# Template Filling with Generative Transformers

**Xinya Du** **Alexander M. Rush** **Claire Cardie**
Department of Computer Science
Cornell University
{xdu, cardie}@cs.cornell.edu
arush@cornell.edu

## Abstract

Template filling is generally tackled by a pipeline of two separate supervised systems – one for role-filler extraction and another for template/event recognition. Since pipelines consider events in isolation, they can suffer from error propagation. We introduce a framework based on end-to-end generative transformers for this task (i.e., GTT). It naturally models the dependence between entities both within a single event and across the multiple events described in a document. Experiments demonstrate that this framework substantially outperforms pipeline-based approaches, and other neural end-to-end baselines that do not model between-event dependencies. We further show that our framework specifically improves performance on documents containing multiple events.

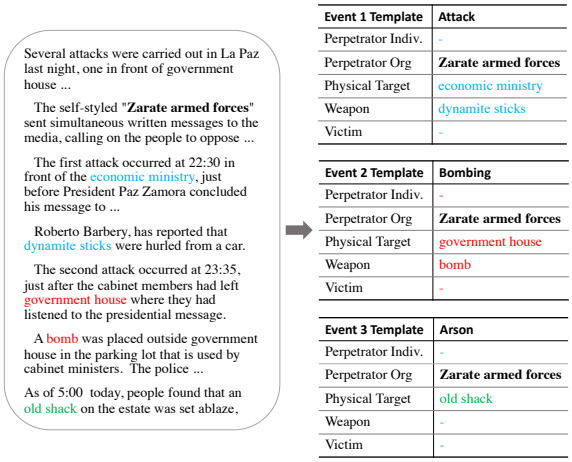


Figure 1: The template-filling task. Role-filler entity extraction is shown on the left, and template recognition is shown on the right. Our system performs both of these document-level tasks with a single end-to-end model.

## 1 Introduction

The classic template-filling task in information extraction involves extracting event-based templates from documents (Grishman and Sundheim, 1996; Jurafsky and Martin, 2009; Grishman, 2019). It is usually tackled by a pipeline of two separate systems, one for *role-filler entity extraction* – extracting event-relevant entities (e.g., noun phrases) from the document; another for *template/event recognition* – assigning each of the candidate role-fillers to the event(s)/template(s) that it participates in and identifying the type of each event/template.

Simplifications of the task (Patwardhan and Riloff, 2009; Huang and Riloff, 2011, 2012; Du et al., 2020) assume that there is one generic template and focus only on role-filler entity extraction. However, real documents often describe multiple events (Figure 1). From the example, we can observe that between-event dependencies are important (e.g., a single organization can participate in multiple events) and can span the entire document (e.g., event-specific targets can be distant from their shared perpetrator organization). Alternative end-to-end event extraction models, even those incorporating pretrained LM representations, only model events in isolation (Wadden et al., 2019; Du and Cardie, 2020), and are mainly evaluated on ACE-style (Doddington et al., 2004) event extraction from single sentences (Yang and Mitchell, 2016; Lin et al., 2020).

To naturally model between-event dependencies across a document for template filling, we propose a framework called "GTT" based on generative transformers (Figure 2). To our best knowledge, this is the first attempt to build an end-to-end learning framework for this task. We build our framework upon GRIT (Du et al., 2020), which tackles role-filler entity extraction (REE), but not template/event recognition. GRIT performs REE by "generating" a sequence of role-filler entities, one role at a time in a prescribed manner. For the template-filling setting, we first extend the GRIT approach to include tokens representing event types



*Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 909–914
June 6–11, 2021. 

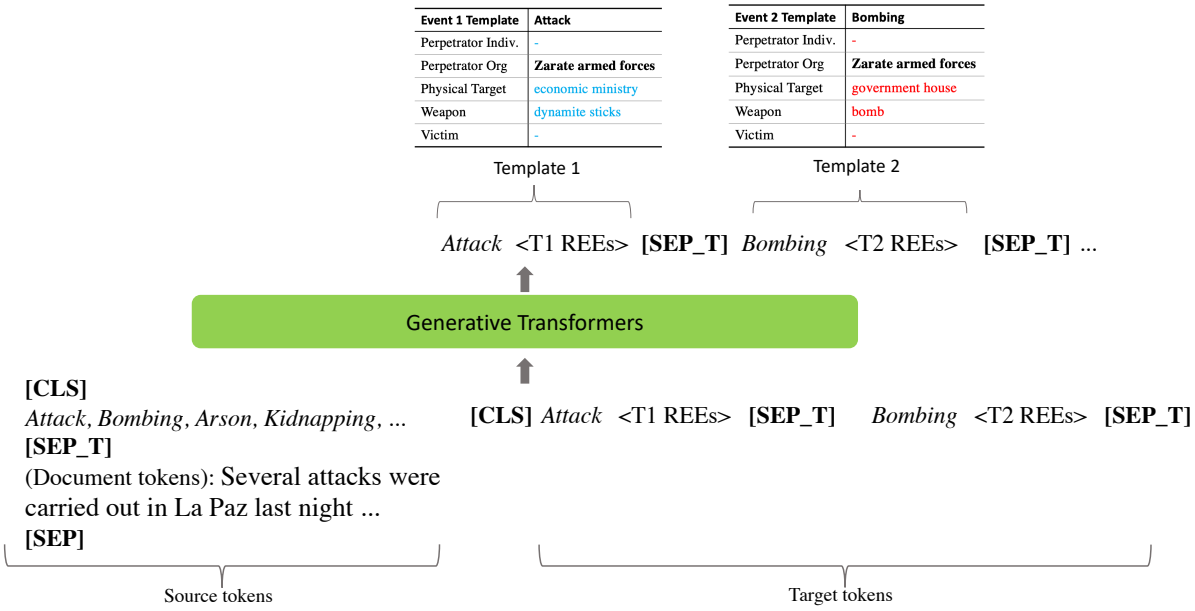


Figure 2: Our generative framework for end-to-end template filling.

(e.g., "attack", "bombing") as part of the input sequence. We further modify the decoder to attend to the event type tokens, allowing it to distinguish among events and associate event types to each role-filler entity that it generates.

We evaluate our model on the MUC-4 (1992) template filling task. Empirically, our model substantially outperforms both pipeline-based and end-to-end baseline models. In our analysis, we demonstrate that our model is better at capturing between-event dependencies, which are critical for documents that describe multiple events. Code and evaluation scripts for the project is open-sourced at https://github.com/xinyadu/gtt.

## 2 Task Definition: Template Filling

Assume we are given a set of $m$ event types ($T_1$, ..., $T_m$). Each event template contains a set of $k$ roles ($r_1$, ..., $r_k$). For a document consisting $n$ words $x_1$, $x_2$, ..., $x_n$, the system is required to extract $d$ templates, where $d \geq 0$ ($d$ is not given as input). Each template consists of $k+1$ slots: the first slot represents the event type (one of $T_1$, ..., $T_m$). The rest of the $k$ slots correspond to an event role (one of $r_1$, ..., $r_k$). The system is required to fill in entities for the corresponding role, which may be filled in as null.

## 3 Methodology

Our framework is illustrated in Figure 2. First we transform the template filling task into a sequence generation problem. Then, we train the base model on the source-target sequence pairs, and apply the model to generate the sequence; finally the sequence is transformed back to structured templates.

### 3.1 Template Filling as Sequence Generation

We first transform the task's input and output data into specialized source and target sequence pair encodings. As shown in Figure 2 and below, the *source sequence* consists of the words of the document ($x_1$, $x_2$, ..., $x_n$) prepended with the general set of tokens representing all event/template types ($T_1$, ..., $T_m$); as well as a separator token denoting the boundary between event templates ([SEP_T]). We also add a classification token ([CLS]) and another separator token ([SEP]) at the beginning and end of this source sequence. [CLS] works as the start token, [SEP] denotes the boundary between REEs.

$$\text{[CLS]}\ T_1, ..., T_m\ \text{[SEP\_T]}$$
$$x_1, x_2, ..., x_n\ \text{[SEP]}$$

The *target sequence* consists of the concatenation of template extractions, separated by the separator token ([SEP_T]). For template $i$, the subsequence consists of its event type $T^{(i)}$ and its role-

filler entity extractions $<$ Role-filler Entities $>^{(i)}$:

$$\begin{gathered}
\text{[CLS]}\ T^{(1)}, < \text{Role-filler Entities} >^{(1)} \\
\text{[SEP_T]}\ T^{(2)}, < \text{Role-filler Entities} >^{(2)} \\
\ldots \\
\text{[SEP_T]}\ T^{(i)}, < \text{Role-filler Entities} >^{(i)} \\
\ldots
\end{gathered}$$

For the $<$ Role-filler Entities $>$ of template $i$, following Du et al. (2020), we use the concatenation of target entity extractions for each role, separated by the separator token ([SEP]). Each entity is represented with its first mention's beginning ($b$) and end ($e$) tokens:

$$e^1_{1_b}, e^1_{1_e}, .. \text{ [SEP] } e^2_{1_b}, e^2_{1_e}, .. \text{ [SEP] } e^3_{1_b}, e^3_{1_e}, ..$$

### 3.2 Base Model and Decoding Constraints

Next we describe the base model as well as special decoding constraints for template filling.

**BERT as Encoder and Decoder** Our model extends upon the GRIT model for REE (Du et al., 2020). The base setup utilizes one BERT (Devlin et al., 2019) model for processing both the source and target tokens embeddings. To distinguish the encoder / decoder representations, it uses partial causal attention mask on the decoder side (Du et al., 2020). The joint sequence of source tokens' embeddings $(\mathbf{a}_0, \mathbf{a}_1, ..., \mathbf{a}_m)$ and target tokens' embeddings $(\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_n)$ are passed through BERT to obtain their contextualized representations,

$$\begin{aligned}
&\hat{\mathbf{a}}_0, \hat{\mathbf{a}}_1, ..., \hat{\mathbf{a}}_{l_{src}}, \hat{\mathbf{b}}_0 ..., \hat{\mathbf{b}}_{l_{tgt}} \\
&= \texttt{BERT}(\mathbf{a}_0, \mathbf{b}_1, ..., \mathbf{a}_{l_{src}}, \mathbf{b}_0, ..., \mathbf{b}_{l_{tgt}})
\end{aligned}$$

**Pointer Decoding** For the final decoder layer, we replace word prediction with a simple pointer selection mechanism. For target time step $t$, we first calculate the dot-product between $\hat{\mathbf{b}}_t$ and $\hat{\mathbf{a}}_0, \hat{\mathbf{a}}_1, ..., \hat{\mathbf{a}}_m$,

$$c_0, c_1, ..., c_{l_{src}} = \hat{\mathbf{b}}_t \cdot \hat{\mathbf{a}}_0, \hat{\mathbf{b}}_t \cdot \hat{\mathbf{a}}_1, ..., \hat{\mathbf{b}}_t \cdot \hat{\mathbf{a}}_{l_{src}}$$

Then we apply softmax to $c_0, c_1, ..., c_{l_{src}}$ to obtain the probabilities of pointing to each source token (which may be a word or an event type), test prediction is done with greedy decoding. At each time step, argmax is applied to find the source token which has the highest probability. The decoding stops when a stop token is predicted.

$$p_0, p_1, ..., p_{l_{src}} = \text{softmax}(c_0, c_1, ..., c_{l_{src}})$$

We also add several special decoding constraints for template filling: (1) downweighting factor (0.01) to the probability of generating [SEP] and [SEP_T], in order to calibrate recall; (2) decoding cutoff stop when it ends the $k^{th}$ template ($k$ =maximum number of events in one document); (3) a constraint to ensure that the pointers for the start and end token for one entity are in order.

## 4 Experiments

We conduct evaluations on the MUC-4 dataset (1992). MUC-4 consists of 1,700 documents with associated templates. We follow prior work in split: 1,300 documents for training, 200 documents (`TST1+TST2`) as the development set and 200 documents (`TST3+TST4`) as the test set. We use the metric for template filling (Chinchor, 1992) and, as in previous work, map predicted templates to gold templates during evaluation so as to optimize scores. We follow content-based mapping restrictions, i.e., the event type of the template is considered essential for the mapping to occur.[1] Missing template's slots are scored as missing, spurious template's slots are scored as spurious. Note that in our work, since we do not extract the set fillers other than the event/template type, they do not affect the performance.

**Baselines and Additional Related Work** As an ablation baseline, we employ a pipeline, GRIT-PIPELINE, that first uses the GRIT model for role-filler entity extraction, and then assigns event types to each of the entities as a multi-label classification problem. We assign types by transforming the problem to multi-class classification (MCC) (Spolaor et al., 2013). As there are 6 event types (i.e., *kidnapping*, *attack*, *bombing*, *robbery*, *arson*, *forced work stoppage*) in MUC-4, we use $2^6$ labels for the MCC problem.

We also compare to end-to-end baselines without modeling between-event dependencies,

[1] The content-based mapping restrictions were added to MUC-4 to prevent fortuitous mappings which occurred in MUC-3 (Chinchor, 1992).

| Models | Event Type | PERPIND | PERPORG | TARGET | VICTIM | WEAPON |
|---|---|---|---|---|---|---|
| GRIT-PIPELINE | 62.28 | 38.40 | 35.36 | 36.30 | **54.97** | 53.45 |
| DYGIE++ (Wadden et al., 2019) | 61.95 | 32.44 | 25.73 | **45.04** | 49.48 | 51.60 |
| SEQTAGGING (Du and Cardie, 2020) | 60.22 | 30.59 | 26.79 | 36.60 | 43.62 | 51.70 |
| GTT | **67.44** | **44.04** | **41.79** | 32.39 | 54.12 | **59.71** |

Table 1: Per-slot F1 score.

DYGIE++ (Wadden et al., 2019)[2] is a span-enumeration based extractive model for information extraction. The model enumerates all the possible spans in the document and passes each representation through a classifier layer to predict whether the span represents certain role-filler entity and what the role is. SEQTAGGING is a BERT-based sequence tagging model for extracting the role-fillers entities. A role-filler entity can appear in templates of different event types (e.g., "Zarate armed force" appear in both attack and bombing event). For both baselines, the prediction goal is multi-class classification. More specially, we *adapt* the DYGIE++ output layer implementation to first predict the role-filler entity's role class, and then predicts its event classes conditioned on the entity's role.

Note that Chambers (2013) and Cheung et al. (2013) propose to do event schema induction with unsupervised learning. Given their unsupervised nature, empirically the performance is worse than supervised models (Patwardhan and Riloff, 2009). Thus we do not add these as comparisons.

| Models | P | R | F1 |
|---|---|---|---|
| GRIT-PIPELINE | **63.88** | 37.56 | 47.31 |
| DYGIE++ (Wadden et al., 2019) | 61.90 | 36.33 | 45.79 |
| SEQTAGGING (Du and Cardie, 2020) | 46.80 | 38.30 | 42.13 |
| GTT | 61.69 | **42.36** | **50.23**$^*$ |

Table 2: Micro-average results on the full test set.

## 5 Results and Analysis

Results on the full test set are shown in Table 2. We report the micro-average performance (precision, recall and F1). We see that our framework substantially outperforms the baseline extraction models in precision, recall and F1, with approximately a 4% F1 increase over the end-to-end baselines. It outperforms the GRIT-PIPELINE system by around 3% F1 ($^*$ denotes $p < 0.05$).

[2]Our own re-implementation.

| Models | P | R | F1 | Δ |
|---|---|---|---|---|
| GRIT-PIPELINE | 65.17 | 26.05 | 37.22 | -21.33% |
| DYGIE++ | 69.90 | 27.05 | 39.01 | -14.81% |
| SEQTAGGING | 51.00 | 29.06 | 37.02 | -12.13% |
| GTT | 56.76 | 38.08 | 45.58 | -9.26% |

Table 3: Performance on the subset of documents which contain more than one gold event. Δ: relative change of F1, as compared to the Full Test setting.

Per-slot F1 score is reported in Table 1. The results demonstrate that our framework more often predicts the correct event type, performs better on PERPIND and PERPORG, and achieves slightly worse performance with GRIT-PIPELINE on roles that appear later in the template (i.e., TARGET and VICTIM). We also found that DYGIE++ performs better on TARGET, mainly due to its high precision in role assignment for spans.

**Between-Event Dependencies** We also show results (Table 3) on the subset of documents that contains more than one gold event. We see the F1 score for all systems drops substantially, proving the difficulty of the task, as compared to the single/no event case. When compared to the Full Test setting in Table 2, the baselines all increase in precision and drop substantially in recall, while our approach's precision and recall drop a little. This change is understandable, as the baseline systems are more conservative and tend to predict fewer templates. As the number of gold templates increases, the fewer templates predictions have a better chance of getting matched, but their recall drops as well.

**How performance changes when $E$ increases** In Figure 3, we see that when the number of gold events in the document is smaller ($E = 1, 2$), our approach performs on par with the pipeline-based and DYGIE++ baselines. However, as $E$ grows larger, the baselines' F1 drop significantly (e.g., over -10% as $E$ grows from 2 to 3).

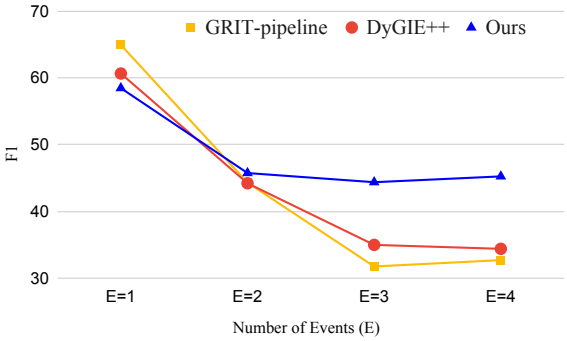


Figure 3: F1 on subset of documents with $E$ events.

**Qualitative Case Analysis** Consider the input document (doc id `TST3-MUC4-0080`)[3], which contains an attack and a bombing template. In the gold annotations, "Farabundo Marti National Liberation Front" acts as PERPORG in both events. Our model correctly extracts the two events and the PERPORG in each while DYGIE++ only predicts the attack event with its PERPORG role entity correctly. Although GRIT-PIPELINE gets both events correct, it failed to extract this PERPORG entity for the second event.

## 6 Conclusion

We revisit the classic NLP problem of template filling and propose an end-to-end learning framework called GTT. Through modeling events relation, our approach better captures dependencies across the document and performs substantially better on multi-event documents.

## Acknowledgments

We thank the anonymous reviewers for helpful feedback and suggestions. The work of XD and AMR was supported by NSF CAREER 2037519; that of XD and CC was supported in part by DARPA LwLL Grant FA8750-19-2-0039.

## References

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Seattle, Washington, USA. Association for Computational Linguistics.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 837–846, Atlanta, Georgia. Association for Computational Linguistics.

Nancy Chinchor. 1992. Muc-4 evaluation metrics. In *MUC*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, page 1. Lisbon.

Xinya Du and Claire Cardie. 2020. Document-level event role filler extraction using multi-granularity contextualized encoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8010–8020, Online. Association for Computational Linguistics.

Xinya Du, Alexander M. Rush, and Claire Cardie. 2020. Grit: Generative role-filler transformers for document-level event entity extraction. In *EACL*.

Ralph Grishman. 2019. Twenty-five years of information extraction. *Natural Language Engineering*, 25(6):677–692.

Ralph Grishman and Beth Sundheim. 1996. Design of the MUC-6 evaluation. In *TIPSTER TEXT PROGRAM PHASE II: Proceedings of a Workshop held at Vienna, Virginia, May 6-8, 1996*, pages 413–422, Vienna, Virginia, USA. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2011. Peeling back the layers: Detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1137–1147, Portland, Oregon, USA. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Dan Jurafsky and James H. Martin. 2009. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International.

[3]For the full document, please refer to Appendix A.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

MUC-4. 1992. Fourth message understanding conference (MUC-4). In *Proceedings of FOURTH MESSAGE UNDERSTANDING CONFERENCE (MUC-4)*, McLean, Virginia.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 151–160, Singapore. Association for Computational Linguistics.

Newton Spolaor, Everton Alvares Cherman, Maria Carolina Monard, and Huei Diana Lee. 2013. A comparison of multi-label feature selection methods using the problem transformation approach. *Electronic Notes in Theoretical Computer Science*, 292:135–151.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299, San Diego, California. Association for Computational Linguistics.

## A Example document for qualitative analysis

Official sources today reported that at least eight people, including soldiers, rebels, and civilians, were killed during clashes between the army and guerrillas over the past weekend in various points of the country.

Military spokesmen for the 6th infantry brigade, headquartered in the eastern usulutan department, told acanefe that two rebels were killed and one wounded during a clash with government troops in San Agustin.

Meanwhile, the armed forces press committee (Coprefa) reported that the bodies of two guerrillas, who were presumably killed during clashes with the army, were found by soldiers in the outskirts of Santa Tecla, in the central la libertad department.

Coprefa reported that two soldiers were killed during a clash with members of the **Farabundo Marti National Liberation Front** (FMLN) in Comasagua, about 28 km to the southwest of (San) Salvador, where a rebel attack on a coffee processing plant was successfully repelled.

It reported that a civilian was killed in the crossfire and that a soldier was also killed during clashes in Zaragoza, south of San Salvador, where two guerrillas were wounded.

...

Salvadoran (red) cross sources today reported that a 48-year-old woman identified as Maria Luz Lopez was wounded last night when a powerful bomb, which damaged several businesses in (San) Salvador, exploded.

The bomb was planted in a heavily commercial area of downtown (San) Salvador causing heavy property loses, according to the owners who provided no specific figures.

This is the fourth dynamite attack on businesses in (San) Salvador so far in 1990.

## B Hyper-Parameters

| hparam name | value |
|---|---|
| BERT model type | `bert-base-uncased` |
| train batch size | 1 |
| eval batch size | 1 |
| num train epochs | 18 |
| seed | 1 |
| number of GPU | 1 |
| learning rate | 5e-5 |
| ADAM epsilon | 1e-8 |
| warmup steps | 0 |
| downweigh factor | 0.01 |

## C Implementations

We build our model upon the HuggingFace's NER models' implementation (`rb.gy/nryu2q`).

**Dependencies**

- Python 3.6.10
- Pytorch 1.4.0
- Pytorch-Lightning 0.7.1
- Transformers: transformers 2.4.1 installed from source.

**Link to Corpus** We obtain the raw corpus from `https://github.com/brendano/muc4_proc`