Zero-Shot Learning in Named-Entity Recognition

Shuya Lin shuya@seas.upenn.edu
Jianrong Zhou zjr@seas.upenn.edu
Can Nie cnie@seas.upenn.edu
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA

Abstract

Zero-shot learning, a machine learning problem that requires a learner to predict which category a data point belongs to, with some categories not being observed during training, was first studied in the field of natural language text classification. Another frequently studied topic in natural language processing is Named Entity Recognition, a technique that identifies named entities in unstructured text into pre-defined categories such as Location, Person, or Time. This paper focuses on Zero-shot of Named Entity Recognition, trying to identify what phrases of text constitute a named entity and to classify it into 1 of 18 different types.

1 Introduction

The task of Named Entity Recognition (NER) has been widely discussed in the field of supervised machine learning, and it often assumes that the data are fully annotated. However, in real word, many data with limited annotations are present. For instance, some languages that are used by a very small group of people lack detailed annotations, and thus require the help of Zero-shot learning in the context of multi-class classification to identify entity types that are absent during the training phase.

Our project aims at providing a model that can classify Named Entity types even when some types are not shown in training data, given a collection of English tokens with annotations from OntoNotes 5.0 dataset. Each annotation consists of the token's B/I/O-tag followed by its NER type tag.

We explored different ways of addressing this problem, including trying out multinomial classification and Constrained Binary Learning, but we mainly followed the procedures describe in Zhou et al. (2018) to build a model that finds a set of related entities types to a given mention (token) in a sentence, and infers from these possible candidates. We also applied additional features to this model to obtain better performance.

2 Related Work

Named Entity Recognition (NER) typically includes entity-identifying and entity-typing. One work that is related to our specific task of addressing the problem of partially annotated dataset (Mayhew et al., 2019) introduces a way of using the Constrained Binary Learning (CBL) algorithm to train a weighted NER model and then apply variants of neural and non-neural NER models on it. We followed this approach in our early stages of separating entities from non-entities.

Despite token-level models like CBL, algorithms that approach the task of Named Entity Identification from the character level also exist. Adding simple features based on the Character-Level Language Model (CLM) implementation to an existing NER system also shows a great performance comparing with prevalent models (Yu et al., 2018).

The work of Zhou et al. (2018) provides some more significant insights on classifying entity types that were not seen in training. Zoe, a zero-shot open entity typing framework that relies on type taxonomies to infer a given token's type, shows a much more powerful performance comparing with state-of-art models, and requires no training on supervised data. It is noticeable that the use of type definition instead of full annotation makes it easier for real life application, and refrains users from retraining the model.

3 Methods

3.1 Multinomial Classification

Given the problem definition to assign predefined entity types to tokens, we decided to first resort to normal multinomial classification approaches. However, since in zero-shot learning, some of the actual entity types are not included in the training set, to include such types in the process, we designed a two-level pipeline to label a token with B/I/O-tag before we calculate its similarity to the definition vector of each predefined entity type.

The B/I/O-prefix tagging was experimented with several different classifiers, while the similarity between tokens and entity labels was defined by the euclidean distance between their embedding vectors. For each entity type, we provide an approximation of its human definition with a list of similar words or sub-types, and represent it with the average word embedding.

Each token was classified into B/I/O classes. For the B/I-tag tokens, we assign them to the closest entity type in terms of euclidean distance and cosine similarity, respectively, to compare their testing performance.

3.2 CBL

Another experiment we did for classifying entities utilizes CBL. We first separated our data into 2 classes by converting the B/I tags into N, letting the O's remain O so that we can use a binary classifier to distinguish the non-entities from the entities. Because our data is partially annotated, some of the O's in the dataset might actually be N. The existence of false negatives implies that our data is noisy, so we want to modify our original training data to avoid getting a noisy classifier.

In order to distinguish true negatives from false negatives, we followed the procedure described in Mayhew et al. (2019) and used a constraint-driven iterative algorithm. This algorithm assigns weight to each instance to indicate how likely it is correctly labeled and modifies their weights according to constraints, allowing us to rebuild a weighted training set and to train a weighted NER model using either a non-neural model and a neural model.

Since we trust the data points that are already labeled as entities, we assign a weight of 1 to each of them and do not attempt to change their weights during the iteration. So we focus only on the instances that are labeled as O. In the end, the false negatives should have very small weights and the others should have big weights. This is how we differentiate the true and false negatives to find out entities.

The algorithm mainly includes 2 phases of training:

phase1: iteratively train on the training set, then predict on it and use our constraints to correct the prediction and update weights and labels on instances.

phase2: use the binary classifier trained in phase1 and assign weights to instances.

After these 2 phases, we train a standard weighted NER classifier with the **original** dataset with the final weights we obtained.

3.3 ZOE

We also worked to implement the Zero-Shot Open Entity Typing system (ZOE) proposed by Zhou et al.(2018), which supports zero-shot open entity typing without a training process. It tries to discover a set of type-compatible entities for a given token in a given context, then rank these potential types based on their context consistency to infer the most possible type for the token.

4 Experiments

Dataset The dataset we used for our experiment comes from OntoNote 5.0, and only the English annotations for NER are considered. We have datasets with 0, 6, 12 and 18 annotated labels. Each is then separated into train, develop, and test sets.

The fully annotated version has 18 labels, which are: PERSON, ORGANIZATION, GPE, EVENT, NORP, TIME, FACILITY, LAW, PERCENT, QUANTITY, WORK OF ART, PRODUCT, DATE, MONEY, ORDINAL, CARDINAL, LANGUAGE, and LOCATION.

4.1 Multinomial Classification

Multiple different setups for feature representation were experimented. First of all, we used pre-trained GloVe embedding with a dimension of 50 and 300 respectively to obtain vector representations for tokens (Pennington et al., 2014). Secondly, as a trial to improve model performance by incorporating order information, we had another setting that includes a $first\_capital$ feature and a $position$ feature. The $first\_capital$ feature records whether the first letter of the token is capitalized. Our intuition was that this feature is crucial from multiple aspects: it can help determine whether the token is at the beginning of a sentence, or whether the token is part of a proper noun like "Chongqing" the city name. The *position* feature, on the other hand, directly records the position of the token in its sentence.

For each feature representation approach, we implemented several different models. The first one is a multinomial logistic regression model. Using the default initialization with a lbfgs solver and L2 penalty, due to the large size of our training set, it failed to converge until we double the maximum iteration count. Hence, we decided to switch to the saga solver which converges faster and works better with larget datasets. We also tried to tune the different penalty configurations including L1, L2, and ElasticNet with a L1 ratio from 0.1 to 0.9, but these configurations had little impact on the test performance of our models. The second model is a SVM model using the LinearSVC implementation in scikit-learn, since it was suggested to work better with a larger dataset. Here, we use the default model except that we config it to run in primal mode instead of dual mode since the size of our dataset is much larger than our feature dimensions. The third model comes with a neural network approach, where we decided to go with a LSTM model since the meaning of a token might have some long-range dependencies on some important previous words, while LSTM can selectively forget some irrelevant token information to maintain relevant long-term relationships.

Other than these three models, in order to achieve better interpretability, we also worked with explainable decision tree models of different height limitations, as well as random forest model which reduces the overfitting problem and may improve the performance by averaging multiple models.

After the classification phase which tags each token with a B/I/O-tag, we then experimented on different approaches to map these tokens onto the specific entity type. Since the goal is to conduct a zero-shot learning where no training labels of entity types are given, our intuition was to enable a manual specification of the definition of wanted entity types. This was achieved by a mapping from label names to definitive words and their example sub-types. For instance, the definition of "FAC" type was a list of words: facility, building, airport, highway or bridge. We used the corresponding average word embedding of all definition words to represent each entity type as described in Chen (2017).

Then, to get the similarity between a token and all pre-defined entity types, we tried two common approaches - Euclidean distance and cosine similarity. For every token with a B/I-tag, we assign it to the most similar entity label with a minimum Euclidean distance or a maximum cosine similarity.

4.2 CBL

To better identify named entities, we adopted codes from the source code provided by Mayhew et al. (2019) and implemented on our test data.[1] Specifically, instead of initializing weights of all negative points to be 1, we added a weighting scheme to boost our performance. This is based on the intuition that tokens like "several" are frequently used non-entities, so they should have a weight of 1, but tokens like "Kasprov" that are more likely to be entities, should have a lower weight. We then combined 2 basic weightings into one: the first one is frequency based, which takes the fact that commonly appeared tokens are rarely named entities into consideration. The other is window based, since named entities do not tend to appear immediately next to each other. Since the format of data required by this CBL algorithm follows the TextAnnotation format, as specified in *ccg_nlpy*, we also converted our data using the Cogcomp NLP Pipeline.[2]

We then experimented with variants of both Cogcomp and BiLSTM-CRF, two commonly used models in NER. The Cogcomp model is modified to use Weighted Averaged Perceptron, in order to adjust with our weights. The only difference is that we multiply the learning rate by the weight of this example when updating the weight. The modified BiLSTM-CRF also adjusts to the weight of each instance in the following way: when weight of an instance is 0, the soft labeling weights over this instance is set to be uniform, since this low weight should not update the model during training. Remaining probability mass of this one will be evenly distributed to other labels.

We ran the algorithm on our train, dev, and test data with 6 and 12 labels. Due to the size of the

[1] https://github.com/mayhewsw/ner-with-partial-annotations

[2] https://github.com/CogComp/cogcomp-nlpy

data and the long time taken for the pipeline to get NER OntoNotes' view, we only processed the first 50000 tokens in our dataset.

However, we run into a problem that the overall-f1, recall, and precision scores are all zeroes, although accuracy was above 0.5. Since the accuracy is non-zero, the number of true positives should be non-zero. We tried to experiment with the original GitHub codes and data for multiple times, but all ended up getting the same disappointing results and no online solution worked. Therefore, we decided to move to other approaches and switched our focus to zero-shot learning.

4.3 ZOE

In order to make use of the ZOE system[3] for our project, several adaptations had to made. First of all, the input format of ZOE is a JSON file where each line is a list of tokens in a sentence, together with the ground-truth entity type, start and end indices of each named entity in this sentence. Hence, we first implemented a function to parse our data format into the required compatible format. Moreover, we looked at the included examples in the GitHub repository and found that the BBN dataset is having a similar entity time format as our dataset, so we constructed a mapping between these two and utilized the cached embeddings for the BBN dataset to conduct a prediction on ours.

Another modification we implemented was a different evaluation metric. In the original ZOE system, only a strict accuracy was calculated where the predicted sets of labels must be exactly the same as the sets of gold labels. This was a great measurement for multi-label datasets used for the system, however, for our dataset which is simpler and contains only one gold label for each entity, this exact matching is not required since most predicted set of labels are having a cardinality larger than 1, and it is almost impossible that they will be the same. Also, we looked at the predictions produced by ZOE and figured that it always gives a larger range entity type label together with a subclass of this type, whereas, it is conducting a more detailed task that is not actually required in our problem formulation. As a result, to adapt this multi-label system to our single-label dataset, we defined a relaxed accuracy where the prediction is considered correct as long as it contains our ground-truth label.

[3]https://github.com/CogComp/zoe

5 Evaluation and Analysis

5.1 Multinomial Classification

The results of our experiment of 5 different classifiers with 4 feature representation setting is shown in Table 1. Among all feature representations, the pre-trained GloVe embedding with 300 dimensions has the best and most stable performance. In terms of different models, we can see that logistic regression and SVM models are having similar performance across all criteria, but the convergence time required for SVM is significantly longer than logistic regression. Other two model-based classification approaches - decision tree and random forest also return similar results. The best one among all is our neural approach with LSTM, producing an F1 score of 53, and leading in almost all metrics except for the recall. Hence, we used the prediction results of this model to conduct further entity typing.

For the final process of entity typing, we used two different approaches to define the similarity between word vectors, the results are listed in Table 3. The first and third rows are score metrics calculated based on the entire B/I/O format typing (e.g. B-CARDINAL), while the second and fourth rows are calculated based on only the detailed entity type (e.g. CARDINAL). Our pipeline is not having a good performance, and this could be due to the low recall for both B and I classes in our trained classifiers (as shown in Table 2), or it could be because our definitions for the labels are not comprehensive enough. Despite the low performance, it is obvious that cosine similarity is a much better definition for similarity here compared with the euclidean distance.

We also tried to train several binary classifiers by combining the B and I classes into a N class, while keeping the O class as it is. As exhibited in Table 1, the performance of these binary classifiers are significantly better than the multi-class models, hence, a possible future exploration is to use these classifiers for the last step in our pipeline. This is gonna be a simpler but similar approach to the above-mentioned CBL approach raised by Mayhew et al. (2019).

Table 1: Macro metrics for B/I/O (or N/O) classification

Model	Precision	Recall	F1	Accuracy
glove50d, LR(multinomial, saga)	65	41	43	87
glove50d, SVM(ovr, linear)	68	40	42	87
glove50d, LSTM(multinomial)	77	46	52	88
glove50d + features, LR(multinomial, saga)	67	41	44	87
glove50d + features, SVM(ovr, linear)	70	40	43	87
glove50d + features, LSTM(multinomial)	79	46	52	88
glove300d, LR(multinomial, saga)	76	45	51	88
glove300d, SVM(ovr, linear)	75	45	50	88
glove300d, LSTM(multinomial)	**80**	**47**	**53**	**88**
glove300d, Decision Tree(multinomial)	78	47	53	88
glove300d, Random Forest(multinomial)	79	47	53	88
glove300d + features, LR(multinomial, saga)	76	45	50	88
glove300d + features, LSTM(multinomial)	77	47	53	88
glove300d, LR(binary, saga)	81	87	84	91
glove300d, SVM(binary, linear)	81	87	84	91
glove300d, LSTM(binary)	**82**	**88**	**84**	**92**
glove300d+features, LR(binary, saga)	82	86	84	91
glove300d+features, SVM(binary, linear)	81	86	84	91
glove300d+features, LSTM(binary)	82	86	84	92

Table 2: Detailed metrics for our chosen classifier

Similarity Metric	Precision	Recall	F1
B	65	19	29
I	85	22	35
O	89	100	94
Macro	80	47	53

5.2 ZOE

The result produced by our modified ZOE system is recorded in Table 4, demonstrating a much better performance than our proposed multinomial classification pipeline with similarity assignment, despite its more strict evaluation metrics. An example among the predicted outcome is: for the sentence ”You will notice that Michael’s real troubles started after he sued Sony chief Tommy Mattola, who was heard to say ’he’ll live to regret the day’.”, the predicted entity type is PERSON for ”Michael” and ORGANIZATION, ORGANIZATION/CORPORATION for ”Sony”, which is very accurate and similar to our human perception.

6 Conclusion

In our project, we approached the task of zero-shot learning in NER from 3 different ways. The first approach uses the pipeline to intuitively train a simple classification model and map the tokens accordingly to different entity types. It is not showing a very promising result as shown in our report. We then considered Constraint Binary Learning for entity identifying. This is implemented in an iterative way with weights assigned to each instance. Although we were unable to finish with this method, the intuitions embedded in resonate with what we studies in class and are insightful for better understanding the challenges in this task.

The best model to accomplish our goal for zero-shot learning was demonstrated to be the third approach, which is adapted from the Zero-shot Open Entity Typing platform designed by Zhou et al. (2018). ZOE is quite easy to be applied widely in real life, since it requires no fully annotated data nor retraining of data. By mapping a given mention to a set of type-compatible Wikipedia en-

Table 3: Macro metrics for similarity-based entity typing

Similarity Metric	**Precision**	**Recall**	**F1**	**Accuracy**
Euclidean Distance(with B/I/O tag)	10	7	6	86
Euclidean Distance(only entity type)	14	10	9	86
Cosine Similarity(with B/I/O tag)	15	10	9	86
Cosine Similarity(only entity type)	22	14	12	86

Table 4: Macro metrics for ZOE

Precision	**Recall**	**F1**	**Accuracy**
55.2	86.9	67.5	86.9

tries and relying on type definition, we are able to infer the most possible entity type for a mention. It is also flexible in the way that it maximizes the utilization of information but is not constrained by existing information. To us, the various approaches, theories and online source codes we studied during this exploratory process was edifying and meaningful.

7 Future Work

So far, adding features seems to have little improvement on our Multinomial Classification models. We plan to add a few more features in the future. For example, we can add a $after\_preposition$ feature to indicate whether a token is immediately after a preposition. Also, a more explainable and comprehensive definition of target types might further enhance the workflow. Moreover, a binary classification could potentially improve the performance of our proposed classification pipeline.

For CBL, we have successfully generated the formatted files for train, validation, and test datasets, using open GitHub source *ccg_nlpy* remotepipeline. However, we encountered problems like getting zero f1 scores. In the future, we hope to continue seeking solutions to these problems, maybe by approaching authors of these codes and experts in this filed to resume our the experiment in CBL. If it works, we will also consider adding extra constraints like capitalization to further boost our model.

Regarding our adaptation from ZOE, potential enhancement include a more comprehensive use of the system to generate a better embeddings from wikilinks and the dataset instead of reusing the cached ones for the BBN dataset.

8 Supplementary Documents

Video Presentation:
`https://drive.google.com/file/d/1tM-nckOViL8KuFk_drO1TjR5WCV8Mnw6/view?usp=sharing`
Code:
`https://drive.google.com/drive/folders/1wWsvjkXcKx1my5r2FT9Dz6dCdAiklbzV?usp=sharing`

References

Minmin Chen. 2017. Efficient vector representation for documents through corruption. In *Proceedings of the 2017 International Conference on Learning Representations (ICLR)*.

Stephen Mayhew, Snigdha Chaturvedi, Chen-Tse Tsai, and Dan Roth. 2019. Named Entity Recognition with Partially Annotated Training Data. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Xiaodong Yu, Stephen Mayhew, Mark Sammons, and Dan Roth. 2018. On the strength of character language models for multilingual named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3073–3077.

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2018. Zero-shot open entity typing as type-compatible grounding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2065–2076.