Zero-shot Learning for Relation Extraction via Capsule Neural Networks

CSE 598: Introduction to Deep Learning
Final Project Report

Amrit Bhaskar	1222590486
Prutha Gaherwar	1222192413
Naveen Kumar	1219514114

1 Project Idea

We will concentrate on utilizing the novel architecture of zero-shot learning ability in a relationship extraction task as part of our Deep Learning project. The Relationship Extraction problem is related to the Information Extraction problem in that it involves detecting and classifying semantic relationship mentions within a set of artifacts, typically from a text corpus. Extracted relationships are those that exist between two or more entities of the same type (for example, people, organizations, and places) and fit into one of several semantic categories (e.g. married to, employed by, lives in). The fact that the labels are not finite and constrained limits the performance of the supervised classification model in relationship extraction tasks. As we collect more and more data, newer linkages and relationships emerge. With this resemblance in mind, we propose applying the concept of zero-shot intent detection to the relationship extraction problem. Being able to predict well even for examples not seen in training data could improve the performance of existing relationship extraction methods for ever-changing relationships between entities.
Link for the GitHub repository:
https://github.com/Amrit-Bhaskar-abhask10/CSE_598_Zero_Shot_Relation_Extraction

2 The Problem

2.1 Problem Definition

Apply zero-shot learning on the relation extraction problem to facilitate relationship extraction by bringing the representations of sentences that contain the identified entities closer together. Our aim is to predict the class of a given sentence and two marked entities. This is done for data that already has labels. Furthermore, we propose zero shot learning for emergent labels.

2.2 Impact Examples

The task consists of two parts:

1. Existing Relation Labels:
   In this first section, to anticipate the relation class, we'll use a supervised classification technique.

   If we are having a class "place of birth", the relation data sample would look like:
   <e1>Barack Obama</e1> was born in the<e2> United States</e2>
   Here, "Barack Obama" and "United States" are two marked entities.

2. Emerging Relation Labels:
   This is an unsupervised classification algorithm to predict the relation class. If we have a class "place of death" which is unseen in the trained model, we will still be able to predict using the zero-shot learning on the trained model.

   The sample would look like:
   <e1>Mahatma Gandhi</e1> was assassinated in <e2>India</e2>
   Here, "Mahatma Gandhi" and "India" are two marked entities.

3 Datasets

We will experiment with the NY10 dataset;

The widely used New York Times dataset (Riedel et al., 2010) contains 53 relation categories including a negative relation (NA) indicating no relation between two entities. We use the version of the data provided by the OpenNRE framework (Han et al., 2019), which removes overlapping pairs between train and test data. The dataset statistics are shown in Table 1. Additional information can be found in Appendix A.1. For the choice of the Knowledge Base, we use a subset of Freebase2 that includes 3 million entities with the most connections, similar to Xu and Barbosa (2019). For all pairs appearing in the test set of NYT10 (both positive and negative), we remove all links in the subset of Freebase to ensure that we will not memorise any relations between them (Weston et al., 2013). The resulting KB contains approximately 24 million triples.

4 Baselines and Evaluation Metrics

4.1 Proposed baseline

The state of art for the supervised relation extraction task can be found here: http://nlpprogress.com/english/relationship_extraction.html
As we are trying for zero shot learning on this task, there are some references for it: https://arxiv.org/pdf/2011.07126.pdf

However, in the zero shot reference mentioned above, the network used consists of CNN encoding. Here, we are proposing a capsule network architecture. Capsule network (https://pechyonkin.me/capsules-1/) has an added advantage over CNN in terms of its output as a vector rather than a scalar.

4.2 Results report with the baseline

The work on zero shot relation extraction is described in the paper (https://arxiv.org/pdf/2011.07126.pdf). However, the pipeline employs web scraping and the information that is not included in the dataset is added to the network explicitly to improve the performance. This can be challenging at times when the data used is specific to a topic area, such as the medical field. So, in order to avoid relying on external sources (web scraping), we will solely use the information included within the dataset. As a result, comparing baseline to this research is not fair. We shall, however, make every effort to obtain outcomes that are comparable to those in this paper.

4.3 Evaluation Metrics

- F1 score
- Accuracy

- Precision
- Recall

5 Experiments and Results

The sample we are using contains four components viz. relation, entity1, entity2 and sentences. The entities ‘entity1’ and ‘entity2’ are marked entities present in the sentence. Our model is trained on the seen relations present and is tested on the unseen relations. Seen relations are present in the training data while the unseen relations emerge after deployment and they do not have training data. The embedding of the seen relations is encouraged to have resemblance to the embedding of the unseen relations.

For example:
Seen relation = [‘place of death’]
Unseen relation = [‘place of birth’]

The architecture we have proposed is the capsule neural network described in this research (https://arxiv.org/pdf/1809.00385.pdf) paper which has leverage over CNNs. This architecture will be replicated with minor changes to the sentence representation to incorporate the position vector of the entities. A Capsule Neural Network (CapsNet) is a sort of neural network which could be used to effectively represent hierarchical relationships. The method is an attempt to emulate actual neuron architecture relatively accurately. The aim is to introduce structures known as “capsules" to a convolutional neural network (CNN) and utilize outputs from multiple of those capsules to construct more durable (in terms of various perturbations) representations for higher capsules. The result is a vector containing the probability of such an observation as well as the pose of such an observation. The task implemented in the reference study is intent detection. This task, however, will be replaced by relation extraction in our work. We had picked up the idea of zero shot learning from the intent detection. While intent detection, the task comprises classifying a sentence(intent) to a particular class(intent label), the relation extraction task isn't limited to the same. The entities present have a great influence over the classification.

For example:
Input sentence: The perfume containing rose is from France.
Here, there are two possible relations;
Entity1: perfume; Entity2: rose; Relation: contains
Entity1: perfume; Entity2: France; Relation: country

Reading the above example influences the model to classify this sentence to either of the two relations. To remove this ambiguity, we must instruct the model to classify the sentence based on the two marked entities. Thus, the position of two marked entities are the essential factor for the relation extraction task. Therefore, in addition to word representation(embedding), we need to add positional embedding of words with respect to each of the entities.

Thus, we will be doing two experiments:

1) Without positional embedding.
2) With positional embedding.

Riedel et al. (2010) publish the NYT10, a distantly-supervised dataset which takes the New York Times news as the source text. The raw file is a JSON file. The data is of the format below:


```
{'rel': '/location/location/contains',
 'sub_id': 'm.0ccvx',
 'obj_id': 'm.05gf08',
 'sub': 'queens',
 'obj': 'belle_harbor',
 'sent': 'Sen. Charles E. Schumer called on federal safety officials yesterday to reopen their invest
igation into the fatal crash of a passenger jet in Belle Harbor , Queens , because equipment failure
, not pilot error , might have been the cause .',
 'rsent': 'Sen. Charles E. Schumer called on federal safety officials yesterday to reopen their inves
tigation into the fatal crash of a passenger jet in Belle Harbor , Queens , because equipment failure
, not pilot error , might have been the cause .'}
```


Here, “rel” is the relation for this sample, “sub” is the first entity, “obj” is the second entity and “sent” is the sentence where these two entities are found and finally the relation is extracted. For our implementation, we processed this JSON into a text file with the following format.

<Relation> <Entity 1 position> <Entity 2 position> <Sentence>

For the above JSON, we processed the sample to the data below after removing the underscore(“_”) punctuation:

Contains queen belle harbor Sen. Charles E. Schumer called on federal safety officials yesterday to reopen their investigation into the fatal crash of a passenger jet in Belle Harbor , Queens , because equipment failure , not pilot error , might have been the cause .

Upon further processing, we got the final sample below:

Contains 26 23 Sen. Charles E. Schumer called on federal safety officials yesterday to reopen their investigation into the fatal crash of a passenger jet in Belle Harbor , Queens , because equipment failure , not pilot error , might have been the cause .
Here, 26 and 23 represent the index of words in the sentence.

We chose 7 existing relations and 2 emerging relations for our task.
Existing relations: ['contains', 'nationality', 'place lived', 'place of death', 'administrative divisions', 'neighborhood of', 'country']
Emerging relations: ['place of birth', 'capital']
The relations are chosen intuitively based on efficient zero shot knowledge transfer between "place of birth"-"place_of_death", and "country"-"capital", as the knowledge transfer is label embedding based.

For word representations, we used a 300 dimension vector for each word. We got the vector from pretrained "glove.840B.300d" embeddings. For position embeddings, we calculated the index based difference between each word with two other entities. Thus, we will get two integers representing the distance of each work with two other entities. We represent each of these integers with a 100 dimension vector using an embedding layer. Thus, we can say that each word is represented using a 500 size vector where 300 comprises word embedding and 200 for positional embedding. As we have referenced our task from Zero shot intent detection where there was no concept of entities, we had to include entities related information to classify the sentence to a particular relation.

We propose two architectures based on capsule models: RelationCAPSNET that is trained to discriminate among utterances with existing labels, e.g. existing relations for relation extraction; RelationCAPSNET-ZSL that gives zero-shot learning ability to RealtionCAPSNET for discriminating unseen labels, i.e. emerging realtions in this case. As shown in Figure 2, the cores of the proposed architectures are three types of capsules: SemanticCaps that extract interpretable semantic features from the utterance, DetectionCaps that aggregate semantic features for relation classification, and Zero-shot DetectionCaps which discriminate emerging relations.

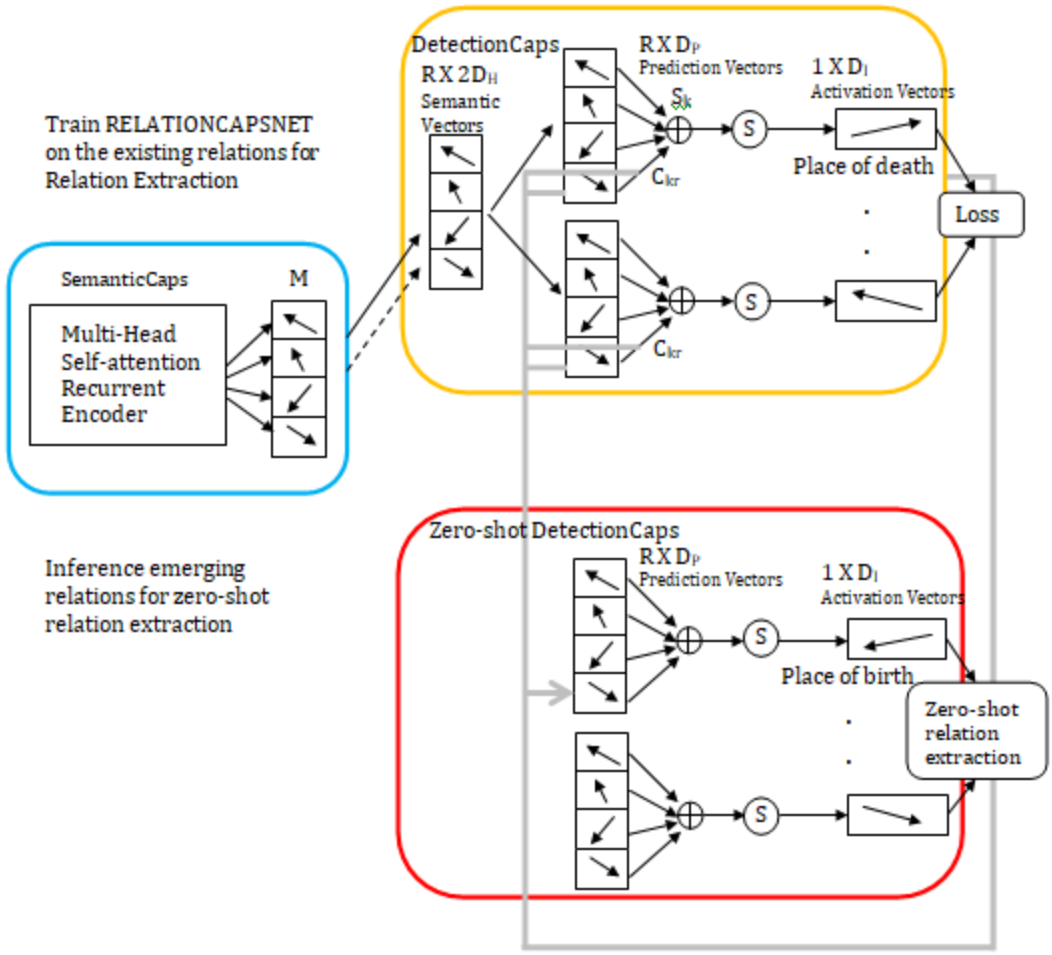


Figure 1: The architecture of RELATIONCAPSNET and RELATION CAPSNET-ZSL.

- **SemanticCaps:**

The SemanticCaps is based on a bi-direction recurrent neural network with multiple self-attention heads, where each self-attention head focuses on certain part of the utterance and extracts a semantic feature that may not be expressed by words in proximity.

Given an input utterance x = (w1, w2, ..., wT) of T words, each word is represented by a vector of dimension DW that can be pre-trained using a glove model. And concatenated with positional embeddings. A recurrent neural network such as a bidirectional LSTM is applied to sequentially encode the utterance into hidden states:

$$\overrightarrow{\mathbf{h}}_t = \mathrm{LSTM}_{fw}(\mathbf{w}_t, \overrightarrow{\mathbf{h}}_{t-1}),$$
$$\overleftarrow{\mathbf{h}}_t = \mathrm{LSTM}_{bw}(\mathbf{w}_t, \overleftarrow{\mathbf{h}}_{t+1}).$$

For each word wt, we concatenate each forward hidden state ht obtained from the forward LSTMfw with a backward hidden state ht from LSTMbw to obtain a hidden state ht for the word wt. The whole hidden state matrix can be defined as H = (h1, h2, ..., hT) ∈ R {T ×2DH} , where DH is the number of hidden units in each LSTM.

Inspired by the success of self-attention mechanisms (Vaswani et al., 2017) for sentence embedding, we adopt a multi-head self-attention framework where each self-attention head is encouraged to be attentive to a specific semantic feature of the utterance, such as certain sets of keywords or phrases in the utterance: one self attention may be attentive for the "born" action in "place of birth", while another one may be attentive to city name in "place of birth": it decides for itself what semantics to be attentive to. A self-attention weight matrix A is computed as:

$$\mathbf{A} = \mathrm{softmax}\left(\mathbf{W}_{s2}\tanh\left(\mathbf{W}_{s1}\mathbf{H}^T\right)\right),$$

where Ws1 ∈ R {DA×2DH} and Ws2 ∈ R {R×DA} are weight matrices for self attention. DA is the hidden unit number of self-attention and R is the number of self-attention heads. The softmax function makes sure for each self-attention head, the attentive scores on all the words sum to one. A total number of R semantic features are extracted from the input utterance, each from a separate self-attention head: M = AH, where M =(m1, m2, ..., mR) ∈ R {R×2DH} . Each mr is a 2DH−dimensional semantic vector.

- **DetectionCaps:**

The output of SemanticCaps are low-level vector representations of R different semantic features extracted from the utterances. To combine these features into higher-level representations, we build DetectionCaps that choose different semantic features dynamically so as to form a relation representation for each relation via an unsupervised routing by-agreement mechanism. As a semantic feature may contribute differently in detecting different relations, the DetectionCaps first encode semantic features with respect to each relation:

$$\mathbf{p}_{k|r} = \mathbf{m}_r \mathbf{W}_{k,r},$$

where k ∈ {1, 2, ..., K}, r ∈ {1, 2, ..., R}. Wk,r ∈ R{2DH×DP} is the weight matrix of the DetectionCaps, pk|r is the prediction vector of the rth semantic feature of an existing relation k, and DP is the dimension of the prediction vector.

❖ **Dynamic Routing-by-agreement:** The prediction vectors obtained from SemanticCaps route dynamically to DetectionCaps. The DetectionCaps computes a weighted sum over all prediction vectors.

$$\mathbf{s}_k = \sum_{r}^{R} c_{kr} \mathbf{p}_{k|r},$$

where ckr is the coupling coefficient that determines how informative, or how much contribution the r-th semantic feature is to the relation yk. ckr is calculated by an unsupervised, iterative dynamic routing-by-agreement algorithm (Sabouret al., 2017), which is briefly recalled in Algorithm

As shown in this algorithm, bkr is the initial logic representing the log prior probability that a SemanticCap r is coupled to a DetectionCap k.

❖ **Algorithm 1 Dynamic routing algorithm**

1: **procedure DYNAMIC ROUTING** (pk|r , iter)
2: for all semantic capsule r and relation capsule k: bkr ← 0.
3: for iter iterations do
4: for all SemanticCaps r: cr ← softmax (br)
5: for all DetectionCaps k: sk ← Σrckrpk|r
6: for all DetectionCaps k: vk = squash(sk)
7: for all SemanticCaps r and DetectionCaps k: bkr ← bkr + pk|r · vk
8: end for
9: Return vk
10: **end procedure**

The squashing function squash(·) is applied to sk to get an activation vector vk for each existing relation class k.

$$\mathbf{v}_k = \frac{\|\mathbf{s}_k\|^2}{1+\|\mathbf{s}_k\|^2}\frac{\mathbf{s}_k}{\|\mathbf{s}_k\|},$$

where the orientation of the activation vector vk represents relation properties while its norm indicates the activation probability. The dynamic routing-by-agreement mechanism assigns low ckr when there is inconsistency between pk|r and vk, which ensures the outputs of the SemanticCaps get sent to the appropriate subsequent DetectionCap.

❖ **Max-margin Loss for Existing Relations.**

The loss function considers both the max-margin loss on each labeled utterance, as well as a regularization term that encourages each self-attention head to be attentive to a different semantic feature of the utterance:

$$\begin{aligned}\mathcal{L} = \sum_{k=1}^{K}\{&[\![y = y_k]\!] \cdot \max(0, m^+ - \|\mathbf{v}_k\|)^2 \\ &+ \lambda\,[\![y \neq y_k]\!] \cdot \max(0, \|\mathbf{v}_k\| - m^-)^2\} \\ &+ \alpha||\mathbf{A}\mathbf{A}^T - I||_F^2,\end{aligned}$$

where [[]] is an indicator function, y is the ground truth relation label for the utterance x, λ is a downweighting coefficient, m+ and m− are margins. α is a non-negative trade-off coefficient that encourages the discrepancies among different attention heads.

- **Zero-shot DetectionCaps**

To detect emerging relations effectively, Zero-shot DetectionCaps are designed to transfer knowledge from existing relations to emerging relations.

❖ **Build Vote Vectors.**

As the routing information and the semantic extraction behavior are strongly coupled (ckr is calculated by pk|r iteratively in Line 4-6 of Algorithm 1) and their products are summarized to get the activation vector vk for relation k (Line 5-6 of Algorithm 1), we denote vectors before summation as vote vectors:

$$\mathbf{g}_{k,r} = c_{kr}\mathbf{p}_{k|r},$$

where gk,r is the r-th vote vector for an existing relation k.

❖ **Zero-shot Dynamic Routing.**

The zero-shot dynamic routing utilizes vote vectors from existing relations to build relation representations for emerging relations via a similarity metric between existing relations and emerging relations. Since there are K existing relations and L emerging relations, the similarities between existing and emerging relations form a matrix Q∈R L×K. Specifically, the similarity between an emerging relations zl∈Z and an existing relation yk∈Y is computed as:

$$q_{lk} = \frac{exp\left\{-d\left(\mathbf{e}_{z_l}, \mathbf{e}_{y_k}\right)\right\}}{\sum_{k=1}^{K} exp\left\{-d\left(\mathbf{e}_{z_l}, \mathbf{e}_{y_k}\right)\right\}},$$

where

$$d\left(\mathbf{e}_{z_l}, \mathbf{e}_{y_k}\right) = \left(\mathbf{e}_{z_l} - \mathbf{e}_{y_k}\right)^T \Sigma^{-1} \left(\mathbf{e}_{z_l} - \mathbf{e}_{y_k}\right).$$

ezl, eyk ∈ R {DI×1} are relation embeddings computed by the sum of word embeddings of the relation label. Σ models the correlations among relation embedding dimensions and we use Σ = σ2I. σ is a hyper-parameter for scaling. The prediction vectors for emerging relations are thus computed as:

$$\mathbf{u}_{l|r} = \sum_{k=1}^{K} q_{lk}\mathbf{g}_{k,r}.$$

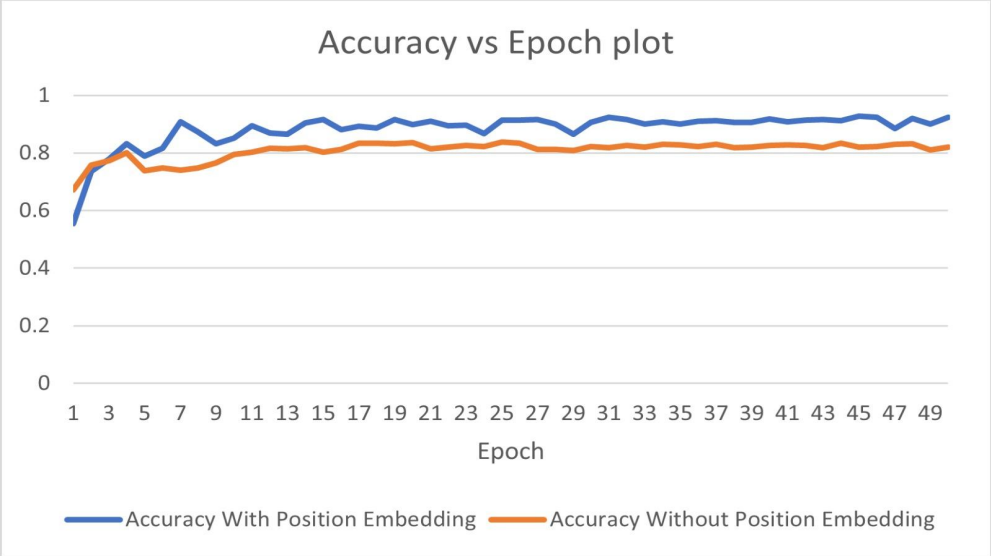
We feed the prediction vector nl to Algorithm 1 and derive activation vectors nl on emerging relations as the output. The final relation representation for each emerging relation is updated toward the direction where it coincides with representative votes vectors.

We can easily classify the utterance of emerging relation by choosing the activation vector with the largest norm

$$\hat{z} = \arg\max_{z_l \in Z} \|\mathbf{n}_l\|.$$

We tried our implementation with various hyperparameter tuning on batch size, learning and number of epochs. We got the optimal result using learning rate 0.001, batch size 64, and number of epochs 50. Learning rate greater than 0.001, we tried with 0.005, 0.01 and 0.05. The model started overfitting within a few epochs. Also, we tried with a higher batch size of 128 and the model started overfitting in a few epochs. The performance with batch size 32 is almost similar to batch size of 64, but we chose 64 to avoid more computation time. We decided on the number of epochs as 50 because the model had a similar performance metric for a range of epochs near 50.

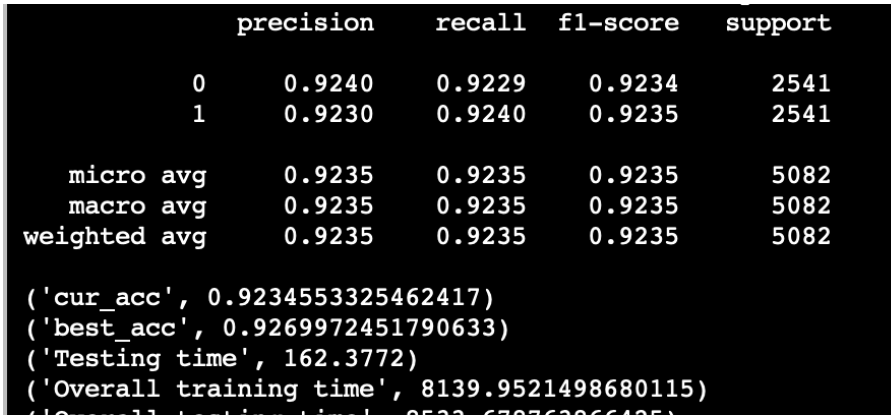
To show the improvement in performance with the addition of positional embedding, we compared our model with the addition of positional embedding and without the addition of positional embedding. We got an improvement of approximately 7% with the addition of position embedding. Intuitively, distance between words and entities is important in relation extraction tasks. The distance provides a sense of syntactic relation between them.



As we can see in the plot, initially when the positional embedding matrix isn't trained, we observe the “without positional embedding” model performs better. And after a few epochs when the positional embedding matrix is trained enough,

the model with “positional embedding” added performs better. The training time of the model with positional embedding added is more compared to the counterpart. This can be attributed to the computation time for the 500 dimension vector used when position embedding is added.

Finally, we report the performance metrics for 2 emerging relations ['place of birth', 'capital'] with positional embeddings:



	precision	recall	f1-score	support
0	0.9240	0.9229	0.9234	2541
1	0.9230	0.9240	0.9235	2541
micro avg	0.9235	0.9235	0.9235	5082
macro avg	0.9235	0.9235	0.9235	5082
weighted avg	0.9235	0.9235	0.9235	5082

```
('cur_acc', 0.9234553325462417)
('best_acc', 0.9269972451790633)
('Testing time', 162.3772)
('Overall training time', 8139.9521498680115)
```

6 Resources

1. Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, Philip S. Yu 2018. Zero-shot User Intent Detection via Capsule Neural Networks. In EMNLP 2018 as a long paper. https://arxiv.org/pdf/1809.00385.pdf

2. Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In Machine Learning and Knowledge Discovery in Databases, pages 148–163, Berlin, Heidelberg, Springer Berlin Heidelberg.

3. Xu Han, Tianyu Gao, Yuan Yao, Deming Ye, Zhiyuan Liu, and Maosong Sun. 2019. OpenNRE: An open and extensible toolkit for neural relation extraction. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations, pages 169–174, Hong Kong, China. Association for Computational Linguistics.

4. Peng Xu and Denilson Barbosa. 2019. Connecting language and knowledge with heterogeneous representations for neural relation extraction. 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 3201–3206, Minneapolis, Minnesota. Association for Computational Linguistics.

5. Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier 2013. Connecting language and knowledge bases with embedding models for relation extraction. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1366–1371, Seattle, Washington, USA. Association for Computational Linguistics.

6. Xu Han, Tianyu Gao, Yankai Lin, Hao Peng, Yaoliang Yang, Chaojun Xiao, Zhiyuan Liu, Peng Li, Jie Zhou, and Maosong Sun. 2020. More data, more relations, more context and more openness: A review and outlook for relation extraction. In Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 745–758, Suzhou, China. Association for Computational Linguistics.

7. Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2019b. Kepler: A unified model for knowledge embedding and pretrained language representation. arXiv preprint arXiv:1911.06136