Abstract

The distribution of knowledge elements such as entity types and event types is long-tailed in natural language. Hence information extraction datasets naturally conform a long-tailed distribution. Although imbalanced datasets can teach the model about the useful real-world bias, deep learning models may learn features not generalizable to rare or unseen mentions of entities or events during evaluation, especially for rare types without sufficient training instances. Existing approaches for the long-tailed learning problem seek to manipulate the training data by re-balancing, augmentation or introducing extra prior knowledge. In comparison, we propose to handle the generalization challenge by making the evaluation instances closer to the frequent training cases. We design a new transformation module that transforms infrequent candidate mention representation during evaluation with the average mention representation in the training dataset. Experimental results on classic benchmarks on three entity or event extraction datasets demonstrate the effectiveness of our framework.

1 Introduction

Long-tailed distributions are common in natural language processing tasks. This natural phenomenon of long-tailed distribution is formulated as Zipf’s Law (Reed, 2001). For information extraction, knowledge elements such as entities, relations and events typically conform a long-tailed distribution in natural language. This leads to the imbalanced distribution of types in the benchmark datasets unless manual manipulation is performed to balance the dataset. We show the entity type distribution of an entity extraction dataset (Few-NERD (Ding et al., 2021)), and the event type distribution of two event extraction datasets (ACE 2005 (Walker et al., 2006) and MAVEN (Wang et al., 2020)) in Figure 1. All three datasets have similar long-tailed distributions. Among them, MAVEN and Few-NERD are two relatively large-scale datasets, but the training mentions are still concentrated on a small number of frequent types and majority of the entity/event types are rare types without sufficient training examples.

Long-tailed distribution is a generic problem not limited to NLP research. In fact, a lot of previous work on the long-tailed learning is in the computer vision domain (Lin et al., 2017; Kang et al., 2020). These approaches consider the imbalance in the type distribution as the major problem and are mostly based on balancing the datasets by up-weighting the rare types and downweighting the frequent types. However, we argue that this line of research faces significant challenges when applying to the information extraction task. First, balancing the dataset breaks the real-world long-tailed prior in the datasets and may lead the model to mistakenly predict the long-tailed types too often. We observe this phenomenon in our experiments in Section 3 especially for the Classifier Re-training baseline (Kang et al., 2020). Second, due to the lack of effective data augmentation approaches as in the computer vision domain, upweighting the training mentions will only make the model to repetitively learn the same set of instances and aggravate the overfitting problem.

Instead of the imbalance in distribution, the real challenge of long-tailed learning for NLP is knowledge insufficiency, i.e. long-tailed datasets don’t contain enough examples to acquire generalizable knowledge for the rare types. Learned models tend to capture features such as event trigger words for rare types and fail to generalize well to rare or even unseen cases (unseen trigger words) of those types during evaluation. For instance, if an event extraction model is trained on a dataset where all Acquit events are triggered by the keyword acquit, it may not identify the Acquit event trig-
In this work we take the task of both entity extraction and event extraction as sequence labeling. Our contributions to solving the long-tailed learning problem are twofold:

- We provide a new framework by bypassing the knowledge insufficiency problem, facilitating the model to capture more generalizable features.
- We demonstrate that our framework aligns with causal inference approaches.

To summarize, we propose a new approach for learning from long-tailed datasets for entity extraction and event extraction. Our contributions are two-fold:

- We provide a new framework by bypassing the knowledge insufficiency problem and propose a novel learning framework in this perspective.
- We found that our framework aligns theoretically with the causal inference approach and can facilitate the model to capture more generalizable features.

## 2 Approach

### 2.1 Task Definition

In this work we take the task of both entity extraction and event extraction as sequence labeling.

- We found that our framework aligns theoretically with the causal inference approach and can facilitate the model to capture more generalizable features.

To summarize, we propose a new approach for learning from long-tailed datasets for entity extraction and event extraction. Our contributions are two-fold:

- We provide a new framework by bypassing the knowledge insufficiency problem and propose a novel learning framework in this perspective.
- We found that our framework aligns theoretically with the causal inference approach and can facilitate the model to capture more generalizable features.
problems. The input to the model $H$ is a sequence of tokens $x = \{t_0, t_1, \ldots, t_n\}$, and the model predicts a label for each token $H(x) = \{l_0, l_1, \ldots, l_n\}$. Each $l_i$ is either one of the entity/event types or the None type for none-entity or none-event tokens.\footnote{We adopt the IO labeling schema instead of BIO labeling schema to be consistent with the annotations in the Few-NERD dataset (Ding et al., 2021)}

### 2.2 Overview of the Learning Framework

The learning of our proposed long-tailed extraction framework includes two steps. First, we finetune a language model with an additional classifier head for the sequence labeling problem without special treatments for the long-tail problem. In the second step, we train the transformation module while fixing the parameters of the language model. We adopt this two-stage training approach because we redeem to be helpful for the learning of the transformation module. We will introduce these two steps in the following sections. The overall architecture is shown in Figure 2 and the two-staged training strategy is illustrated in Figure 3.

### 2.3 Finetuning Step

Given an input sequence $x = \{t_0, t_1, \ldots, t_n\}$, we first encode it with a pretrained language model $M$ into $M(x) = \{t_0, t_1, \ldots, t_n\}$. We then adopt a linear discriminator $D$ to predict the label for each token

$$l_i = \arg \max D(t_i).$$

We finetune $M$ and $D$ with the cross entropy loss.

### 2.4 Learning the Transformation Module

The transformation module $S$ transforms the representation for each token $t_i$ into a weighted combination of its original representation and the average training representation of each event type.

Our goal is to compute weights $g_e, g_k$ for the final prediction $P(y_k|g_e + g_k r_k, c)$ for each type $y_k$. However, this would require feeding the inputs multiple times to the language model, which is inefficient. Hence, we simplify the prediction to $P(y_k|g_e + \sum_{i=1}^{K} g_k r_k, c)$. In other words, our transformation module combines the average representations of all types together in the hidden layer.

We can essentially perform the transformation in any hidden layers of the pretrained language model $M$. In this work, we perform transformation in the embedding layer to take full advantage of $M$’s capability of encoding contextual information.

Specifically, let $E(t_i) = e_i$ be the embedding representation of the token $t_i$. Let $\{r_j\}_{j=1}^{K}$ be the average training embedding representation for the entity/event type $j$, i.e.,

$$r_j = \frac{1}{\#j} \sum_{i|l(t_i)=j} E(t).$$

![Figure 2: Overall long-tail learning information extraction framework. We ignore the original token $t_i$ and only show the token embeddings $e_i$ in this figure for conciseness. Here we use $M\setminus E$ to indicate the language model layers after the embedding layer and $D$ as the classifier heads for the sequence labeling task. The transformation module $S$ combines rare candidate mentions with average training mentions. The inputs to the transformation module includes the frequency feature $f_i$, the similarity features $b_i$ between the token embedding $e_i$ and the average training embeddings $\{r_j\}_{j=1}^{K}$, and the context features $c_i$.](image-url)
Here \( \{t_l \mid (t_l) = j \} \) refers to the training tokens with the label \( j \) and \( \#j \) is the total number of such tokens. The transformation module computes weights to combine each \( e_t \) with \( \{r_j\}_{j=1}^{K} \). We use the following information as input to compute the weights for combination:

- \( t_i \)'s frequencies of being labeled as entity/event types: \( f_i = [\exp(\gamma f_{ij})]_{j=1}^{K} \). Here we rescale the original frequencies inspired by (Lin et al., 2020). We use \( \gamma = 0.1 \) following (Lin et al., 2020). We expect the transformation module to provide most help in rare cases when \( f_i \) is small.

- The similarity of \( t_i \)'s embedding \( e_t \) with average training embeddings \( r_j \): \( b_i = [\cos(e_t, r_j)]_{j=1}^{K} \). If \( b_i \) contains large values, \( t_i \) is close to some event type’s average embedding and potentially not a rare mention. On the other hand, if \( b_i \) is very small, \( t_i \) deviates significantly from some event type’s average embedding and the transformation may significantly alter the meaning of the input. Taking these into consideration, we propose to model the similarity value \( b_i \) with a quadratic function, i.e. we use \( \tilde{b}_i = [b_i; b_i^2] \) as the similarity features to the transformation module.

- The context representation \( c_t \), which is computed by the pretrained language model \( \mathcal{M} \) while masking \( t_i \) in the input sequence \( x \). Hence \( c_t \) is independent of the token \( t_i \). This is to make sure that the transformation module is not overfitted only on the seen candidate tokens during training, because our goal is to apply the transformation module to rare cases in the evaluation corpus.

We decompose the computation of these weights into two steps. We first compute aggregation weights \( \{\alpha_j\}_{j=1}^{K} \) for \( \{r_j\}_{j=1}^{K} \) with an attention module based on the context \( c_t \):

\[
\alpha_j = \frac{\exp(\langle a_j, c_t \rangle)}{\sum_{k=1}^{K} \exp(\langle a_k, c_t \rangle)}.
\]

(3)

Here \( \{a_j\}_{j=1}^{K} \) are trainable weights. We then feed all three features \( \{f_i; \tilde{b}_i; c_t\} \) into a linear layer with the sigmoid activation to compute two gating scores \( \langle g_{e}, g_{s} \rangle \in [0, 1]^2 \). The final embedding representation is computed as

\[
h_i = g_{e} e_t + g_{s} s_i.
\]

(4)

We then substitute \( e_t \) with \( h_i \) for the following layers in \( \mathcal{M} \).\(^3\)

We train the transformation module \( S \) using the same cross entropy loss as the finetuning step. We also compute the cross entropy on the attention weights of the entity/event types in Equation (3) as an additional loss. We fix the parameters of the finetuned language model \( \mathcal{M} \). This ensures that the similarity features \( \tilde{b}_i \) and the context representations \( c_t \) remain fixed during the learning of \( S \). We provide training details in the Appendix.

\(^3\)From Equation (4) we essentially have \( g_{e} = g_{s}\alpha_k \).
2.5 Connection to Causal Inference

We found our architecture can be mathematically interpreted as a backdoor adjustment (Pearl et al., 2000) method in causal inference, which is used to mitigate the effect of some confounding factor $u$ in making decisions. In our case, it can be formulated as the following prediction probability

$$P(l_i|do(t_i)) = \sum_u P(l_i|t_i, U = u)P(U = u),$$

(5)

where $u$ is the value space of $U$. We refer readers to (Pearl et al., 2000) for the derivation of the above equation. If we select the confounding factor $U$ as the type prediction based on the context around the token $t_i$ (independent of $t_i$), Equation (5) becomes

$$P(l_i|do(t_i)) = \sum_{j=1}^{K} P(l_i|t_i,j)P(j|c_i),$$

(6)

The attention weights $\{\alpha_j\}_{j=1}^{K}$ in Equation (3) is can be considered as probabilities of event types dependent on $c_i$, thus can be seen as $P(j|c_i)$ in the above equation. If we further model $P(l_i|t_i,j) = P(l_i|e_i + r_j)$ and apply the Normalized Weighted Geometric Mean (NWGM) approximation following (Yue et al., 2020),

$$P(l_i|do(t_i)) \approx P(l_i|e_i + \sum_{j=1}^{k} \alpha_j r_j).$$

(7)

In our framework, we are essentially computing

$$P(l_i|t_i) = P(l_i|g_t e_i + g_s \sum_{j=1}^{k} \alpha_j r_j).$$

(8)

This means that $\alpha_j$ composes the deconfounding priors $P(U)$ and $(g_t, g_s)$ serve as the switch for the backdoor adjustment decisions. We expect the model to make decisions on rare or unseen mentions with the backdoor adjustment, and rely on the model’s own prediction with more frequent cases.

3 Experiments

3.1 Datasets and Evaluation

We experiment on an entity extraction dataset, Few-NERD (Ding et al., 2021) and two event extraction datasets, ACE 2005 (Walker et al., 2006) and MAVEN (Wang et al., 2020). We provide data

statistics in Table 1. Few-NERD, MAVEN and ACE include 67, 168, and 33 types respectively. For ACE 2005, we split the dataset such that all event types are covered in the evaluation data which is different from the splits used in previous work. For Few-NERD, we used the “supervised” split for experiments.

For evaluation, we use the macro F1 score as the main metric to reflect the influence of long-tail types. We also report micro F1 for additional reference. We provide macro recall, macro precision, micro recall and micro precision in Appendix. Previous work also reports scores on subsets of rare types (Nan et al., 2021). Instead of manually determine a frequency threshold for the “rarest” types, we plot curves of F1 scores on all types in Figure 4. All results are averaged over three runs using different random seeds.

3.2 Methods in Comparison

We group previous work on long-tailed learning into two categories: balancing-based approaches and causal inference approaches. For balancing-based approaches, we mainly follow (Kang et al., 2020) and implemented Classifier Re-training (CRT), Nearest Class Mean classifier (NCM), $\tau$-normalized classifier ($\tau$-norm) and Learnable weight scaling (LWS). We also implemented the focal loss approach (Lin et al., 2017). For causal inference work, we implemented Momentum (Tang et al., 2020) and CFIE (Nan et al., 2021). Although (Nan et al., 2021) also experimented on the ACE 2005 and MAVEN, we re-implemented5 their methods due to the difference in splits and evaluation strategies. CFIE also has different designs for the entity extraction and event extraction models. We didn’t re-implement the entity model due to insufficient details. We also compare with a Vanilla baseline which is the sequence labeling model without the transformation module.

3.3 Main Results

We show our main results in Table 2 and Figure 4. From Table 2, our framework achieves the best macro F1 scores across three datasets. We also noticed that our approach improves macro F1 scores without suffering from inferior micro F1 scores. Al-

---

5Due to insufficient details in their paper and released code, we implemented their framework to the best of our knowledge. CFIE also requires named entity extraction results as part of the inputs to event extraction. Therefore it is not clear how the entity extraction model is designed.
though the Momentum baseline performs closely to our approach in ACE Macro F1. However, Momentum has a much worse micro F1 sacrificing frequent types and worse performance on the other two datasets. These results also show the superiority of our approach and indicate that our model’s performance on frequent types is not harmed. This is because we do not tackle the long-tailed learning problem in terms of balancing that may affect the natural distribution in the dataset. Instead, we transform the long-tailed evaluation samples into frequent training samples with the transformation module.

Although macro F1 metric puts extra stress on the long-tailed types, the performance on the rare types is still dilated by frequent types especially for datasets with a large number of types such as MAVEN. In order to further investigate the performance on different subsets of event types, we show in Figure 4 macro F1 improvements (over the Vanilla baseline) for the top X fraction of the event types with the fewest training examples. We notice that our approach, together with other approaches, indeed improves the learning with long-tailed types when X is small. We notice that our approach can significantly outperform the baseline for the less frequent types with a maximum of over 10% on ACE 2005 event dataset. We found our approach has the best performance of the moderately long-tailed event types compared with other long-tailed learning approaches. For extremely long-tailed types, our performance is also close to the best method Momentum (Tang et al., 2020). One possible reason of our model’s ineffectiveness on extremely long-tailed types is that we cannot learn reliable representation $e_i$ to compute similarity values $\hat{b}_i$ for the transformation module. Another possible reason is that the extreme long-tailed types don’t have enough training instances to provide an informative average training representation $r_k$. A potential solution would be introducing external knowledge (e.g., a few keywords as candidate event triggers or entity mentions) to enrich the $r_k$. Moreover, our approach is more consistent than other methods when X grows larger. This further validates that our approach improves the long-tailed types without sacrificing the performance of frequent event types. For more references, we also provide top 10 event types in MAVEN dataset where our method achieves most F1 performance gain on in Appendix. Majority of those types are rare types.

### 3.4 Improving Classifier Features

In our second training stage, we finetune the entity/event classifier head together with the transformation module. Finetuned classifiers from our approach should benefit from the connections of our approach and the backdoor adjustment approach for causal inference in Section 2.5, which encourages the model to capture generalizable features instead of surface correlations. To test the fine-tuned classifiers alone, we evaluate our framework with the transformation module disabled, i.e. forcing $(g_e, g_s) = (1, 0)$ in Equation (4). The forward architecture becomes exactly the same as the Vanilla model. In Table 3, we observe that fine-tuned classifiers alone outperform the Vanilla baseline. Since we fixed the language model in the second stage, the improvements purely come from the finetuned classifiers that have learned to avoid surface features for prediction.

In addition to the interpretation of the improvements from the causal inference theory, we give a more intuitive explanation based on Equation (4). During the second training step, Equation (4) can be seen as changing the original candidate representation with an interpolation of itself and other candidates’ representations of the same event type. This is similar to augmenting the dataset by interchanging mention spans of the same type across sentences, though we interchange it with the average representation of that type in the entire training dataset. The classifier may benefit from this “aug-

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Few-NERD #Sentences</th>
<th># Mentions</th>
<th>MAVEN #Sentences</th>
<th># Mentions</th>
<th>ACE 2005 #Sentences</th>
<th># Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>train</td>
<td>131,767</td>
<td>340,387</td>
<td>32,431</td>
<td>77,993</td>
<td>16,807</td>
<td>4,254</td>
</tr>
<tr>
<td>dev</td>
<td>18,824</td>
<td>48,770</td>
<td>8,042</td>
<td>18,904</td>
<td>2056</td>
<td>500</td>
</tr>
<tr>
<td>test</td>
<td>37,648</td>
<td>96,902</td>
<td>9,400</td>
<td>21,835</td>
<td>1,930</td>
<td>570</td>
</tr>
</tbody>
</table>

Table 1: Dataset statistics.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Few-NERD</th>
<th>MAVEN</th>
<th>ACE 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Macro F1</td>
<td>Micro F1</td>
<td>Macro F1</td>
</tr>
<tr>
<td>Vanilla</td>
<td>62.4</td>
<td>67.8</td>
<td>60.1</td>
</tr>
<tr>
<td>Focal</td>
<td>62.5</td>
<td>67.8</td>
<td>59.4</td>
</tr>
<tr>
<td>CRT</td>
<td>44.9</td>
<td>54.3</td>
<td>51.2</td>
</tr>
<tr>
<td>LWS</td>
<td>61.4</td>
<td>66.9</td>
<td>60.5</td>
</tr>
<tr>
<td>τ-norm</td>
<td>62.4</td>
<td>67.8</td>
<td>60.2</td>
</tr>
<tr>
<td>NCM</td>
<td>51.5</td>
<td>50.4</td>
<td>57.5</td>
</tr>
<tr>
<td>CFIE</td>
<td>-</td>
<td>-</td>
<td>56.3</td>
</tr>
<tr>
<td>Momentum</td>
<td>59.7</td>
<td>63.9</td>
<td>60.1</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>62.6</strong></td>
<td><strong>67.8</strong></td>
<td><strong>61.2</strong></td>
</tr>
</tbody>
</table>

Table 2: Macro and micro F1 scores (in %) on three datasets. References to the methods above is provided in Section 3.2.

Figure 4: Performance difference compared with Vanilla w.r.t the portion of rarest event types. We first rank event types by the number of training mentions from low to high, and then compute average F1 scores (minus that of the Vanilla baseline) for the first X fraction of the event types. We omitted those approaches that are significantly lower than Vanilla.

Table 3: Performance (in %) with the transformation module disabled (No Sub) on the ACE 2005 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Ours</th>
<th>No Sub</th>
<th>Vanilla</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro F1</td>
<td>62.4</td>
<td>61.9</td>
<td>60.4</td>
</tr>
</tbody>
</table>

3.5 Case Study

We also show some examples that the Vanilla baseline misses but our approach correctly identifies and classifies the candidate mentions in Table 4. We also visualize the transformation weights \((g_e, g_s)\) in Equation (4). As discussed in Section 3.3, our model is the best at handling moderately long-tailed types. For these two examples, the event Start-Position in the first case has 94 mentions and became appears only 2 times as the trigger. In the second case, the Action appears 709 times, but only 16 of them are triggered by undertaken\(^6\). It is worth mentioning that although \(g_s\) is not as large as \(g_e\) in both cases, we found that they are indispensable since the model fails on both cases if we disable the transformation module by forcing \(g_s = 0\).

4 Related Work

4.1 Balancing-based Long-tailed Learning for Computer Vision

Long-tailed learning is closely related to imbalanced learning. In the computer vision domain, Lin et al. (2017) proposes the focal loss to handle

\(^6\)As a comparison, war appears 270 times as the trigger of Attack in ACE 2005 and storm appears 757 times as the trigger of Catastrophe in MAVEN.
This upcoming visit to Russia will be my first trip aboard since I became president of China. (Event: Start-Position)

The operation was undertaken so that Allies could secure a beachhead. (Event: Action)

<table>
<thead>
<tr>
<th>Input Sentence Dataset Transformation Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>This upcoming visit to Russia will be my first trip aboard since I became president of China. (Event: Start-Position)</td>
</tr>
<tr>
<td>The operation was undertaken so that Allies could secure a beachhead. (Event: Action)</td>
</tr>
</tbody>
</table>

Table 4: Examples of missing event triggers by the Vanilla model. Transformation weights ($g_e$, $g_e$) in Equation (4) are visualized.

the imbalanced learning problem in object detection. The focal loss downweights the loss terms for confident training samples so that the model predicts high probabilities for the gold standard labels. In addition to this, some work (Zhang et al., 2019; Rebuffi et al., 2017), though not focusing on the long-tailed learning problem, adopts special training strategies to tackle the imbalance in the datasets. Kang et al. (2020) summarize these approaches and evaluate them with the computer vision tasks. These approaches are mostly based on balancing the datasets by upweighting the rare types and downweighting the frequent types, either by modifying the sampling strategy or associating weights to the loss terms. These methods first learn the feature extractor with the original imbalanced distribution and then perform special treatments to the classification layer. Classifier Re-training (CRT) retrains the classification layer by sampling examples from each type uniformly. Nearest Class Mean classifier (NCM) uses the average training features as the type-level features and uses certain distance metrics to perform the nearest neighbor classification. $\tau$-normalized classifier ($\tau$-norm) normalizes the type weights in the classification layer. This can make sure the weights for frequent types are not significantly larger than rare types. Learnable weight scaling (LWS) is similar to $\tau$-norm but learns the normalization weights while sampling examples from each type uniformly.

### 4.2 Long-tailed Learning with External Knowledge for IE

Balancing-based methods are usually ineffective in information extraction as shown in our experiments. Existing work usually tackles the long-tailed problem in information extraction (Han et al., 2018; Zhang et al., 2019) by incorporating external structural knowledge. Han et al. (2018) adopt hierarchical structures among relation types to transfer knowledge from the frequent relation types to their siblings. Zhang et al. (2019) incorporate label semantics and knowledge graph embeddings to transfer knowledge from frequent types to rare types. Required expertise in selecting external knowledge limits the generalization of these methods. (Yu et al., 2021) also leverages correlation between event types to help the learning of rare event types in the context of lifelong learning.

### 4.3 Causal Inference and Its Application to Long-tailed Learning

Due to the potential of causal inference (Pearl et al., 2000) theory to reduce the spurious correlation, there have been explorations on its application in machine learning (Lopez-Paz et al., 2017; Magliacane et al., 2018; de Haan et al., 2019; Bengio et al., 2020; Yang et al., 2020; Li et al., 2021; Park et al., 2021). Since the the spurious correlation is more common in limited data, some work attempts to interpret the long-tailed learning problem under a causal inference framework. The core component of this interpretation is to find a confounding factor that affects the distribution of the input features and output labels at the same time. Tang et al. (2020) consider the training momentum of the gradients to be the confounding factor and proposed the corresponding deconfounded training with the backdoor adjustment approach. They use the total direct effect (TDE) for inference, which essentially lowers the probabilities of frequent types systematically. Nan et al. (2021) work on the information extraction tasks and take a set of linguistic features as the confounding factor. They also adopt an inference approach similar to TDE to lower the probabilities of frequent types.

### 4.4 Related Few-shot Learning Methods

Few-shot learning aims at training models with a small number of instances, which is similar to the
goal of improving rare types in long-tailed learning. Some few-shot learning methods also overlap with long-tailed learning methods. Snell et al. (2017) proposed a prototypical network that has a similar framework as NCM for long-tailed learning. Yue et al. (2020) proposed a backdoor adjustment approach based on the causal inference theory to reduce the spurious correlation in the model, since the few-shot learning also has the limited data size problem. However, few-shot learning usually works on the $N$-way $K$-shot setting. This makes these approaches usually not directly applicable to the long-tailed learning scenario.

5 Conclusions and Future Work

In this work we propose a new long-tailed learning framework for entity and event extraction by candidate transformation. We design a novel transformation module to convert representations of rare or unseen mentions during evaluation into representations of average training mentions. Experimental results have validated the effectiveness of our framework. Our framework can significantly improve the performance on long-tailed types, and outperform other long-tailed learning methods especially for moderately long-tailed types. Moreover, our framework does not sacrifice the performance on frequent types. We also discover the connections between our learning framework and the backdoor adjustment in the causal inference theory. We empirically observe that our training strategy can improve the model’s capability in capturing more generalizable features, which aligns with the causal inference theory. In the future, we will explore: (1) adapt the concept of transformation module to other NLP tasks; (2) based on our connection with the causal inference theory, it is possible to design a better transformation module by choosing a better confounding factor other than the context information.

6 Limitations

In terms of the framework design, our current design of the framework is only applicable to the sequence labeling task, although we believe it can be adapted to other NLP tasks without significant modifications. Besides, our framework should be most helpful if the semantics of the elements in the sequence of the same type are close to each other and thus may require additional modifications in more heterogeneous cases, such as vision-language models where visual features may have the same type as the textual token embeddings.

In terms of time efficiency, our framework will require an additional training stage to learn the transformation module, which will cost extra time. Since we fixed the language model, which composes majority of parameters in our framework, this additional cost is acceptable. We also recommend to pre-process input features (since they are fixed with respect to training samples during the second stage) to the transformation module to further reduce the time cost in the second stage.

7 Acknowledgement

This research is based upon work supported in part by U.S. DARPA LORELEI Program No.HR0011-15-C-0115, U.S. DARPA AIDA Program No. FA8750-18-2-0014 and KAIROS Program No. FA8750-19-2-1004. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References


Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018. Hierarchical relation extraction with coarse-to-fine grained attention. In Proceedings of


Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. 2019. Long-tail relation extraction via knowledge graph

A Appendix

A.1 Details on Two-stage Training

In the first training stage, all parameters in the language model $M$ and entity/event classifier head $D$ are updated. In the second stage, we fix the entire language model and train both $D$ and the transformation module $S$. Apart from the cross entropy loss for entity/event extraction, we also apply loss to the weights $\{\alpha_j\}_{j=1}^K$ in Equation (3). This loss is only applicable to entity/event mention tokens, since $\{\alpha_j\}_{j=1}^K$ corresponds to entity/event types. For these mention tokens, we compute cross entropy between attention weights and the labeled types.

For the input features to the transformation module, we want to make the values of all three kinds of features to be in the similar range. Therefore, we adopt a batch normalization module to these features. We also add dropout with probability 0.2 to similarity features and frequency features, and dropout with probability 0.5 to the context features.

A.2 Implementation Details

For all experiments, we use learning rate to be $1e^{-5}$ and batch size to be 8 to fit in a single Nvidia Tesla V100 GPU with 16GB memory. We evaluate performance after each epoch and select the best model based on the development performance. We use early-stop strategy with a patience of 5 epochs. We report average performance over 3 runs initialized with 3 different random seeds.

The approximate number of parameters is 3.5 million (RoBERTa). Added parameters from the transformation module is significantly less and dependant on the number of target entity/event types. Approximately the transformation module has 2,500 parameters. For the first stage training, it takes about 10-20 minutes to train an epoch on ACE 2005 data, 20-30 minutes on MAVEN data and about 40 minutes on Few-NERD. The time difference mainly comes from the total number of sentences in these datasets. However these are just rough estimations since the performance largely depends on the environmental factors such as temperatures and also the workload of other gpus/cpus in the same machine during training. In the second stage, we didn’t preprocess any features in advance but block the backpropagation to language model parameters. In general it is 4-5 times faster than the first stage. We incorporate the transformation module by modifying the RoBERTa implementation from Transformers’ Library.

A.3 Dataset Licenses

This dataset is licensed by LDC. Membership is required for access. The dataset can be used for research purpose. Few-NERD dataset is distributed under the CC BY-SA 4.0 license. MAVEN dataset is distributed under MIT License.

A.4 Additional Results

We include precision and recall scores in Table 5. Moreover, we include top 10 event types in the MAVEN dataset that our approach achieves most improvements over the Vanilla baseline, as well as their number of training mentions in Table.

\[\text{https://huggingface.co/docs/transformers/index}\]

\[\text{https://www.ldc.upenn.edu}\]
Table 5: Macro and micro precision and recall scores (in %) on three datasets.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla</td>
<td>60.1, 65.3</td>
<td>65.8, 70.0</td>
<td>61.9, 61.1</td>
<td>65.7, 68.6</td>
<td>63.6, 62.0</td>
<td>73.4, 72.1</td>
</tr>
<tr>
<td>Focal</td>
<td>60.6, 64.9</td>
<td>66.0, 69.8</td>
<td>57.8, 63.1</td>
<td>62.0, 69.1</td>
<td>61.9, 63.0</td>
<td>73.8, 74.1</td>
</tr>
<tr>
<td>CRT</td>
<td>40.7, 51.1</td>
<td>50.6, 58.8</td>
<td>41.1, 75.1</td>
<td>41.2, 79.8</td>
<td>58.6, 67.2</td>
<td>66.7, 77.9</td>
</tr>
<tr>
<td>LWS</td>
<td>58.1, 65.7</td>
<td>64.3, 69.7</td>
<td>60.1, 63.9</td>
<td>62.4, 71.8</td>
<td>62.5, 64.7</td>
<td>72.0, 73.0</td>
</tr>
<tr>
<td>$\tau$-norm</td>
<td>60.8, 63.8</td>
<td>65.9, 69.9</td>
<td>62.2, 61.1</td>
<td>64.9, 69.4</td>
<td>63.5, 61.8</td>
<td>73.4, 72.0</td>
</tr>
<tr>
<td>NCM</td>
<td>56.0, 65.6</td>
<td>61.0, 43.3</td>
<td>58.9, 63.6</td>
<td>57.3, 65.7</td>
<td>64.2, 56.5</td>
<td>76.1, 68.7</td>
</tr>
<tr>
<td>CFIE</td>
<td>-</td>
<td>-</td>
<td>50.9, 66.5</td>
<td>52.5, 73.5</td>
<td>59.7, 56.9</td>
<td>58.0, 69.7</td>
</tr>
<tr>
<td>Momentum</td>
<td>60.3, 65.4</td>
<td>59.7, 62.5</td>
<td>61.8, 61.9</td>
<td>66.3, 66.5</td>
<td>60.9, 68.3</td>
<td>66.9, 69.5</td>
</tr>
<tr>
<td>Ours</td>
<td>60.3, 65.4</td>
<td>65.8, 70.0</td>
<td>61.3, 63.2</td>
<td>65.5, 69.4</td>
<td>63.5, 65.5</td>
<td>71.4, 74.6</td>
</tr>
</tbody>
</table>

Table 5: Macro and micro precision and recall scores (in %) on three datasets.

Table 6: Top 10 event types in the MAVEN dataset that our approach achieves most improvements over the Vanilla baseline. We also include the number of training mentions and F1 score improvements (in %) in the second the third columns. MAVEN has a total of 168 event types and our training split includes 77,987 training event mentions in total for all event types.