A Neural Knowledge Language Model

Sungjin Ahn [1] **Heeyoul Choi** [2] **Tanel Pärnamaa** [3] **Yoshua Bengio** [1 4]

Abstract

Current language models have significant limitation in the ability to encode and decode factual knowledge. This is mainly because they acquire such knowledge from statistical co-occurrences although most of the knowledge words are rarely observed. In this paper, we propose a Neural Knowledge Language Model (NKLM) which combines symbolic knowledge provided by the knowledge graph with the RNN language model. By predicting whether the word to generate has an underlying fact or not, the model can generate such knowledge-related words by copying from the description of the predicted fact. In experiments, we show that the NKLM significantly improves the performance while generating a much smaller number of unknown words.

1. Introduction

> *Kanye West, a famous <unknown> and the husband of <unknown>, released his latest album <unknown> in <unknown>.*

A core purpose of language is to communicate knowledge. For human-level language understanding, it is thus of primary importance for a language model to take advantage of knowledge. Traditional language models are good at capturing statistical co-occurrences of entities as long as they are observed frequently in the corpus (e.g., words like verbs, pronouns, and prepositions). However, they are in general limited in their ability in dealing with factual knowledge because these are usually represented by named entities such as person names, place names, years, etc. (as shown in the above example sentence of Kanye West.)

Traditional language models have demonstrated to some extent the ability to encode and decode fatual knowledge (Vinyals & Le, 2015; Serban et al., 2015) when trained with a very large corpus. However, we claim that simply feeding a larger corpus into a bigger model hardly results in a good knowledge language model.

The primary reason for this is the difficulty in learning good representations for rare and unknown words. This is a significant problem because these words are of our primary interest in knowledge-related applications such as question answering (Iyyer et al., 2014; Weston et al., 2016; Bordes et al., 2015) and dialogue modeling (Vinyals & Le, 2015; Serban et al., 2015). Specifically, in the recurrent neural network language model (RNNLM) (Mikolov et al., 2010), the computational complexity is linearly dependent on the number of vocabulary words. Thus, including all words of a language is computationally prohibitive. Even if we can include a very large number of words in the vocabulary, according to Zipf's law, a large portion of the words will still be rarely observed in the corpus.

The fact that languages and knowledge can change over time also makes it difficult to simply rely on a large corpus. Media produce an endless stream of new knowledge every day (e.g., the results of baseball games played yesterday) that is even changing over time. Furthermore, a good language model should exercise some level of reasoning. For example, it may be possible to observe many occurrences of Barack Obama's year of birth and thus able to predict it in a correlated context. However, one would not expect current language models to predict, with a proper reasoning, the blank in "Barack Obama's age is ____" even if it is only a simple reformulation of the knowledge on the year of birth[1].

In this paper, we propose a Neural Knowledge Language Model (NKLM) as a step towards addressing the limitations of traditional language modeling when it comes to exploiting factual knowledge. In particular, we incorporate symbolic knowledge provided by the knowledge graph (Nickel et al., 2015) into the RNNLM. This connection makes sense particularly by observing that facts in knowledge graphs come along with textual representations which are mostly about rare words in text corpora.

In NKLM, we assume that each word generation is either

[1]Université de Montréal, Canada [2]Handong Global University, South Korea [3]Work done during internship at the Université de Montréal, Canada [4]CIFAR Senior Fellow. Correspondence to: Sungjin Ahn <sjn.ahn@gmail.com>.

[1]We do not investigate the reasoning ability in this paper but highlight this example because the explicit representation of facts would help to handle such examples.

based on a fact or not. Thus, at each time step, before generating a word, we predict whether the word to generate has an underlying fact or not. As a result, our model provides predictions over facts in addition to predictions over words. Hence, the previous context information on both facts and words flow through an RNN and provide a richer context. The NKLM has two ways to generate a word. One option is to generate a "vocabulary word" using the vocabulary softmax as is in the RNNLM. The other option is to generate a "knowledge word" by predicting the position of a word within the textual representation of the predicted fact. This makes it possible to generate words which are not in the predefined vocabulary and consequently resolves the rare and unknown word problem. The NKLM can also immediately adapt to adding or modifying knowledge because the model learns to predict facts, which can easily be modified without having to retrain the model.

The contributions of the paper are:

- To propose the NKLM model to resolve limitations of traditional language models in dealing with factual knowledge by using the knowledge graph.
- To develop a new dataset called *WikiFact* which can be used in knowledge-related language models by providing text aligned with facts.
- To show that the proposed model significantly improves the performance and can generate named entities which in traditional models were treated as unknown words.
- To propose new evaluation metrics that resolve the problem of the traditional perplexity metric in dealing with unknown words.

2. Related Work

There have been remarkable recent advances in language modeling research based on neural networks (Bengio et al., 2003; Mikolov et al., 2010). In particular, the RNNLMs are interesting for their ability to take advantage of longer-term temporal dependencies without a strong conditional independence assumption. It is especially noteworthy that the RNNLM using the Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) has recently advanced to the level of outperforming carefully-tuned traditional n-gram based language models (Jozefowicz et al., 2016).

There have been many efforts to speed up the language models so that they can cover a larger vocabulary. These methods approximate the softmax output using hierarchical softmax (Morin & Bengio, 2005; Mnih & Hinton, 2009), importance sampling (Jean et al., 2015), noise contrastive estimation (Mnih & Teh, 2012), etc. Although helpful to mitigate the computational problem, these approaches still suffer from the rare or unknown words problem.

To help deal with the rare/unknown word problem, the pointer networks (Vinyals et al., 2015) have been adopted to implement the copy mechanism (Gulcehre et al., 2016; Gu et al., 2016) and applied to machine translation and text summarization. With this approach, the (unknown) word to copy from the context sentence is inferred from neighboring words. Similarly, Merity et al. (2016) proposed to copy from the context sentences and Lebret et al. (2016) from Wikipedia infobox. However, because in our case the context can be very short and often contains no known relevant words (e.g., person names), we cannot use the existing approach directly.

Our knowledge memory is also related to the recent literature on neural networks with external memory (Bahdanau et al., 2014; Weston et al., 2015; Graves et al., 2014). In Weston et al. (2015), given simple sentences as facts which are stored in the external memory, the question answering task is studied. In fact, the tasks that the knowledge-based language model aims to solve (i.e., predict the next word) can be considered as a fill-in-the-blank type of question answering. The idea of jointly using Wikipedia and knowledge graphs has also been used in the context of enriching word embedding (Celikyilmaz et al., 2015; Long et al., 2016).

Context-dependent (or topic-based) language models have been studied to better capture long-term dependencies, by learning some context representation from the history. (Gildea & Hofmann, 1999) modeled the topic as a latent variable and proposed an EM-based approach. In (Mikolov & Zweig, 2012), the topic features are learned by latent Dirichlet allocation (LDA) (Blei et al., 2003).

3. Model

3.1. Preliminary

A *topic* is associated to *topic knowledge* and *topic description*. Topic knowledge $\mathcal{F}$ is a set of facts $\{a^1, a^2, \ldots, a^{|\mathcal{F}|}\}$ on the topic and topic description W is a sequence of words $(w_1, w_2, \ldots, w_{|W|})$ describing the topic. We can obtain the topic knowledge from a knowledge graph such as Freebase and the topic description from Wikipedia. In the corpus, we are given pairs of topic knowledge and topic description for K topics, i.e., $\{(\mathcal{F}_k, W_k)\}_{k=1}^{K}$. In the following, we omit index k when we indicate an arbitrary topic.

A fact is represented as a triple of *subject*, *relationship*, and *object* which is associated with a textual representation, e.g., (*Barack Obama, Married-To, Michelle Obama*). Note that all facts in a topic knowledge have the same subject entity which is the topic entity itself.

We define *knowledge words* $\mathcal{O}_a$ of a fact a as a sequence of words $(o_1^a, o_2^a, \ldots, o_N^a)$ from which we can copy a word to generate output. We also maintain a global vocabulary $\mathcal{V}$ containing frequent words. Because the words describing relationships (e.g., "married to") are common and thus can be generated via the vocabulary $\mathcal{V}$ not via copy, we limit the knowledge words of a fact to be the words for the object entity (e.g., $\mathcal{O}_a$ = (o_1^a="Michelle", o_2^a="Obama"). In addition, to make it possible to access the subject words from the knowledge words, we add a special fact, (Topic, Topic_Itself, Topic), to all topic knowledge.

We train the model in a supervised way with labels on facts and words. This requires aligning words in the topic description with their corresponding facts in the topic knowledge. Specifically, given $\mathcal{F}$ and W for a topic, we perform simple string matching between the words in W and all the knowledge words $\mathcal{O}_{\mathcal{F}} = \cup_{a\in\mathcal{F}}\mathcal{O}_a$ in such a way to associate fact a to word w if w appears in knowledge words $\mathcal{O}_a$. As a result, from $\mathcal{F}$ and W, we construct a sequence of augmented observations $Y = \{y_t = (w_t, a_t, z_t)\}_{t=1:|W|}$. Here, z_t is a binary variable indicating whether w_t is observed in the knowledge words or not:

$$z_t = \begin{cases} 1, & \text{if } w_t \in \mathcal{O}_{a_t}, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

In addition, because not all words are associated to a fact (e.g., words like, *is, a, the, have*), we introduce a special fact type, called Not-a-Fact (NaF), and to which assign such words. The following is an example of an augmented observation induced from a topic description and knowledge.

Example. Given a topic on Fred Rogers with topic description

W = "Rogers was born in Latrobe, Pennsylvania in 1928"

and topic knowledge $\mathcal{F} = \{a^{42}, a^{83}, a^0\}$ where

a^{42} = *(Fred_Rogers, Place_of_Birth, Latrobe_Pennsylvania)*
a^{83} = *(Fred_Rogers, Year_of_Birth, 1928)*
a^0 = *(Fred_Rogers, Topic_Itself, Fred_Rogers),*

the augmented observation Y is

Y = {(w="Rogers", a=0, z=1), ("was", NaF, 0), ("born", NaF, 0), ("in", NaF, 0), ("Latrobe", 42, 1), ("Pennsylvania", 42, 1), ("in", NaF, 0), ("1928", 83, 1)}.

During inference and training of a topic, we assume that the topic knowledge $\mathcal{F}$ is loaded in the *knowledge memory* in a form of a matrix $\mathbf{F} \in \mathbb{R}^{D_a \times |\mathcal{F}|}$ where the i-th column is a fact embedding $\mathbf{a}^i \in \mathbb{R}^{D_a}$. The fact embedding is the concatenation of subject, relationship, and object embeddings. We obtain these entity embeddings from a preliminary run of a knowledge graph embedding method such as TransE (Bordes et al., 2013). Note that we fix the fact embedding during the training. Thus, there is no drift of fact embeddings after training and thus the model can deal with new facts at test time; we learn the embedding of the Topic_Itself.

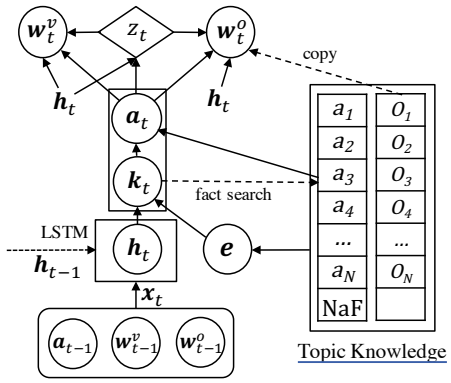

Figure 1. The NKLM model. The input consisting of a word (either $\mathbf{w}_{t-1}^o$ or $\mathbf{w}_{t-1}^v$) and a fact ($\mathbf{a}_{t-1}$) goes into LSTM. The LSTM's output $\mathbf{h}_t$ together with the knowledge context $\mathbf{e}$ generates the fact key $\mathbf{k}_t$. Using the fact key, the fact embedding $\mathbf{a}_t$ is retrieved from the topic knowledge memory. Using $\mathbf{a}_t$ and $\mathbf{h}_t$, the word generation source z_t is determined, which in turn determines the next word generation source $\mathbf{w}_t^v$ or $\mathbf{w}_t^o$. The copied word $\mathbf{w}_t^o$ is a symbol taken from the fact description $\mathcal{O}_{a_t}$.

For notation, to denote the vector representation of an object of our interest, we use bold lowercase. For example, the embedding of a word w is represented by $\mathbf{w} = \mathbf{W}[w]$ where $\mathbf{W}^{D_w \times |\mathcal{V}|}$ is the word embedding matrix, and $\mathbf{W}[w]$ denotes the w-th column of $\mathbf{W}$.

3.2. Inference

At each time step, the NKLM performs the following four sub-steps:

1. Using both the word and fact predictions of the previous time step, make an input to the current time step and update the LSTM controller.

2. Given the output of the LSTM controller, predict a fact and extract its corresponding embedding.

3. With the extracted fact embedding and the state of the LSTM controller, make a binary decision to determine the source of word generation.

4. According to the chosen source, generate a word either from the global vocabulary or by copying a word from the knowledge words of the selected fact.

A model diagram is depicted in Fig. 1. In the following, we describe these steps in more detail.

3.2.1. Input Representation and LSTM Controller

As shown in Fig. 1, the input at time step t is the concatenation of three embedding vectors corresponding to a fact a_{t-1}, a (global) vocabulary word $w^v_{t-1} \in \mathcal{V}$, and a knowledge word $w^o_{t-1} \in \mathcal{O}_{a_{t-1}}$, respectively. However, because the predicted word comes at a time step only either from the vocabulary or by copying from the knowledge words, i.e., $w_{t-1} \in \{w^v_{t-1}, w^o_{t-1}\}$, we set either $\mathbf{w}^v_{t-1}$ or $\mathbf{w}^o_{t-1}$ to a zero vector when it is not the generation source at the previous step. The resulting input representation $\mathbf{x}_t = f_{\text{concat}}(\mathbf{a}_{t-1}, \mathbf{w}^v_{t-1}, \mathbf{w}^o_{t-1})$ is then fed into the LSTM controller, and obtain the output states $\mathbf{h}_t = f_{\text{LSTM}}(\mathbf{x}_t, \mathbf{h}_{t-1})$.

3.2.2. Fact Extraction

We then predict a relevant fact a_t on which the word w_t will be based. Predicting a fact is done in two steps.

First, a fact-key $\mathbf{k}_{\text{fact}} \in \mathbb{R}^{D_a}$ is generated by a function $f_{\text{factkey}}(\mathbf{h}_t, \mathbf{e}_k)$ which is in our experiments a multilayer perceptron (MLP) with one hidden layer of ReLU nonlinearity and linear outputs. Here, $\mathbf{e}_k \in \mathbb{R}^{D_a}$ is the embedding of the topic knowledge which provides information about what facts are currently available in the topic knowledge. This would help the key generator adapt, without retraining, to changes in the topic knowledge such as removal or modification of some facts. Our experiments use mean-pooling to obtain $\mathbf{e}_k$, but one can also consider using a more sophisticated method such as the soft-attention mechanism (Bahdanau et al., 2014).

Then, using the generated fact-key $\mathbf{k}_{\text{fact}}$, we select a fact by key-value lookup across the knowledge memory $\mathbf{F}$ and then retrieve its embedding $\mathbf{a}_t$ as follows:

$$P(a|h_t) = \frac{\exp(\mathbf{k}^\top_{\text{fact}}\mathbf{F}[a])}{\sum_{a' \in \mathcal{F}} \exp(\mathbf{k}^\top_{\text{fact}}\mathbf{F}[a'])}, \quad (2)$$

$$a_t = \operatorname*{argmax}_{a \in \mathcal{F}} P(a|h_t), \quad (3)$$

$$\mathbf{a}_t = \mathbf{F}[a_t]. \quad (4)$$

3.2.3. Selecting Word Generation Source

Given the context $\mathbf{h}_t$ and the extracted fact $\mathbf{a}_t$, the model decides the source for the next word generation: either from the vocabulary $\mathcal{V}$ or from the knowledge words $\mathcal{O}_{a_t}$. We define the probability of selecting generation-by-copy as:

$$\hat{z}_t = p(z_t|h_t, a_t) = \text{sigmoid}(f_{\text{copy}}(\mathbf{h}_t, \mathbf{a}_t)). \quad (5)$$

Here, f_{copy} is an MLP with one ReLU hidden layer and a single linear output unit.

Word w_t is generated from the source indicated by $\hat{z}_t$ as follows:

$$w_t = \begin{cases} w^v_t \in \mathcal{V}, & \text{if } \hat{z}_t < 0.5, \\ w^o_t \in \mathcal{O}_{a_t}, & \text{otherwise.} \end{cases}$$

Algorithm 1 NKLM inference at time step t

```
## Make input
x_t = f_concat(a_{t-1}, w^v_{t-1}, w^o_{t-1})
## Update LSTM controller
h_t = LSTM(h_{t-1}, x_t)
## Fact prediction and extract embedding
a_t = argmax_{a∈F} P(a|h_t, e_k)
a_t = F[a_t]
## Decide word generation source
z_t = 𝕀[p(z_t|h_t, a_t) > 0.5]
if z_t == 0 then
    ## Word generation from vocabulary
    w_t = w^v_t = argmax_{w∈V} P(w|h_t, a_t)
    w^o_t = 0
else
    ## Word generation by copy
    n_t = argmax_{n=0:|O_{a_t}|-1} P(n|h_t, a_t)
    w_t = w^o_t = O_{a_t}[n_t]
    w^v_t = 0
end if
```

3.2.4. Word Generation

Generation from Vocabulary Softmax: For vocabulary word $w^v_t \in \mathcal{V}$, we follow the usual way of selecting a word using the softmax function:

$$P(w^v_t = w|h_t, a_t) = \frac{\exp(\mathbf{k}^\top_{\text{voca}}\mathbf{W}[w])}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{k}^\top_{\text{voca}}\mathbf{W}[w'])}, \quad (6)$$

where $\mathbf{k}_{\text{voca}} \in \mathbb{R}^{D_w}$ is obtained by $f_{\text{voca}}(\mathbf{h}_t, \mathbf{a}_t)$ which is an MLP with a ReLU hidden layer and D_w linear output units.

Generation by Copy from Knowledge Words: To copy a knowledge word $w^o_t \in \mathcal{O}_{a_t}$, we first predict the *position* of the word within the knowledge words and then copy the word on the predicted position. This copy-by-position allows us not to rely on the word embeddings by instead learning position embeddings.

One reason to use position prediction is that the traditional copy mechanism (Gulcehre et al., 2016; Gu et al., 2016) is difficult to apply to our context because the knowledge words usually consist of only unknown words and/or are short in length. Furthermore, it makes sense when considering the fact that we mostly need to copy the knowledge words in increasing order from the first word. For example, given that the first symbol o_1 = *"Michelle"* was used in the previous time step and prior to that other words such

# topics	# toks	# uniq toks	# facts	# entities	# relations	$\max_k \lvert\mathcal{F}_k\rvert$	$\text{avg}_k \lvert\mathcal{F}_k\rvert$	$\max_a \lvert O_a\rvert$	$\text{avg}_a \lvert\mathcal{O}_a\rvert$
10K	1.5M	78k	813k	560K	1.5K	1K	79	19	2.15

Table 1. Statistics of the WikiFacts-FilmActor-v0.1 dataset.

as *"President"* and *"US"* were also observed, the model can easily predict that it is time to select the second symbol, i.e., $o_2 =$ *"Obama"*.

More specifically, we first generate the position key $\mathbf{k}_{\text{pos}} \in \mathbb{R}^{D_o}$ by a function $f_{\text{poskey}}(\mathbf{h}_t, \mathbf{a}_t)$ which is again an MLP with one hidden layer and linear outputs whose dimension is the maximum number of positions, e.g., the maximum length of the knowledge words (e.g., $N^o_{\max} = \max_{a \in \bar{\mathcal{F}}} |O_a|$ where $\bar{\mathcal{F}} = \cup_k \mathcal{F}_k$). Then, the word to copy is chosen by

$$P(n|h_t, a_t) = \frac{\exp(\mathbf{k}_{\text{pos}}^\top \mathbf{P}[n])}{\sum_{n'} \exp(\mathbf{k}_{\text{pos}}^\top \mathbf{P}[n'])}, \tag{7}$$

$$n_t = \operatorname*{argmax}_{n=0:|\mathcal{O}_{a_t}|-1} P(n|h_t, a_t), \tag{8}$$

$$w^o_t = \mathcal{O}_{a_t}[n_t], \tag{9}$$

with position n' running from 0 to $|\mathcal{O}_{a_t}| - 1$. Here, $\mathbf{P}^{D_o \times N^o_{\max}}$ is the matrix of position embeddings of dimension D_o. Note that $N^o_{\max}$ is typically a much smaller number (e.g., 20 in our experiments) than the size of vocabulary, and thus the computation for copy is efficient. The position embedding matrix $\mathbf{P}$ is learned during training.

Although in our experiments we find that the simple position prediction performs well, we note that one could also consider a more advanced encoding such as one based on a convolutional network (Kim, 2014) to model the knowledge words.

To compute $p(w_t|w_{<t}, \mathcal{F})$, we first obtain $\{z_{<t}, a_{<t}\}$ from $\{w_{<t}\}$ and $\mathcal{F}$ using the augmentation procedure, and perform the above inference process with hard decisions taken about z_t and a_t based on the model's predictions. The inference procedure is summarized in Algorithm 1.

3.3. Learning

We perform supervised learning on the augmented observation Y, similarly to Reed & de Freitas (2016). That is, given word observations $\{Y_k\}_{k=1}^K$ and knowledge $\{\mathcal{F}_k\}_{k=1}^K$, our objective is to maximize the log-likelihood of the augmented observation w.r.t the model parameter θ,

$$\theta^* = \operatorname*{argmax}_\theta \sum_k \log P_\theta(Y_k|\mathcal{F}_k). \tag{10}$$

By the chain rule, we can decompose the probability of the observation Y_k as

$$\log P_\theta(Y_k|\mathcal{F}_k) = \sum_{t=1}^{|Y_k|} \log P_\theta(y^k_t|y^k_{1:t-1}, \mathcal{F}_k). \tag{11}$$

Then, after omitting $\mathcal{F}_k$ and k for simplicity, we can rewrite the single step conditional probability as

$$\begin{aligned} P_\theta(y_t|y_{1:t-1}) &= P_\theta(w_t, a_t, z_t|h_t) \\ &= P_\theta(w_t|a_t, z_t, h_t) P_\theta(a_t|h_t) P_\theta(z_t|h_t). \end{aligned} \tag{12}$$

We maximize the above objective using stochastic gradient optimization.

4. Evaluation

An obstacle in developing the proposed model is the lack of datasets where the text is aligned with facts at the word level. While the Penn Treebank (PTB) dataset (Marcus et al., 1993) has been frequently used in language modeling, as pointed by Merity et al. (2016), its limited vocabulary containing a relatively small amount of named entities makes it difficult to use them for knowledge-related tasks where rare words are of primary interest; we would have only a very small amount of words to be associated with facts. As other larger datasets such as in Chelba et al. (2013) also have problems in licensing or in the format of the dataset, we produce the *WikiFacts* dataset for evaluation of the proposed model and the baseline model. The dataset is freely available in `https://bitbucket.org/skaasj/wikifact_filmactor`.

4.1. The WikiFacts Dataset

In *WikiFacts*, we align Wikipedia descriptions with corresponding Freebase[2] facts. Because many Freebase topics provide a link to its corresponding topic in Wikipedia, we choose a set of topics for which both a Freebase entity and a Wikipedia description exist. In the experiments, we used a version called `WikiFacts-FilmActor-v0.1` where the domain is restricted to the */Film/Actor* in Freebase.

We used the summary part (first few paragraphs) of the Wikipedia page as the text to be modeled, but discarded topics for which the number of facts is too large (> 1000) or the Wikipedia description is too short (< 3 sentences). For the string matching, we also used synonyms and alias information provided by WordNet (Miller, 1995) and Freebase.

We augmented the fact set $\mathcal{F}$ with the *anchor* facts $\mathcal{A}$ whose relationship is all set to `UnknownRelation`. That is, observing that an anchor (a word under a hyperlink) in a Wikipedia description has a corresponding Freebase entity as well as being semantically closely related to the topic

[2]Freebase has migrated to Wikidata. `www.wikidata.org`

	Validation			Test			
Model	PPL	UPP	UPP-f	PPL	UPP	UPP-f	# UNK
RNNLM	39.4	97.9	56.8	39.4	107.0	58.4	23247
NKLM	**27.5**	**45.4**	**33.5**	**28.0**	**48.7**	**34.6**	**12523**
no-copy	38.4	93.5	54.9	38.3	102.1	56.4	29756
no-fact-no-copy	40.5	98.8	58.0	40.3	107.4	59.3	32671
no-TransE	48.9	80.7	59.6	49.3	85.8	61.0	13903

Table 2. **We compare four different versions of the NKLM to the RNNLM on three different perplexity metrics.** We used 10K vocabulary. In **no-copy**, we disabled the generation-by-copy functionality, and in **no-fact-no-copy**, using topic knowledge is also additionally disabled by setting all facts as NaF. Thus, **no-fact-no-copy** is very similar to RNNLM. In **no-TransE**, we used random vectors instead of the TransE embeddings to initialize the knowledge graph entities. As shown, the NKLM shows best performance in all cases. The **no-fact-no-copy** performs similar to the RNNLM as expected (slightly worse partly because it has a smaller number of model parameters than that of the RNNLM). As expected, **no-copy** performs better than **no-fact-no-copy** by using additional information from the fact embedding, but without the copy mechanism. In the comparison of the NKLM and **no-copy**, we can see the significant gain of using the copy mechanism to predict named entities. In the last column, we can also see that, with the copy mechanism, the number of predicting unknown decreases significantly. Lastly, we can see that the TransE embedding is important.

	Validation			Test			
Model	PPL	UPP	UPP-f	PPL	UPP	UPP-f	# UNK
NKLM_5k	**22.8**	**48.5**	**30.7**	**23.2**	**52.0**	**31.7**	**19557**
RNNLM_5k	27.4	108.5	47.6	27.5	118.3	48.9	34994
NKLM_10k	**27.5**	**45.4**	**33.5**	**28.0**	**48.7**	**34.6**	**12523**
RNNLM_10k	39.4	97.9	56.8	39.4	107.0	58.4	23247
NKLM_20k	**33.4**	**45.9**	**37.9**	**34.7**	**49.2**	**39.7**	**9677**
RNNLM_20k	57.9	99.5	72.1	59.3	108.3	75.5	13773
NKLM_40k	**41.4**	**49.0**	**44.4**	**43.6**	**52.7**	**47.1**	**5809**
RNNLM_40k	82.4	107.9	92.3	86.4	116.9	97.9	9009

Table 3. **The NKLM and the RNNLM are compared for vocabularies of four different sizes [5K, 10K, 20K, 40K]**. As shown, in all cases the NKLM significantly outperforms the RNNLM. Interestingly, for the standard perplexity (PPL), the gap between the two models increases as the vocabulary size increases while for UPP the gap stays at a similar level regardless of the vocabulary size. This tells us that the standard perplexity is significantly affected by the UNK predictions, because with UPP the contribution of UNK predictions to the total perplexity is very small. Also, from the UPP value for the RNNLM, we can see that it initially improves when vocabulary size is increased as it can cover more words, but decreases back when the vocabulary size is largest (40K) because the rare words are added last to the vocabulary.

in which the anchor is found, we make a synthetic fact of the form (*Topic*, `UnknownRelation`, *Anchor*). This potentially compensates for some missing facts in Freebase. Because we extract the anchor facts from the full Wikipedia page and they all share the same relation, it is more challenging for the model to use these anchor facts than using the Freebase facts.

As a result, for each word w in the description, we obtain a tuple (w, z, a, n, k). Here, w is word id, z the copy indicator, a fact id, n the position to copy from $\mathcal{O}_a$ if $z = 1$, and k topic id. We provide a summary of the dataset statistics in Table 1.

4.2. Experiments

4.2.1. Setup

We split the dataset into 80/10/10 for train, validation, and test. As a baseline model, we use the RNNLM. For both the NKLM and the RNNLM, two-layer LSTMs with dropout regularization (Zaremba et al., 2014) are used. We tested models with different numbers of LSTM hidden units [200, 500, 1000], and report results from the 1000 hidden-unit model. For the NKLM, we set the symbol embedding dimension to 40 and word embedding dimension to 400. Under this setting, the number of parameters in the NKLM is slightly smaller than that of the RNNLM.

We used 100-dimension TransE embeddings for Freebase entities and relations, and concatenate the relation and object embeddings to obtain fact embeddings. We averaged all fact embeddings in $\mathcal{F}_k$ to obtain the topic knowledge em-

Warm-up	Louise Allbritton (3 july <unk>february 1979) was
RNNLM	a <unk><unk>who was born in <unk>, <unk>, <unk>, <unk>, <unk>, <unk>, <unk>
NKLM	an english [Actor]. he was born in [Oklahoma] , and died in [Oklahoma]. he was married to [Charles] [Collingwood]
Warm-up	Issa Serge Coelo (born 1967) is a <unk>
RNNLM	actor . he is best known for his role as <unk><unk>in the television series <unk>. he also
NKLM	[Film] director . he is best known for his role as the <unk><unk>in the film [Un] [taxi] [pour] [Aouzou]
Warm-up	Adam wade Gontier is a canadian Musician and Songwriter .
RNNLM	she is best known for her role as <unk><unk>on the television series <unk>. she has also appeared
NKLM	he is best known for his work with the band [Three] [Days] [Grace] . he is the founder of the
Warm-up	Rory Calhoun (august 8 , 1922 april 28
RNNLM	, 2010) was a <unk>actress . she was born in <unk>, <unk>, <unk>. she was
NKLM	, 2008) was an american [Actor] . he was born in [Los] [Angeles] california . he was born in

Table 4. **Sampled Descriptions**. Given the warm-up phrases, we generate samples from the NKLM and the RNNLM. We denote the copied knowledge words by [word] and the UNK words by <unk>. Overall, the RNNLM generates many UNKs (we used 10K vocabulary) while the NKLM is capable to generate named entities even if the model has not seen some of the words at all during training. In the first case, we found that the generated symbols (words in []) conform to the facts of the topic (Louise Allbritton) except that she actually died in Mexico, not in Oklahoma. (We found that the place_of_death fact was missing.) While she is an actress, the model generated a word [Actor]. This is because in Freebase, there exists only /profession/actor but no /profession/actress. It is also noteworthy that the NKLM fails to use the gender information provided by facts; the NKLM uses "he" instead of "she" although the fact /gender/female is available. From this, we see that if a fact is not detected (i.e., NaF), the statistical co-occurrence governs the information flow. Similarly, in other samples, the NKLM generates movie titles (Un Taxi Pour Aouzou), band name (Three Days Grace), and place of birth (Los Angeles). In addition, to see the NKLM's ability to adapt to knowledge updates without retraining, we changed the fact /place_of_birth/Oklahoma to /place_of_birth/Chicago and found that the NKLM replaces "Oklahoma" by "Chicago" while keeping other words the same.

bedding $\mathbf{e}_k$. We unrolled the LSTMs for 30 steps and used minibatch size 20. We trained the models using stochastic gradient ascent with gradient clipping range [-5,5]. The initial learning rate was set to 0.5 for the NKLM and 1.5 for the RNNLM, and decayed after every epoch by a factor of 0.98. We trained for 50 epochs and report the results chosen by the best validation set results.

4.2.2. The Unknown Penalized Perplexity

The perplexity $\exp(-\frac{1}{N}\sum_{i=1}^{N}\log p(w_i))$ is the standard performance metric for language modeling. This, however, has a problem in evaluating language models for a corpus containing many named entities: *a model can get good perplexity by accurately predicting UNK words as the UNK class*. As an extreme example, when all words in a sentence are unknown words, a model predicting everything as UNK will get a good perplexity. Considering that unknown words provide virtually no useful information, this is clearly a problem in tasks where named entities are important such as question answering, dialogue modeling, and knowledge language modeling.

To this end, we propose a new evaluation metric, called the Unknown-Penalized Perplexity (UPP), and evaluate the models on this metric as well as the standard perplexity (PPL). Because the actual word underlying the UNK should be one of the out-of-vocabulary (OOV) words, in UPP we penalize the likelihood of unknown words as follows:

$$P_{\text{UPP}}(w_{\text{unk}}) = \frac{P(w_{\text{unk}})}{|\mathcal{V}_{\text{total}} \setminus \mathcal{V}_{\text{voca}}|}.$$

Here, $\mathcal{V}_{\text{total}}$ is a set of all unique words in the corpus, and $\mathcal{V}_{\text{voca}} \subset \mathcal{V}_{\text{total}}$ is the global vocabulary used for word generation. In other words, in UPP we assume that the OOV set is equal to $\mathcal{V}_{\text{total}} \setminus \mathcal{V}_{\text{voca}}$ and thus assign a uniform probability to OOV words. In another version, UPP-fact, we consider the fact that the RNNLM can also use the knowledge given to the NKLM to some extent, but with limited capability (because the model is not designed for it). For this, we assume that the OOV set is equal to the total knowledge words of a topic k, i.e.,

$$P_{\text{UPP-fact}}(w_{\text{unk}}) = \frac{P(w_{\text{unk}})}{|\mathcal{O}_{\mathcal{F}_k}|},$$

where $\mathcal{O}_{\mathcal{F}_k} = \cup_{a\in\mathcal{F}_k}\mathcal{O}_a$. In other words, by using UPP-fact, we assume that, for an unknown word, the RNNLM can pick one of the knowledge words with uniform probability.

4.2.3. Observations from experiment results

We describe the detail results and analysis on the experiments in detail in the captions of Table 2, 3, and 4. Our observations from the experiment results are as follows.

- The NKLM outperforms the RNNLM in all three perplexity measures.

- The copy mechanism is the key of the significant performance improvement. Without the copy mechanism, the NKLM still performs better than the RNNLM due to its usage of the fact information, but the improvement is not so significant.
- The NKLM results in a much smaller number of UNKs (roughly, a half of the RNNLM).
- When no knowledge is available, the NKLM performs as well as the RNNLM.
- Knowledge graph embedding using TransE is an efficient way of representing facts in our model.
- The NKLM generates named entities in the provided facts whereas the RNNLM generates many more UNKs.
- The NKLM shows its ability to adapt immediately to the change of the knowledge.
- The standard perplexity is significantly affected by the prediction accuracy on the unknown words. Thus, one need carefully consider when using it as a metric for knowledge-related language models.

5. Conclusion

In this paper, we presented a novel Neural Knowledge Language Model (NKLM) that brings the symbolic knowledge from a knowledge graph into the expressive power of RNN language models. The NKLM significantly outperforms the RNNLM in terms of perplexity and generates named entities which are not observed during training, as well as immediately adapting to changes in knowledge. We believe that the WikiFact dataset introduced in this paper, can be useful in other knowledge-related language tasks as well. In addition, the Unknown-Penalized Perplexity introduced in order to resolve the limitation of the standard perplexity, can also be useful in evaluating other language tasks.

The task that we investigated in this paper is limited in the sense that we assume that the true topic of a given description is known. Relaxing this assumption by making the model search for a proper topic on-the-fly will make the model more practical and scalable. We believe that there are many more open research challenges related to the knowledge language models.

Acknowledgments

The authors would like to thank Alberto García-Durán, Caglar Gulcehre, Chinnadhurai Sankar, Iulian Serban, Sarath Chandar, and Peter Clark for helpful feedbacks and discussions as well as the developers of Theano (Bastien et al., 2012), NSERC, CIFAR, Facebook, Google, IBM, Microsoft, Samsung, and Canada Research Chairs for funding, and Compute Canada for computing resources.

References

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Bergstra, James, Goodfellow, Ian, Bergeron, Arnaud, Bouchard, Nicolas, Warde-Farley, David, and Bengio, Yoshua. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

Bengio, Yoshua, Ducharme, Réjean, Vincent, Pascal, and Jauvin, Christian. A neural probabilistic language model. In *Journal of Machine Learning Research*, 2003.

Blei, David M, Ng, Andrew Y, and Jordan, Michael I. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Bordes, Antoine, Usunier, Nicolas, Garcia-Duran, Alberto, Weston, Jason, and Yakhnenko, Oksana. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pp. 2787–2795, 2013.

Bordes, Antoine, Usunier, Nicolas, Chopra, Sumit, and Weston, Jason. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

Celikyilmaz, Asli, Hakkani-Tur, Dilek, Pasupat, Panupong, and Sarikaya, Ruhi. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. In *2015 AAAI Spring Symposium Series*, 2015.

Chelba, Ciprian, Mikolov, Tomas, Schuster, Mike, Ge, Qi, Brants, Thorsten, Koehn, Phillipp, and Robinson, Tony. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

Gildea, Daniel and Hofmann, Thomas. Topic-based language models using em. *EuroSpeech 1999*, 1999.

Graves, Alex, Wayne, Greg, and Danihelka, Ivo. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

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

Gulcehre, Caglar, Ahn, Sungjin, Nallapati, Ramesh, Zhou, Bowen, and Bengio, Yoshua. Pointing the unknown words. *ACL 2016*, 2016.

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

Iyyer, Mohit, Boyd-Graber, Jordan L, Claudino, Leonardo Max Batista, Socher, Richard, and Daumé III, Hal. A neural network for factoid question answering over paragraphs. In *EMNLP 2014*, pp. 633–644, 2014.

Jean, Sebastien, Cho, Kyunghyun, Memisevic, Roland, and Bengio, Yoshua. On using very large target vocabulary for neural machine translation. *ACL 2015*, 2015.

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

Kim, Yoon. Convolutional neural networks for sentence classification. *EMNLP 2014*, 2014.

Lebret, Rémi, Grangier, David, and Auli, Michael. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*, 2016.

Long, Teng, Lowe, Ryan, Cheung, Jackie Chi Kit, and Precup, Doina. Leveraging lexical resources for learning entity embeddings in multi-relational data. 2016.

Marcus, Mitchell P, Marcinkiewicz, Mary Ann, and Santorini, Beatrice. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

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

Mikolov, Tomas and Zweig, Geoffrey. Context dependent recurrent neural network language model. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pp. 234–239. IEEE, 2012.

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

Miller, George A. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Mnih, Andriy and Hinton, Geoffrey E. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pp. 1081–1088, 2009.

Mnih, Andriy and Teh, Yee Whye. A fast and simple algorithm for training neural probabilistic language models. *ICML 2012*, 2012.

Morin, Frederic and Bengio, Yoshua. Hierarchical probabilistic neural network language model. *AISTATS 2005*, pp. 246, 2005.

Nickel, Maximilian, Murphy, Kevin, Tresp, Volker, and Gabrilovich, Evgeniy. A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. *arXiv preprint arXiv:1503.00759*, 2015.

Reed, Scott and de Freitas, Nando. Neural programmer-interpreters. *ICLR 2016*, 2016.

Serban, Iulian V, Sordoni, Alessandro, Bengio, Yoshua, Courville, Aaron, and Pineau, Joelle. Building end-to-end dialogue systems using generative hierarchical neural networks. *30th AAAI Conference on Artificial Intelligence*, 2015.

Vinyals, Oriol and Le, Quoc. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

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

Weston, Jason, Chopra, Sumit, and Bordes, Antoine. Memory networks. *ICLR 2015*, 2015.

Weston, Jason, Bordes, Antoine, Chopra, Sumit, and Mikolov, Tomas. Towards ai-complete question answering: A set of prerequisite toy tasks. *ICLR 2016*, 2016.

Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

APPENDIX: HEATMAPS

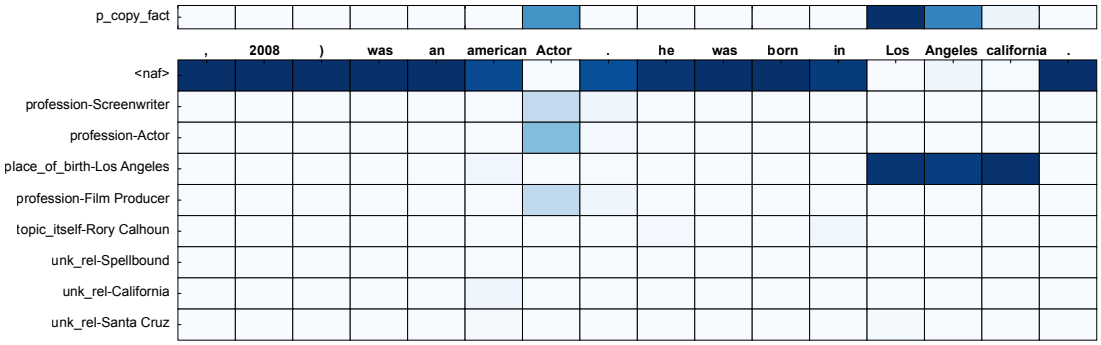

Figure 2. This is a heatmap of an example sentence generated by the NKLM having a warmup *"Rory Calhoun (august 8 , 1922 april 28"*. The first row shows the probability of selecting copy (Equation 5 in Section 3.1). The bottom heat map shows the state of the topic-memory at each time step (Equation 2 in Section 3.1). In particular, this topic has 8 facts and an additional <NaF> fact. For the first six time steps, the model retrieves <NaF>from the knowledge memory, copy-switch is off and the words are generated from the general vocabulary. For the next time step, the model gives higher probability to three different profession facts: "Screenwriter", "Actor" and "Film Producer." The fact "Actor" has the highest probability, copy-switch is higher than 0.5, and therefore "Actor" is copied as the next word. Moreover, we see that the model correctly retrieves the place of birth fact and outputs "Los Angeles." After that, the model still predicts the place of birth fact, but copy-switch decides that the next word should come from the general vocabulary, and outputs "California."

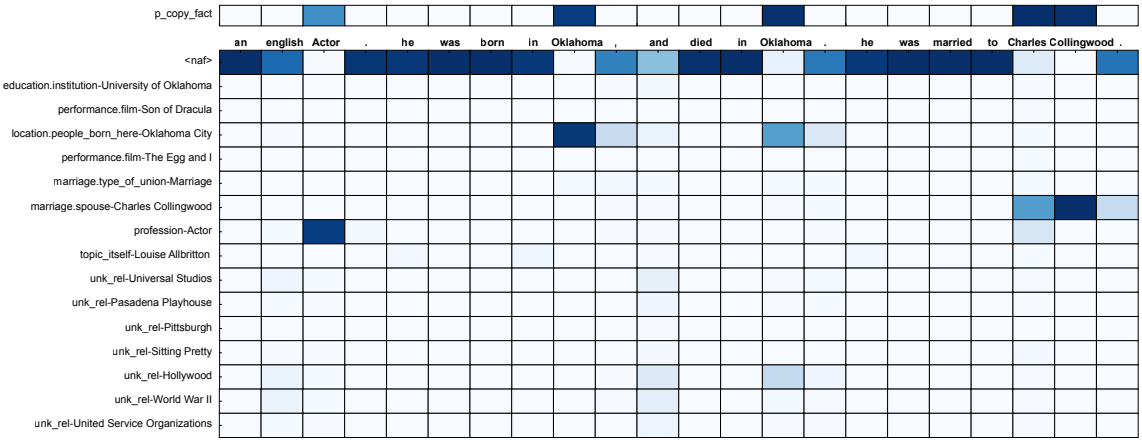

Figure 3. This is an example sentence generated by the NKLM having a warmup *"Louise Allbritton (3 july <unk>february 1979) was"*. We see that the model correctly retrieves and outputs the profession ("Actor"), place of birth ("Oklahoma"), and spouse ("Charles Collingwood") facts. However, the model makes a mistake by retrieving the place of birth fact in a place where the place of death fact is supposed to be used. This is probably because the place of death fact is missing in this topic memory and then the model searches for a fact about location, which is somewhat encoded in the place of birth fact. In addition, *Louise Allbritton* was a woman, but the model generates a male profession "Actor" and male pronoun "he". The "Actor" is generated because there is no "Actress" representation in Freebase.