Appendix of
Looking Beyond Label Noise: Shifted Label Distribution Matters in Distantly Supervised Relation Extraction

A Detailed Experiment Setup

For a fair and meaningful comparison, we use the same experimental setup in all experiments.

A.1 Model Details

We consider two popular classes of relation extraction methods here, *i.e.*, feature-based and neural models. For each relation mention, these models will first construct a representation $\mathbf{h}$, and then make predictions based on $\mathbf{h}$. [1]

$$p(y = r_i|\mathbf{h}) = \frac{\exp(\mathbf{r}_i^T\mathbf{h} + b_i)}{\sum_{r_j} \exp(\mathbf{r}_j^T\mathbf{h} + b_j)}$$

where $\mathbf{r}_i$ and b_i are the parameters corresponding to i-th relation type.

A.1.1 Feature-based model

We included three feature-based models, *i.e.*, CoType-RM (Ren et al., 2017), ReHession (Liu et al., 2017), and multi-class logistic regression. For each relation mention z, these methods would first extract a list of features, $\mathcal{F} = \{f_1, f_2, ..., f_m\}$. These features require additional resources like POS-taggers and brown clustering. Detailed description of these features are listed in Table 4.

CoType-RM is a variant of CoType (Ren et al., 2017), a unified learning framework to get both the feature embedding and the label embedding. It leverages a partial-label loss to handle the label noise, and uses cosine similarity to conduct inference. Here, we only use its relation extraction part.

ReHession (Liu et al., 2017) directly maps each feature to an embedding vector, treats their average as the relation mention representation $\mathbf{h}$, and uses a softmax to make predictions. This method was initially proposed for heterogeneous supervision, and is modified to fit our distantly supervised relation extraction task. Specifically, for a relation mention annotated with a set of relation $Y = \{r_{l_1}, \cdots, r_{l_m}\}$, it would first calculate a cross entropy as the loss function:

$$\mathcal{L} = -\sum_{r_i} q(y = r_i|Y, \mathbf{h}) \log p(y = r_i|\mathbf{h}) \quad (9)$$

where $p(.|\mathbf{h})$ is defined in Equation 1, and $q(.|Y, \mathbf{h})$ is used to encode supervision information in a self-adapted manner:

$$q(y = r_i|Y, \mathbf{h}) = \frac{\exp(\mathbf{r}_i^T\mathbf{h} + b_i) \cdot \mathbb{I}(r_i \in Y)}{\sum_{r_j \in Y} \exp(\mathbf{r}_j^T\mathbf{h} + b_j)}$$

We can find that when $|Y| = 1$ (only one label is assigned to the relation mention), $q(.|Y, \mathbf{h})$ would be one-hot and Equation 9 would become the classical cross entropy loss.

Logistic Regression is applied over the extracted features as a baseline method.[2]

A.1.2 Neural Models

We employed several popular neural structure to calculate the sentence representation $\mathbf{h}$. As to the objective function, we use Equation 9 for all the following neural models.

Bi-LSTMs and Bi-GRUs use Bidirectional RNNs to encode sentences and concatenate their final states in the last layer as the representation. Following previous work (Zhang et al., 2017), we use two types of RNNs, Bi-LSTMs and Bi-GRUs. Both of them have 2 layers with 200d hidden state in each layer.

Position-Aware LSTM computes sentence representation with an attention over the outputs of LSTMs. It treats the last hidden state as the query and integrates a position encoding to calculate the attention (Zhang et al., 2017).

CNNs and PCNNs use convolution neural networks as the encoder. In particular, CNN directly appends max-pooling after the convolution layer (Zeng et al., 2014); PCNN uses entities to split each sentence into three pieces, and does max-pooling respectively on these three pieces after the convolution layer. Their outputs are concatenated as the final output (Zeng et al., 2015).

A.2 Model Training

We run each model for 5 times and report the average F1 and standard variation.

[1] CoType and Logistic Regression are exceptions as they don't adopt softmax to generate output.

[2] We use liblinear package from https://github.com/cjlin1/liblinear

Feature Name	Description	Example
Brown cluster	Brown cluster ID for each token	"*BROWN_010011001*"
Part-of-speech (POS) tag	POS tags of tokens between two EMs	"*VBD*" , "*VBN*" , "*IN*"
Entity Mention Token	Tokens in each entity mention	"*TKN_EM1_Hussein*"
Entity mention (EM) head	Syntactic head token of each entity mention	"*HEAD_EM1_HUSSEIN*"
Entity mention order	whether EM 1 is before EM 2	"*EM1_BEFORE_EM2*"
Entity mention distance	number of tokens between the two EMs	"*EM_DISTANCE_3*"
Entity mention context	unigrams before and after each EM	"*EM1_AFTER_was*"
Tokens Between two EMs	each token between two EMs	"*was*" , "*born*" , "*in*"
Collocations	Bigrams in left/right 3-word window of each EM	"*Hussein was*" , "*in Amman*"

Table 4: Text features used in feature-based models. ("*Hussein*", "*Amman*", "*Hussein was born in Amman*") is used as an example.

Optimization. We use Stochastic Gradient Descent (SGD) for all models. Learning rate is set at 1.0 initially, and is dynamically updated during training, *i.e.*, once the loss on the dev set has no improvement for 3 consecutive epochs, the learning rate will be factored by 0.1.

Hyper-parameters. For ReHession, dropout is applied on input features and after average pooling. We tried the two dropout rates in $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

For Position Aware LSTM, Bi-LSTM and Bi-GRU, dropout (Srivastava et al., 2014) is applied at the input of the RNN, between RNN layers and after RNN before linear layer. Following (Melis et al., 2017) we tried input and output dropout probability in $\{0.4, 0.5, 0.6, 0.7, 0.8\}$, and intra-layer dropout probability in $\{0.1, 0.2, 0.3, 0.4\}$. We consider them as three separate hyper-parameters and tune them greedily. Following previous work (Zhang et al., 2017), dimension of hidden states are set at 200.

Following previous work (Lin et al., 2016), the number of kernels is set to 230 for CNNs or PCNNs, and the window size is set at 3. Dropout is applied after pooling and $\tanh$ activation . We tried the dropout rates in $\{0.1, 0.2, 0.3, 0.4, 0.5\}$.

B Additional Figures and Tables

Figure 7 shows the full label distribution of TACRED dataset, and the simulated TACRED S5 dataset with a shifted distribution. Figure 8 shows the full label distribution of NYT dataset. NYT is constructed with distant supervision and has a shifted distribution.

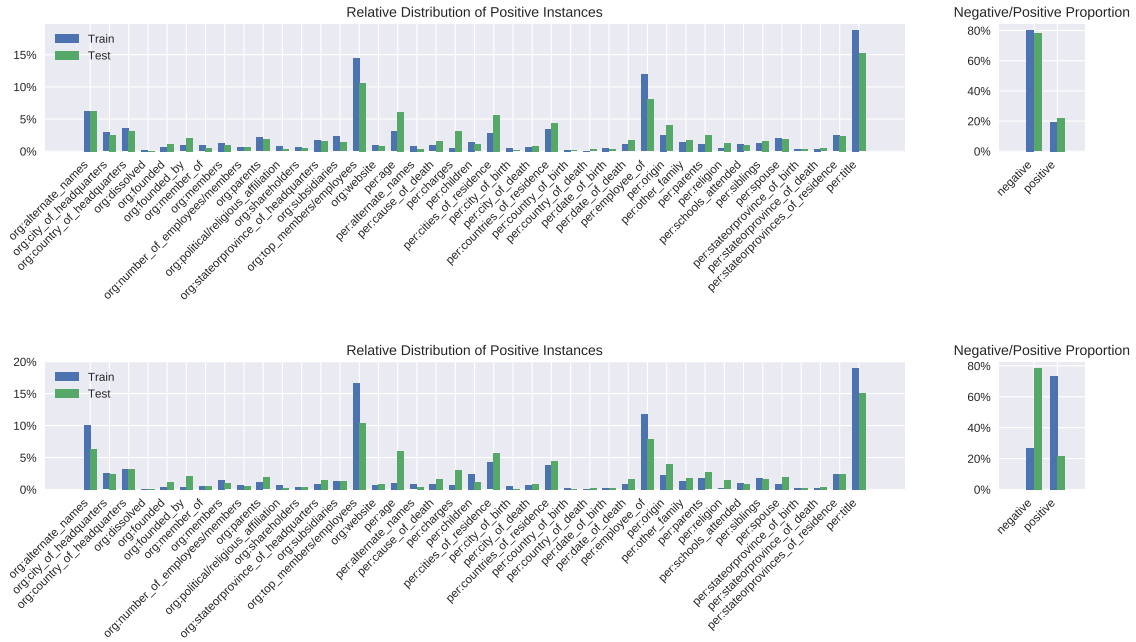

Figure 7: Top: Label distribution of original TACRED; **Bottom:** Using a randomly generated distribution for S5 train set, and keeping original test set. Label distribution of other synthesized datasets (S1-S4) are generated with linear interpolation of these two train set distributions.

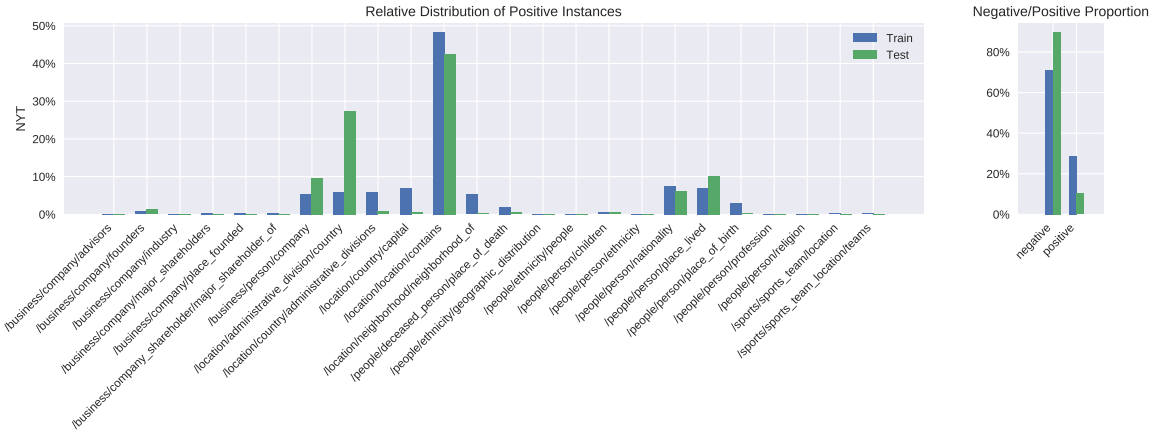

Figure 8: Label Distribution of original NYT. Similar to KBP, label distributions are shifted.