed vocabulary. This option is used to generate any word that isn't part of an object entity participating in a relation. The probability is:

$$P(x_{\ell_t} \mid x_{<\ell_t}) = \text{softmax}(\text{Linear}_w(\mathbf{h}_{\ell_t})),$$

where $\text{Linear}(\mathbf{h})$ is a linear transform with a bottleneck of dimension K into a vector over vocabulary size L:

$$\text{Linear}(\mathbf{h}) = \mathbf{W}_1(\mathbf{W}_2 \mathbf{h} + \mathbf{b}_2) + \mathbf{b}_1,$$

where $\mathbf{W}_1 \in \mathbb{R}^{L \times K}$, $\mathbf{b}_1 \in \mathbb{R}^L$, $\mathbf{W}_2 \in \mathbb{R}^{K \times D}$, $\mathbf{b}_2 \in \mathbb{R}^D$ are trainable parameters. Empirically we found this low-rank version to outperform a full linear transform.

Unknown Word Generation Since our task is language modeling under an open-vocabulary setting, we must be able to generate words even if they are out of vocabulary. Following Luong and Manning (2016), we do so by having a character-level LM "spell-out" any unknown words. If the unknown word is $x = c_1 \ldots c_{|c|}$ with $|c|$ characters:

$$P(x \mid x_{<\ell_t}) = P(\texttt{<UNK>} \mid x_{<\ell_t}) P(c_1 \ldots c_{|c|}; \theta_{\text{char}}),$$

where θ_{char} are the parameters of the character LM. We pre-train this model on the set of all unique words in the training set and fix its parameters while training LRLM.

Algorithm 1 Generative Process of LRLM

```
Input  previous span $\sigma_{t-1} = (\ell_{t-1}, r_{t-1})$, previously generated tokens $x_{<r_{t-1}}$.
Output  span $\sigma_t = (\ell_t, r_t)$, source $\pi_t$, relation $\rho_t = (e_t, a_t)$, and token subsequence $x_{\sigma_t}$.
 1: $\ell_t \leftarrow r_{t-1} + 1$        t step                ▷ Update the beginning of span. :1
 2: $\widehat{\pi}_t \sim P(\pi_t \mid x_{<\ell_t})$        ▷ Choose whether to generate a word or relation. :2
 3: if $\widehat{\pi}_t$ = WORD then                              ▷ Generating a word. :3
 4:     $P(\sigma_t, x_{\sigma_t}, \rho_t \mid \pi_t = \text{WORD}, x_{<\ell_t}) := P(x_{\ell_t} \mid x_{<\ell_t})$     ▷ Simplify the probability. :4
 5:     $\widehat{x}_{\ell_t} \sim P(x_{\ell_t} \mid x_{<\ell_t})$      ▷ Choose a word from model vocabulary. :5
 6:     if $\widehat{x}_{\ell_t}$ = <UNK> then
 7:         $\widehat{x}_{\ell_t} \sim P(c_1 \ldots c_{|c|}; \theta_{\text{char}})$     ▷ Generate a word using a character model. :7
 8:     else if $\widehat{x}_{\ell_t}$ = <EOS> then
 9:         End generation.
10:     end if
11: else if $\widehat{\pi}_t$ = REL then                           ▷ Generating a relation. :11
12:     $P(\sigma_t, x_{\sigma_t}, \rho_t \mid \pi_t = \text{REL}, x_{<\ell_t}) := P(e_t \mid x_{<\ell_t}) P(a_t \mid e_t, x_{<\ell_t})$     ▷ Factor the probability. :12
13:     $\widehat{e}_t \sim P(e_t \mid x_{<\ell_t})$                ▷ Choose a relation. :13
14:     $\widehat{a}_t \sim P(a_t \mid \widehat{e}_t, x_{<\ell_t})$    span    ▷ Choose a surface form from the selected relation. :14
15:     $\widehat{x}_{\sigma_t} \leftarrow \widehat{a}_t$            ▷ Generate a phrase. :15
16: end if
```

Table 1: Training set statistics: number of training documents, vocabulary size, relations per head entity, tokens per document, and entity mentions per document.

Dataset	Doc	Vocab	Rel/Ent	Tok/Doc	Ment/Doc
WikiFacts	7856	40.0k	82.71	157.25	16.04
WikiText-S	27685	71.1k	11.38	295.75	11.20
WikiText-F	27685	264k	11.38	3559.91	73.01

Relation Generation The goal of relation generation is to find the most suitable span that can be copied into the text. As Line 12 of Algorithm 1 depicts, this is factorized into two steps: relation selection and surface form selection.

- **Relation selection.** We utilize pre-trained KG embeddings from OpenKE (Han et al. 2018) for entities and relation types. For a relation $e_i\colon \langle s, \omega_i, o_i \rangle$, we concatenate KG embeddings for ω_i and o_i to obtain the *relation embedding* $\mathbf{e}_i$.[2] We then compute the probability of selecting each relation as:

$$P(e_i \mid x_{<\ell_t}) = \text{softmax}(\mathbf{e}_i^\top \text{Linear}_o(\mathbf{h}_{\ell_t})).$$

- **Surface form selection.** We featurize surface forms via fastText embeddings (Bojanowski et al. 2017) pre-trained on the training corpus, and calculate probability of surface form a_k as:

$$P(a_k \mid e_i, x_{<\ell_t}) = \text{softmax}(\mathbf{f}_{a_k}^\top (\mathbf{W}_a \mathbf{h}_{\ell_t} + \mathbf{b}_a)),$$

where $\mathbf{f}_{a_k}$ is the fastText embedding for a_k and $\mathbf{W}_a$, $\mathbf{b}_a$ are trainable parameters.

4 Datasets

We use two datasets with different characteristics for experiments; statistics are shown in Table 1.

WikiFacts

WikiFacts (Ahn et al. 2016) is a collection of Wikipedia articles restricted to `/film/actor` domain entities in Freebase (Bollacker et al. 2008).[3] Each example consists of the *first section* of the original article. Since official splits for evaluation are not provided, we follow previous work and performed a random split of 80/10/10%.

This dataset assumes a single alias for each entity (*i.e.*, $\forall o \in V'; |\mathcal{A}(o)| = 1$). Hence, the surface form selection module acts as oracle, where it always assigns a probability of 1 to the correct surface form.

WikiText

While WikiFacts has been used in previous work on LMs using structured data (Ahn et al. 2016), the domain is limited. To investigate the capability of knowledge-infused LMs in an open-domain setting with a wide variety of relations, we build a large-scale open-domain dataset from the existing WikiText-103 dataset (Merity et al. 2017) by associating articles with entities in Wikidata (Vrandečić and Krötzsch 2014). We employ the same data splits from the original dataset. Bridging KGs and the articles from WikiText-103 involves two steps (more details in Appendix A).

- **Constructing subgraphs for articles.** As discussed in Section 2, we take the original KG and extract a relevant subgraph G' for each article. While there are many options on how to extract this subgraph, we choose the subgraph G' consisting of *direct neighbors* of the topic entity for each article. This forms a star-shaped subgraph, with the topic entity as the central node, connected by the related entities and attributes. We found on average 11.38 neighbors and 3.1 surface forms for each neighbor.

[2] We train embeddings for each relation type not covered by pre-trained embeddings, and an UNK embedding for attributes and entities not covered by pre-trained embeddings.

[3] The original WikiFacts also includes topic entities from other articles linked to the page to be generated. However, these (gold) entities are inaccessible when actually attempting to generate new articles. We experiment without them, but also report results with them in Appendix C.

Table 2: Perplexity values of different models on open vocabulary language modeling, lower is better. Best results are in bold. Asterisk symbols represent statistical significance according to Wilcoxon signed-rank test (Dror et al. 2018) against the best baseline model, with $p < 0.05$ (*) and $p < 0.01$ (**), respectively.

Base model	Dataset	Dev				Test			
		Vanilla LM	Alias LM	NKLM	LRLM	Vanilla LM	Alias LM	NKLM	LRLM
LSTM	WikiFacts	231.03	213.34	96.77	**93.55**	225.40	207.57	93.18	**88.37***
	WikiText-S	68.37	70.07	46.16	**45.84**	86.12	87.75	55.98	**55.38**
	WikiText-F	45.13	46.18	44.46	**42.18***	49.47	50.88	48.54	**45.70***
Transformer-XL	WikiFacts	172.27	158.54	99.46	**84.76****	167.91	154.27	94.36	**79.35****
	WikiText-S	42.63	39.65	43.05	**37.75****	52.96	50.60	52.51	**44.98****
	WikiText-F	30.14	31.20	32.19	**29.56****	33.01	34.37	35.27	**32.20****

- **Linking mentions with the KG.** For each object entity in G', we search for occurrences of all surface forms in the article while allowing token overlaps among them. Note that, similarly to distant supervision for relation extraction (Mintz et al. 2009), this string-matching process can produce false positive mentions. We rely on our model's ability to handle such noisy mentions by learning to assign high probabilities only on the correct mentions.

We name the dataset obtained through this process as WikiText-F (Full). We also create WikiText-S (Short) by only using the first sections of WikiText-F documents.

5 Experimental Settings

In this section, we explain the evaluation metric, configurations, and baseline models compared against LRLM.

Evaluation Measure

We report token-level perplexity under the *open-vocabulary* setting. We use pre-trained character-level LMs from Section 3 for each dataset to discount the probability of out-of-vocabulary words based on its spelling.[4] This is done for all tested models, both proposed and baselines.

Model Configuration

For WikiFacts, we use a fixed word vocabulary size of 40,000 following Ahn et al. (2016). For WikiText-derived datasets, we include all words with frequencies no less than 3 in our dataset following Merity et al. (2017). We use adaptive embeddings (Baevski and Auli 2019) and adaptive softmax (Grave et al. 2017) to handle large vocabulary.

To calculate the hidden state $\mathbf{h}_{x_{<i}}$, we test two varieties of neural sequence models: standard LSTMs (Hochreiter and Schmidhuber 1997), and the state-of-the-art Transformer-XL (Dai et al. 2019). We implement all models in PyTorch (Paszke et al. 2017). Training details and hyperparameters are summarized in Appendix B.

[4]This contrasts to UPP (Ueberla 1994), which adjusts likelihood of OOV words based on a uniform probability equivalent to the size of the vocabulary, which does not actually measure the ability to generate words outside of training data. Results using closed vocabulary setting or UPP can be found in Appendix C and E, respectively.

Baselines

We compare LRLM against three baselines that utilizes information from KGs to various degrees.

Vanilla language model (Vanilla LM) This is a standard language model baseline that does not condition on KGs, such as LSTM (Merity, Keskar, and Socher 2017) or Transformer-XL (Dai et al. 2019).

Alias-prepended language model (Alias LM) The same model as above, but prepending to the text the concatenated aliases of all entities in G' which appear in the article.[5] This gives a simple baseline LM conditioned on the KG.

Neural Knowledge Language Model (NKLM) Similarly to LRLM, the Neural Knowledge Language Model (NKLM; Ahn et al. (2016)) also has the ability to copy from a given set of KG triples, but differs from LRLM in several ways:

1. LRLM allows generation of multi-word entities at once, while NKLM predicts one word at a time and the model needs to repeatedly predict the right relation until copying of an object is done.
2. LRLM marginalizes over all derivations of a sequence, which allows processing of overlapped tokens among spans, while NKLM makes all decisions in a hard fashion and cannot handle such overlapped tokens.[6]

The original NKLM does not differentiate between surface forms, so we incorporate the same surface form selection module as LRLM for fair comparison.

6 Results and Analysis

In this section, we summarize the main results and perform analyses of the learned model.

Main Results

Perplexities over the datasets are shown in Table 2. We observe that for both sequence models, LRLM outperforms the baselines on all datasets and improvements are *more* significant on the stronger sequence model. Particularly on the

[5]This simulates the table-to-text generation setting, where all input entities should be mentioned in the generated sentence.

[6]Due to this limitation, we perform additional data preprocessing on WikiText for NKLM, detailed in Appendix D.

Warm-up: <s> = Sonic the Hedgehog (1991 video game) =	
Vanilla LM:	Sonic the Hedgehog is an action action - adventure video game published by Sonic of programmers for the 1999 Nintendo GameCube 's SNES video game Sonic the Hedgehog 2 . It was released for the Nintendo DS on September 16 , 1994 in North America and Europe in the latter part of the original Halo 2 . It was played in a post - apocalyptic fantasy fantasy universe , by Nintendo Computer Entertainment on March 6 , 1999 under the ...
Alias LM:	is the twelfth video game developed and published by EA Sports . It is the sequel to the 1992 Sonic - 6 gameSonic the Hedgehog for the Nintendo Genesis , created by Sonic system creator Pinball Ka . Doctor the Hedgehog has since gone on a hiatus in choosing an estimated global community when it quickly becomes a launch member . The game puts Princess unlock from a mansion , which once everything is devastated by a bro and I ...
NKLM:	Sonic the Hedgehog[TITLE] Sonic the Hedgehog[TITLE] Sonic the Hedgehog[TITLE] (Jordon VS . ストリートファイター , Picardi Sutorīto Faitā Jobs Sutorīto Faitā) is A[CERO rating] platform video game[instance of] developed by Sonic Team[developer] and published by Sega[publisher] . In 2008 , the game[instance of] was released for Sega[platform] 3[platform] and 1320 platforms in March 2009 , replacing the original game[instance of] Sony[platform] Drive[platform] counterpart Sonic the Hedgehog[characters] for the Android[platform] GameCube[platform] . It was re - released on March 12 , 2010 , in ...
LRLM:	Sonic the Hedgehog[TITLE] (also known as Sonic the Hedgehog 3 and Sonic[series] the Hedgehog 2) is a 1986 role - playing video game developed by Sonic Team[developer] and published by Sony Computer Entertainment (SEGA[publisher]) for the PlayStation 3[platform] (Xbox 360[platform]) . It was developed and published by Sega[publisher] in 1997 for the Wii , and was ported as a third installment in the Sonic the Hedgehog[series] series and released in Japan in 1996 . On the ...

$s =$ **Sonic the Hedgehog (1991 video game)**

$(\omega, o) = \{$

(`<TITLE>`,	*Sonic the Hedgehog (1991 video game)*),
(`<instance of>`,	*video game*),
(`<CERO rating>`,	*A*),
(`<developer>`,	*Sonic Team*),
(`<publisher>`,	*Sega*),
(`<platform>`,	*Sega Mega Drive*),
(`<platform>`,	*Wii*),
(`<platform>`,	*Nintendo GameCube*),
(`<platform>`,	*Xbox 360*),
(`<platform>`,	*Playstation 3*),
(`<platform>`,	*Android*),
(`<characters>`,	*Sonic the Hedgehog*),
(`<series>`,	*Sonic the Hedgehog (video game series)*),
...	

$\}$

Figure 3: Samples from the models for the topic entity "*Sonic the Hedgehog (1991 video game)*" with the corresponding subgraph on the right. Square brackets denote the relation type of copied objects. Highlighted spans in light green are full mentions, and those in dark red are partial mentions. Underlined tokens are unknown words sampled from the character model.

two WikiText-derived datasets, our model outperformed the simpler Vanilla LM and Alias LM baselines, while NKLM had difficulty utilizing the KGs and in some cases results in worse perplexities than these baselines. Alias LM underperformed Vanilla LM in some cases, demonstrating that this simpler and more indirect method of conditioning on the linearized KG is not sufficient to achieve stable improvements.

Generated Samples

To illustrate behaviors of the learned models, we take the models using Transformer-XL trained on WikiText-S, draw 10 samples while conditioning on G' and $s =$ "*Sonic the Hedgehog*", and show the sample with lowest perplexity in Figure 3. We highlight tokens generated by the relation predictor and use different colors to represent full and partial mentions. A *full mention* is an identical copy of an entity surface form, while a *partial mention* is an incomplete subphrase of an entity surface form. A perfect model should not generate partial mentions as it leads to possibly corrupted phrases, and should generate the same set of full mentions as the gold article.

Although NKLM generates more mentions, it suffers from generating partial mentions because it 1) is unaware of the length of surface forms, and 2) requires making copy decisions as many times as the surface form lengths. As shown in Figure 3, we often observe NKLM repeating the same entity, or switching entities halfway through (*e.g.*, "*Sega 3*"). In contrast, LRLM, by design, only generates full mentions.

We quantitatively show this in Table 3 by counting the average number of partial and full mentions in samples. We took 10 samples from 10 random topic entities in the development set, and manually annotated "valid" full mentions,

Table 3: Average number of partially generated, fully generated, and valid and invalid full mentions over 100 samples from the development set or gold human-generated article.

	Partial	Full	Valid	Invalid
NKLM	16.9	7.81	6.37	1.44
LRLM	–	6.32	5.63	0.69
Gold	–	9.00	9.00	0.00

which we deemed as semantically correct based on the sentential context. NKLM generates more invalid mentions than LRLM, most of which are false positives and repetitions of the same mention. LRLM has almost no repetitions, but sometimes incorrectly predicts the "theme" of the topic entity, *e.g.*, generating an article about a TV episode for a topic entity of a song.

Posterior Probability of Spans

One of the advantages of our model is its capability to calculate the posterior probability of a span being generated as a relation in existing text. We calculate the joint probability of a span ($\sigma = (\ell, r)$) and the surrounding text[7] by marginalizing over the latent variable Z for both sides of context, and normalize over all possible spans:

$$P(X, Z) = \alpha_{\ell-1} \cdot P(Z \,|\, x_{<\ell}) \cdot \beta_{r+1},$$
$$P(Z \,|\, X) = P(X, Z) \,/ \sum_{Z \in \mathcal{Z}} P(X, Z),$$

[7]We consider the text segment in the batch where the span appears as the surrounding text.

Table 4: Posterior probability of spans (underlined) in contexts. `word` represents word-based generation. The second relation in the last example means generation of "*the*" using `word`, followed by relation-based generation of "*United States*" using the `<origin>` relation.

Title: Sorry (Madonna Song)		
... song by American singer Madonna from her tenth ...		
Relations:	`<performer>`	**0.9697**
	`<lyrics by>`	0.0289
	`word`	0.0014
... written and produced by Madonna and Stuart Price , ...		
Relations:	`<performer>`	0.1545
	`<lyrics by>`	**0.7693**
	`word`	0.0762
... continuation from the " Hung Up " music video		
Relations:	`<follows>`	**1.0000**
	`word`	0.0000
... . However , in the United States , the song did ...		
Relations:	`<origin>`	0.0000
	`word` → `<origin>`	0.0003
	`word`	**0.9997**

where α_i and β_i are the forward and backward probabilities computed following Section 3. Table 4 shows spans with posterior probabilities of various relation types from an article about "*Sorry (Madonna song)*". The model demonstrates the ability to relate the entity "*Madonna*" to the topic with appropriate relation types based on context. We also observe that the model tends to generate multi-word spans through relations rather than word-by-word from vocabulary. However, our model often favors word-based generation for common phrases even if related entities exist.

Effect of Subgraph Size

Finally, we measure the performance of models with respect to the richness of resources available for conditioning. We group WikiFacts articles into 10 bins by the number of relations available, and plot binned word-average log-probabilities in Figure 4. While all models have slightly higher log-probabilities as the number of relations increase, LRLM achieves the largest gain.

7 Related Work

A variety of entity-aware LMs exist, conditioning on information sources such as coreference annotations (Ji et al. 2017), entity annotations (Logan et al. 2019), or keywords (Kiddon, Zettlemoyer, and Choi 2016; Parvez et al. 2018). Among them, NKLM (Ahn et al. 2016) uses relational information and is the most relevant. Our proposed LRLM formulation is more successful at lowering perplexity and allows calculating posterior probabilities of relations.

Incorporating KGs for natural language generation (NLG) has a long history (Goldberg, Driedger, and Kittredge 1994; Reiter et al. 2005; Chen and Mooney 2008). With the

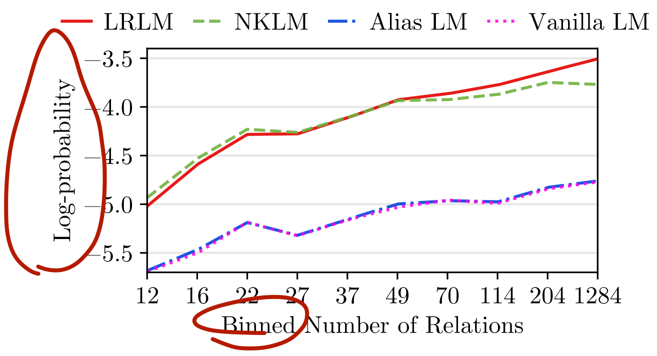


Figure 4: Word-average log-probabilities on development set of WikiFacts grouped by the average number of relations. LRLM shows a larger gain over the baselines as the number of relations increases.

recent advancement of neural sequence modeling, prevalent approaches for language generation from KGs employ sequence-to-sequence models with special attention mechanisms tailored for input structures such as graphs (Wang et al. 2018) or tables (Liu et al. 2018). Unlike our focus, however, this class of research focuses on learning discriminative models that do not explicitly generate the referent entity as latent variables, like we do in Section 6.

While not directly related to our core task, there have been a number of other methods for incorporating latent variables into NLG problems. Latent structure has included predicting latent sequences of topics (Wiseman, Shieber, and Rush 2018), chunking of word sequences into n-grams (Buckman and Neubig 2018), deciding between input sources (Gu et al. 2016), or generating compressed summary tokens (Miao and Blunsom 2016). Our model borrows its underlying structure from Ling et al. (2016), who focused on an entirely different task of source code generation. We use a similar method for selecting latent sources for Wikipedia article language modeling with a repository of KG triples.

8 Conclusion

In this work, we propose Latent Relation Language Models, a class of conditional LMs on knowledge graphs which models text as a latent sequence of spans matching related entities in the KG. The generative framework allows the model to not only outperform previous work, but also score spans with their posterior relation probability, which can be used for downstream tasks.

Acknowledgements

This research was supported in part by Funai Foundation for Information Technology and Amazon. The authors also thank the reviewers and NeuLab members for their helpful feedback, and Qian Wang for the figure design.

References

Ahn, S.; Choi, H.; Pärnamaa, T.; and Bengio, Y. 2016. A neural knowledge language model. *CoRR* arXiv:1608.00318.

Arthur, P.; Neubig, G.; and Nakamura, S. 2016. Incorporating discrete translation lexicons into neural machine translation. In *EMNLP*, 1557–1567.

Baevski, A., and Auli, M. 2019. Adaptive input representations for neural language modeling. In *ICLR*.

Baum, L. E.; Petrie, T.; Soules, G.; and Weiss, N. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics* 41(1):164–171.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *JMLR* 3(Feb):1137–1155.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *TACL* 5:135–146.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 1247–1250.

Buckman, J., and Neubig, G. 2018. Neural lattice language models. *TACL* 6:529–541.

Ceccarelli, D.; Lucchese, C.; Orlando, S.; Perego, R.; and Trani, S. 2013. Learning relatedness measures for entity linking. In *CIKM*, 139–148.

Chen, D. L., and Mooney, R. J. 2008. Learning to sportscast: A test of grounded language acquisition. In *ICML*, 128–135.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.; and Salakhutdinov, R. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL*, 2978–2988.

Dror, R.; Baumer, G.; Shlomov, S.; and Reichart, R. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *ACL*, 1383–1392.

Felice, M.; Yuan, Z.; Andersen, Ø. E.; Yannakoudakis, H.; and Kochmar, E. 2014. Grammatical error correction using hybrid systems and type filtering. In *CoNLL*, 15–24.

Ganea, O.-E., and Hofmann, T. 2017. Deep joint entity disambiguation with local neural attention. In *EMNLP*, 2619–2629.

Goldberg, E.; Driedger, N.; and Kittredge, R. I. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert* 9(2):45–53.

Grave, É.; Joulin, A.; Cissé, M.; Grangier, D.; and Jégou, H. 2017. Efficient softmax approximation for GPUs. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, 1302–1310.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*, 1631–1640.

Han, X.; Cao, S.; Lv, X.; Lin, Y.; Liu, Z.; Sun, M.; and Li, J. 2018. OpenKE: An open toolkit for knowledge embedding. In *EMNLP*, 139–144.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Ji, Y.; Tan, C.; Martschat, S.; Choi, Y.; and Smith, N. A. 2017. Dynamic entity representations in neural language models. In *EMNLP*, 1830–1839.

Kiddon, C.; Zettlemoyer, L.; and Choi, Y. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*, 329–339.

Konstas, I., and Lapata, M. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research* 48:305–346.

Lebret, R.; Grangier, D.; and Auli, M. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, 1203–1213.

Ling, W.; Blunsom, P.; Grefenstette, E.; Hermann, K. M.; Kočiský, T.; Wang, F.; and Senior, A. 2016. Latent predictor networks for code generation. In *ACL*, 599–609.

Liu, T.; Wang, K.; Sha, L.; Chang, B.; and Sui, Z. 2018. Table-to-text generation by structure-aware seq2seq learning. *AAAI*.

Logan, R.; Liu, N. F.; Peters, M. E.; Gardner, M.; and Singh, S. 2019. Barack's wife Hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*, 5962–5971.

Luong, M.-T., and Manning, C. D. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *ACL*, 1054–1063.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2017. Pointer sentinel mixture models. In *ICLR*.

Merity, S.; Keskar, N. S.; and Socher, R. 2017. Regularizing and optimizing LSTM language models. *CoRR* arXiv:1708.02182.

Miao, Y., and Blunsom, P. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *EMNLP*, 319–328.

Mikolov, T.; Karafiát, M.; Burget, L.; Černockỳ, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *ACL*, 1003–1011.

Neubig, G., and Dyer, C. 2016. Generalizing and hybridizing count-based and neural language models. In *EMNLP*, 1163–1172.

Parvez, M. R.; Chakraborty, S.; Ray, B.; and Chang, K.-W. 2018. Building language models for text with named entities. In *ACL*, 2373–2383.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *NAACL*, 2227–2237.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *Preprint*.

Reiter, E.; Sripada, S.; Hunter, J.; Yu, J.; and Davy, I. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence* 167(1-2):137–169.

Salton, G., and McGill, M. J. 1986. *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc.

Sundermeyer, M.; Schlüter, R.; and Ney, H. 2012. LSTM neural networks for language modeling. In *INTERSPEECH*.

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

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

Wang, Q.; Pan, X.; Huang, L.; Zhang, B.; Jiang, Z.; Ji, H.; and Knight, K. 2018. Describing a knowledge base. In *INLG*, 10–21.

Wiseman, S.; Shieber, S.; and Rush, A. 2018. Learning neural templates for text generation. In *EMNLP*, 3174–3187.

Zipf, G. K. 1949. *Human behavior and the principle of least effort: An introduction to human eoclogy*. Addison-Wesley Press.