# Building Language Models for Text with Named Entities

Md Rizwan Parvez, Saikat Chakraborty, Baishakhi Ray, Kai-Wei Chang

https://github.com/uclanlp/NamedEntityLanguageModel

## Abstract


❖ Language models often cannot predict entity names in text well due to their diverse vocabularies.

❖ We propose a language model for texts with entity names by leveraging the deterministic entity types.

- ***Type Model*:** predicts the type of the entity name
- ***Entity Composite Model*:** generates the conditional actual entity name given the entity type as prior

❖ We conduct experiments on two applications:

- Recipe generation and Code generation


## Motivation

❖ Generating entity names in text is very challenging (e.g., *olive oil, canola oil, grape oil*, etc.—all are different varieties of *oil).*

❖ Existing language models:

- fail to address this problem.
- focus on copying/matching the entity names from the reference corpus[1,2].
- are based on generative models of low performance.

## Methodology

Our model learns the probability distribution over the candidate words by decomposing it into two sub-problems:

❖ ***Type Model*** ($\theta_t$) :

- considers all of entities with the same type equal and represent them by their type information.
- reduces the vocab size to a great extent.
- predicts the type ***s(w)*** of each entity (***w***) more accurately.

❖ ***Entity Composite Model*** ($\theta_v$) :

- uses the entity type generated by the ***type model*** as a prior.
- calculates the conditional probability distribution of the actual entity names.

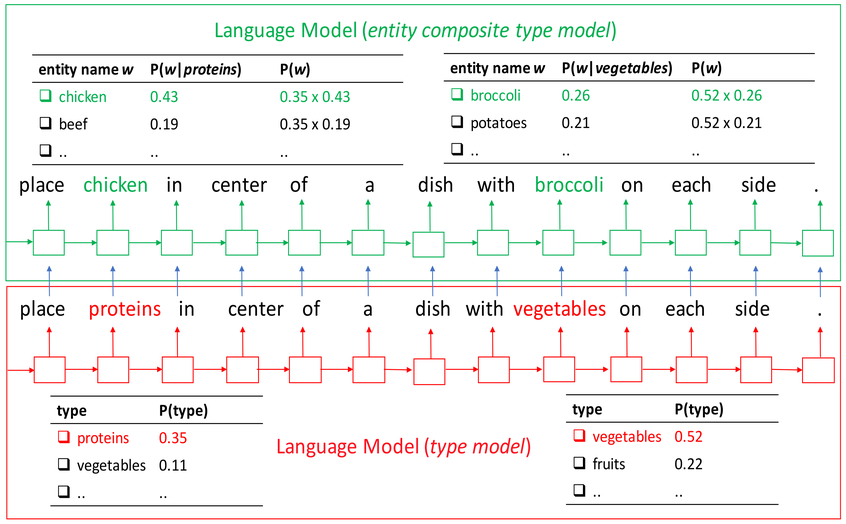


Figure 1. An illustration of our model Given a context, the ***type model*** (in bottom red block) generates the type of the words. Further, for that given context & type of each candidate generated by the ***type model***, the ***entity composite model*** (in upper green block) generates the conditional actual entity names.

❖ Given the context $\overline{w}$, type of the context $s(\overline{w})$, our goal is to model the probability of the next word ***w***, $P(w|\overline{w};\theta_t,\theta_v) \approx P(w, s(w)|\overline{w}, s(\overline{w});\theta_t,\theta_v)$. As a word can be either a named entity or not, we model it by the following two cases:

$$P(w|\overline{w}, s(w), s(\overline{w});\theta_v) \times \begin{cases} P(s(w)|s(\overline{w});\theta_t) & \text{if } w \text{ is an entity name} \\ 1 - \sum_{s \epsilon S} P(s(w) = s|s(\overline{w});\theta_t) & \text{otherwise} \end{cases} \quad (1)$$

where ***S*** is set of all entity types.

❖ We train $\theta_t$ and $\theta_v$ independently by any state-of-art language model.

❖ We do a joint inference according to Equ-1.

## Experimental Results

We create two benchmark datasets that contain many named entities.

❖ **Recipe Generation Corpus:**

- 95,786 cooking recipes[4]
- Manually categorized 8 super-ingredients (i.e., types); e.g., proteins, vegetables, fruits etc.

- Recipe generation performance:

| Model | corpus | Vocab | Perplexity |
|---|---|---|---|
| AWD_LSTM (state-of-art language model) | original | 52,742 | 20.23 |
| AWD_LSTM | type | 51, 675 | 17.62 |
| AWD_LSTM (entity type as an additional input) | original | 52,742 | 18.23 |
| Our model | original | 52,742 | **9.67** |

- Accuracy improvement while predicting the exact ingredient name:
  - Free-form (i.e., without any option)
  - From a set of options (MCQ)

| Entity name | Training frequency | #Blanks | Free-form AWD_LSTM | Free-form ours | MCQ AWD_LSTM | MCQ ours |
|---|---|---|---|---|---|---|
| Milk | 14,136 | 4001 | 26.94 | **59.34** | 80.83 | **94.90** |
| Salt | 33,306 | 9,888 | 37.12 | **62.47** | 89.29 | **95.75** |
| Apple | 7,205 | 720 | 1.94 | **30.28** | 37.65 | **89.86** |
| Bread | 11,673 | 3,074 | 32.43 | **52.64** | 78.85 | **94.53** |
| Tomato | 19,875 | 6,072 | 22.50 | **45.24** | 77.70 | **94.63** |

❖ **Code Generation Corpus:**

- 500 open source Android projects collected from GitHub
- Abstract Syntax Tree (AST) based entity type
- **22.06%** better in perplexity with respect to state-of-art code generation baseline SLP-Core[3]

## References


1. Gu, Jiatao, et al. "Incorporating copying mechanism in sequence-to-sequence learning." *arXiv preprint arXiv:1603.06393* (2016).
2. Wiseman, Sam, Stuart M. Shieber, and Alexander M. Rush. "Challenges in data-to-document generation." *arXiv preprint arXiv:1707.08052* (2017).
3. Hellendoorn, Vincent J., and Premkumar Devanbu. "Are deep neural networks the best choice for modeling source code?." *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 2017.
4. Collected from http://www.ffts.com/recipes.htm.


## Acknowledgement

This work was supported in part by National Science Foundation Grants IIS1760523, CCF-16-19123, CNS-16-18771 and an NVIDIA hardware grant.