Implementation: Deeper Task-Specificity Improves Joint Entity and Relation Extraction

Vedant Choudhary
vc389

Aditya Vyas
av634

Abstract

Multi-task learning (MTL) is a subset of machine learning which attempts to share representations between related tasks to achieve a generalized model instead of single task models. For the final project of CS533:Natural Language Processing, we reimplement a preprint (Crone, 2020) which improves joint named entity recognition (NER) and relation extraction (RE) through deeper task-specificity, whilst maintaining a small parameter size and achieving state-of-the-art (SOTA) results. We worked on CoNLL04 dataset. The essential part of the project is to provide an open-source implementation [1] of the preprint and validating its results. Additionally, we look into replacing shared BiRNNs with a novel variant of transformer (Vaswani et al., 2017) architecture which we call *ResBN-Transformer* and see that it performs better than the biRNN architecture on the experiment batch. Before concluding, we also talk about what worked and what didn't work for the project.

1 Introduction

Generally in machine learning, models are created to solve a single individual task even if there is a close similarity between Task A and Task B. This laser-focused approach often leads to good results but it ignores information which can be gained from a similar task. Multi-task learning aims to alleviate this by sharing representations between tasks, either through soft or hard parameter sharing (Ruder, 2017).
NER and RE are very similar and important information extraction tasks with applications in search, question answering etc. NER is a structured prediction problem, in which you are given a token sequence $x = (x_1, ..., x_n) \in X^n$, where X is the vocabulary, and a set of labels Y, such that for every x_i there is a y_i. The goal of the model is to learn a scoring function so that for any $x_1, ..., x_n$ input sequence, you return the most likely output sequence, $y_1, ..., y_n$. Essentially, conditional probabilities $p(y|x)$ are learned. RE classifier will then take into account the two entity candidates pairs (e_i, e_j) drawn from a YxY matrix and the context of the words to estimate the relation from R by again using conditional probabilities. In this case, it can be shown by $p(r|x_{i,j}, e_{i,j})$.
The easiest approach is to assume these tasks are independent and make a sequential model which sends the output of NER task to RE, however, it is susceptible to error propagation from the NER model to RE model, which might make the model learn on the errors. To alleviate this, a joint approach using MTL is taken. MTL usually edges in these kind of situations (Caruana, 1997), (Vandenhende et al., 2019), (Cipolla et al., 2018), (Li et al., 2019a) due to the fact that solutions to related tasks often rely on similar information.
In this project, we have re-implemented (Crone, 2020) which jointly learns named entity recognition and relation extraction tasks through hard parameter sharing. The novelty of the preprint comes from:

- Allowing deeper task-specificity than previous work for both tasks. This is achieved by additional task-specific bidirectional recurrent neural networks (BiRNNs)
- Making number of shared and task-specific layers hyper-parameters of the model so that the model can be customized as per the domain. This is beneficial because not all NER and RE tasks are constant in terms of relation between them
- Small parameter size compared to recent models, yet achieving SOTA results for NER and RE

In addition to that, we have done some extension primarily in the sense that we have replaced shared BiRNN layers with a unique transformer based architecture (*ResBN-Transformer*) suitable for our task.

2 Related Work

This section focuses on work currently being done in the intersection of MTL, NER and RE. Most of the work in NER use BIO or BILOU scheme for NER task. In BIO, the labels indicate (B)eginning of an entity, (I)nside an entity, (O)utside the entity, whereas, in BILOU the labels are used to indicate if a token is the (L)ast token

[1] https://github.com/vedantc6/mtl-dts/

of an entity, or is a (U)nit i.e. the only token within an entity (Crone, 2020).

There have been several approaches where NER and RE have been treated as a single task. (Gupta et al., 2016) considers joint entity and word-level relation extraction task as a table-filling problem to model their interdependencies. The cells in the table are a pair of tokens (t_i, t_j) and filled through a BiRNN. (Li et al., 2019b) use a novel approach of converting entity-relation extraction task to a multi-turn question answering problem. The input text is queried with question templates to detect entities and then, queried to get any relations between them. They use BERT (Devlin et al., 2019) as a backbone model.

Another kind of approach is when the tasks are considered separately. (Katiyar and Cardie, 2017) and (Bekoulis et al., 2018) use models which send token representations to shared BiLSTM layers. (Katiyar and Cardie, 2017) use a softmax layer to tag tokens with BILOU schema and use an attention layer to detect relations amongst the predicted entities. (Bekoulis et al., 2018) use conditional random fields (CRF) for the NER task and then the results are sent to relation scoring and sigmoid layers to predict the relations.

An alternative to BIO/BILOU schema is the span-based approach. The current SOTA model for joint NER and RE is a span-based approach performed by (Eberts and Ulges, 2019). The authors obtain the token representations from a fine-tuned BERT model which are first sent to an entity classification layer to solve NER task and then the representations of the predicted spans along with the context for the respective spans are passed through a final layer with sigmoid activation to predict relations.

The aforementioned approaches have experimented little with deeper task specificity and concentrated more on shared layers and then individual or independent scoring layers. The paper (Crone, 2020) which we reimplement includes additional task-specific layers beyond scoring layers, which no other work has done according to the authors. Further, the number of shared and task-specific layers are treated as hyperparameters unlike the previous papers. This allows for the model to adjust to different textual domains.

3 Model

The paper proposes a multi-task learning architecture to combine both the NER and RE tasks into a single model.

3.1 Shared Layers

Each token embedding t_i is obtained using ELMo 5.5B model (Peters et al., 2018) t_i^{elmo}, 300-d GloVE embeddings t_i^{glove}, a character-level embedding learned via a single BiRNN layer (Lample et al., 2016) t_i^{char} and one-hot encoded casing vector t_i^{casing}. The casing vector has cases when letters of the word are all numeric, lower alphabetic, upper alphabetic, title alphabetic, > 50 numeric, atleast one numeric and others.

The final representation of the token is a concatenation.

$$s_i = t_i^{elmo} \circ t_i^{glove} \circ t_i^{char} \circ t_i^{one-hot}$$

To get the shared representation, the token representations of an input text s_i are sent to $N \in Z^*$ bi-directional recurrent neural networks (biRNNs).

3.2 NER-Specific Layers

The NER layers take as input the shared representations and generates scores for the NER tags for the current input text. The score for token t_i, $score_i^e$ is obtained by passing the shared representations h_i^e through two feed forward layers:

$$score_i^e = FF^{(e2)}(FF^{e1}(h_i^e))$$

The activation function and output size for FF^{e1} is an hyperparameter, FF^{e2} uses linear activation with its output size as $|E|$ denoting the set of possible entity types. The sequence of scores are then sent to a linear-chain CRF layer to produce the final BIO tag sequence. During inference, Viterbi decoding is used to retrieve the sequence.

3.3 RE-Specific Layers

This task is relatively more complicated than the NER task. The idea is to utilise the information from ground truth NER tags to aid the RE task during training, while using predicted NER tags at inference. We have to predict relations between entities e_i and e_j, so we filter the learned token representations such that we use the last token of the above entity spans. Then, the representation h_i^r is concatenated with NER label embeddings for a token t_i:

$$g_i^r = h_i^r \circ l_i^e$$

Next, the scores are calculated for every possible pair (g_i^r, g_j^r) using DistMult score (Yang et al., 2014) for every relation $r_k \in R$

$$DistMult^{r_k}(g_i^r, g_j^r) = (g_i^r)^T M^{r_k} g_j^r$$

Here, $M^{r_k} \in R^{p \times p}$ is a diagonal matrix where p is the dimensionality of g_i^r. After this, the scores are sent to a feed-forward layer where again, the activation function and output size are treated as hyperparameters.

$$f_i^r = FF^{(r1)}(g_i^r)$$

. The final scores are calculated by concatenating DistMult, the above feed-forward and cosine distance between the entity representations. Then, these are sent to a second feed-forward network $FF^{(r2)}$ with linear activation and output size as $|R|$. Final scores are calculated by sending them through element-wise sigmoid layer. A relation is predicted is its score is $\geq \Theta^r$

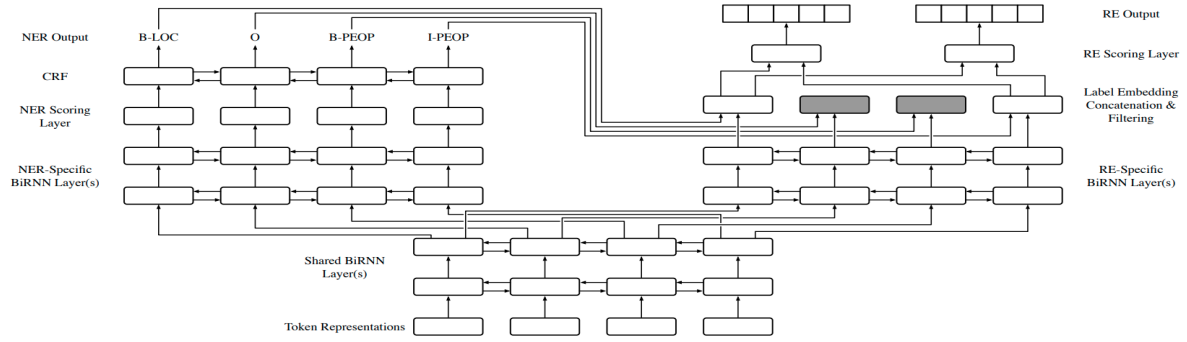


Figure 1: Model Architecture: Only the final token in each entity span is used for predictions for the RE task; grey boxes indicate tokens that are not used for relation predictions

3.4 Training

Character-level embeddings, label embeddings, all BiRNN weights, all feed-forward networks and M^{r_k} are trained in a supervised manner. For NER, BIO tagging is used for labels and the loss L_{NER} is calculated through $CRFLoss$ from *Assignment 4*. For ever relation in RE, for every pair of tokens corresponding to the final token of entity e_i and e_j, a relation is $True$ if the tuple (e_i, e_j, r_k) exists, else it is 0. The loss L_{RE} is computed by $BinaryCrossEntropyLoss$. The total model loss is given by $L = L_{NER} + \lambda^r L_{RE}$, where λ^r is a hyperparameter and allows for tuning the relative importance of the tasks. Mini-batch size is set at 2, learning rate at 0.003. $\Theta^r = 0.9$

Hyperparameter	Value
BiRNN Type	GRU
Character BiRNN Size	32
Non-Character BiRNN Size	256
# Shared Layers	1
# NER-Specific Layers	1
# RE-Specific Layers	2
Label Embedding Size	25
Pre-BiRNN Dropouts	0.35
Pre-RE Scoring Dropout	0.5
RE Threshold, λ^r	5

Table 1: Hyperparameter values

4 Experiments

CoNLL04 (Roth and Yih, 2004) consists of 1,441 sentences from news articles annotated with four entity types *(Location, Organization, People, and Other)* and five relation types *(Works-For, Kill, Organization-Based-In, and Located-In*. The dataset has no overlapping entities. It was fetched from the scripts provided at SpERT Github Repo. There was although some preprocessing involved since the data is in JSON format with entity spans and not BIO tagging. The codebase is in *Python* with *Pytorch* GPU support (Paszke et al., 2019). Additional libraries are saved in *requirements.txt* in the code base.

To evaluate NER performance, a predicted entity is only considered a true positive if both the entity's span and span type are correctly predicted. In evaluating RE performance, a strict evaluation method is followed. It means that a relation is considered correct only if the spans corresponding to the two entities, entity types and the relation score is $\geq \Theta^r$. Hyperparameters for the model are listed in Table 1.

4.1 Results

To judge the performance of the model, precision, recall and f1 scores are calculated. Our implementation does decently well and is comparable to the results presented in the original paper, which shows that our paper implementation is correct, albeit small differences which can possibly be because of the averaged results by the author, small differences in architecture not mentioned in the paper and our less than optimized NER architecture. However, we achieve SOTA results on the Relation Extraction task in precision and f1 score. RE recall and NER scores are still higher in the span-based model provided in (Eberts and Ulges, 2019).

4.2 Extensions

Having completed the original architecture implementation, we tried to improve the results by replacing shared BiRNN layers with Transformer encoder. There are also some other variants which we tried, however those were not fruitful and are explained in roadblocks. This part of the project is done on a limited dataset i.e. randomly seeded **100** batches of training and **25** batches of testing.

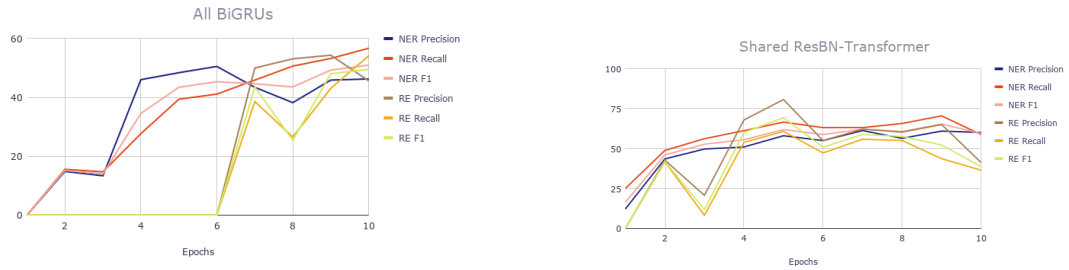
Before going through the results, let us first explain what ResBN-Transformer is. As can be perceived from the name it comprises of a transformer encoder (with positional encoding and layer-norm) followed a batch-norm. Then, the output is added with the input representation so that a residual connection is made. We did this to alleviate vanishing gradient problem and its inception arises from the roadblocks mentioned in the next section. Adding the residual component makes the model sensitive to the input representation and helps with back-

Model	NER			RE		
	Precision	Recall	F1	Precision	Recall	F1
(Eberts and Ulges, 2019)	85.78	**86.84**	86.25	74.75	**71.52**	72.87
(Crone, 2020)	**87.92**	86.42	**87.00**	77.73	68.38	72.63
Our implementation	84.11	82.39	83.24	**84.09**	70.31	**76.59**

Table 2: Performance Results: Actual paper results are averaged across multiple trials

Model	NER			RE		
	Precision	Recall	F1	Precision	Recall	F1
Baseline (BiGRUs)	46.289	56.71	50.973	45.614	54.167	49.524
Shared ResBN-Transformer	**61.049**	**70.563**	**65.462**	**65.116**	43.75	52.336
All ResBN-Transformer	60.515	61.038	60.775	62.5	**72.727**	**67.226**

Table 3: Extension performance results - results differ from presentation slides because the slides for the experiment section had evaluation calculation error



(a) Shared BiGRUs

Figure 2: Performance on extension parts

propagating the information in bigger networks.
As can be seen from 3 our new architecture *ResBN-Transformer* performs way better than the baseline of biGRUs. When it is used as a shared layer, it performs significantly better than the baseline in all metrics. Replacing all layers with *ResBN-Transformer* provides additional benefits especially for the RE part. Additionally, 2 shows that shared *ResBN-Transformer* starts learning from early stages of the epoch runs compared to biGRUs. It shows that transformers even with their simplicity outperform biRNN for this MTL model.

4.3 Roadblocks

- Shared + Task specific transformers: Intuitively, the first extension we could think of was to replace biRNN layers with transformers encoder layers with positional encoding and layer norm. However, on doing so, the outputs from the first shared layer itself were coming out so small, that while back-propagating no weights were being updated and we were ending up with vanishing gradient problems.

- Shared transformer with batch-normed outputs: The idea behind adding an unconventional batch-norm on the output of the transformer embeddings was to scale up the values, which would help with solving vanishing gradient problems. This was somewhat helpful as the model started learning, but the results were poor for the usual number of runs we performed our experiments.

- Not tested on ADE data: This is one difference between our implementation and the actual paper. The actual paper tests on ADE dataset too. However, due to high compute time, we did not run on ADE. The code base has all the codes to run the ADE dataset, so we preferred to do extensions into the model rather than running the same thing on another dataset.

5 Conclusion

In this project, we re-implemented a preprint which tackles NER and RE through MTL adds task-specific layers on top of shared layers leading to SOTA scores. Our re-implementation is comparable to the actual paper results and even beats it in RE task. Further, we extend the preprint by adding a variant of transformer encoder layers, which we refer to as *ResBN-Transformer* and observe that it has added benefits in terms of evaluation scores and surpasses the shared biRNN model in this specific setup. Additionally, the model starts learning the representations quicker than biRNNs.
As future work, we can look into some ablation study for the model architecture to analyze what part of model contributes the most to the overall architecture. We would also like to make use of this architecture to add

more closely related tasks to NER and RE such as co-reference resolution or entity matching etc. Finally, we would like to train and test *ResBN-Transformer* on the whole dataset, and not just the experiment.

6 Acknowledgement

This work is part of CS533:Natural Language Processing taken at Rutgers University, New Brunswick under the supervision of Prof. Karl Stratos. We are thankful to him in exposing us to NLP research and literature. Additionally, we appreciate all the advice provided by him and fellow classmates throughout our project.

References

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

R. Cipolla, Y. Gal, and A. Kendall. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7482–7491.

Phil Crone. 2020. Deeper Task-Specificity Improves Joint Entity and Relation Extraction. *arXiv e-prints*, page arXiv:2002.06424.

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.

Markus Eberts and Adrian Ulges. 2019. Span-based Joint Entity and Relation Extraction with Transformer Pre-training. *arXiv e-prints*, page arXiv:1909.07755.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547, Osaka, Japan. The COLING 2016 Organizing Committee.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 917–928, Vancouver, Canada. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Jianquan Li, Xiaokang Liu, Wenpeng Yin, Min Yang, and Liqun Ma. 2019a. An empirical evaluation of multi-task learning in deep neural networks for natural language processing. *ArXiv*, abs/1908.07820.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019b. Entity-relation extraction as multi-turn question answering. In *ACL*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks.

Simon Vandenhende, Bert De Brabandere, and Luc Van Gool. 2019. Branched multi-task networks: Deciding what layers to share. *ArXiv*, abs/1904.02920.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases.