NAIST Participation in the TAC KBP 2016 Cold Start Slot Filling Task: Combining CNN-based and Bootstrapping-based Methods

Kodai Sudo Akihiko Kato Van-Thuy Phi[*] Hiroyuki Shindo Yuji Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
{sudo.kodai.rx2, kato.akihiko.ju6,
phi.thuy.ph8, shindo, matsu}@is.naist.jp

Abstract

Nara Institute of Science and Technology (NAIST) participated in the English Slot Filling Task in TAC-KBP 2016. Our system consists of two stages: (1) *Candidate Generation stage*, and (2) *Candidate Validation stage*. In the candidate generation stage, we retrieve documents relevant to a given query and select candidates based on the *RelationFactory* system. We also add more constraints to deal with some new slots this year. In the candidate validation stage, we combine results from two separate classifiers, namely CNN-based classifier and bootstrapping-based classifier.

1 Introduction

The main goal of the Slot Filling task in TAC-KBP 2016 is to extract relational information about query entities from a large text corpus. Our proposed system consists of two stages: *Candidate Generation Stage* and *Candidate Validation Stage*. In the candidate generation stage, we retrieve documents relevant to a given query and select candidates by using the *RelationFactory* system (Roth et al., 2014). In the candidate validation stage, we select top-*k* candidates in terms of confidence scores by combining results from two separate classifiers: one CNN-based classifier and one bootstrapping-based classifier. Finally, our system returns values for predefined slots (attributes) for given entities.

[*]The first three authors contributed equally to this work.

2 Dataset

2.1 Resource for CNN-based Classifier

We use Angeli's dataset (Angeli et al., 2014) as the training data for the CNN-based classifier. The dataset is built by using Amazon Mechanical Turk to crowdsourcing annotations. Angeli et al. collect 5 annotations for each sentence, and use the most commonly agreed answer as the ground truth. A total of 23,725 examples were annotated, and the dataset includes confidence scores acquired by aggregating annotations.

2.2 Resource for Bootstrapping-based Classifier

For the bootstrapping-based classifier, we use the simple high-precision patterns[1] for relations used in the *RelationFactory* system to learn reliable instances for each slot type. These new instances are acquired automatically in a bootstrapping process described in Section 4.2.2. We keep them in the seed set for each slot/relation type.

We use *ReVerb Extractions 1.1* as the main dataset to acquire more instances given the seed set. *ReVerb* (Fader et al., 2011) is a program that automatically identifies and extracts binary relationships from English sentences, where the target relations cannot be specified in advance. It contains a set of *(x, r, y)* extraction triples of binary relations, for example, *(bananas, be source of, potassium)*.

A collection of 15 million high-precision ReVerb

[1]https://github.com/beroth/relationfactory/blob/master/resources/manual_annotation/context_patterns2012.txt

extractions is available for academic use[2]. *ReVerb Extractions 1.1 dataset* is the result of running ReVerb on the ClueWeb09 dataset and a portion of English Wikipedia. The following statistics are the number of distinct tuples, argument strings, and relation strings in the data set:

- Tuples: 14,728,268.
- Argument Strings: 2,263,915.
- Relation Strings: 664,746.

3 Baseline System

We adopt the *RelationFactory* system (Roth et al., 2014) as the baseline system. It includes the candidate generation stage and the candidate validation stage. In the candidate generation stage, documents relevant to a given query are retrieved by using the original query name and the query expansion. Then, a sequence of named-entity tags is predicted for each retrieved sentence by the NER-tagger. Finally, sentences that include entities, whose named-entity types are consistent with the given slot, are passed to the next stage. In the candidate validation stage, the *RelationFactory* system examines whether each candidate actually realizes the relation or not by using patterns/rules or machine learning classifiers.

4 System Architecture

Our system architecture are illustrated in Figure 1. It includes the candidate generation stage, and the candidate validation stage. We describe each stage in Section 4.1 and Section 4.2.

4.1 Candidate Generation Stage

As mentioned earlier, we used the *RelationFactory* system as the candidate generator. Moreover, we use *GeoLite2*, an external geographical dataset, to filter out noise candidates related to "*city*", "*stateorprovinces*" or "*country*". GeoLite2 is a location dictionary, which includes *country names*, *subdivision names* and *city names*. We treat "*stateorprovinces*" in the Slot Filling task as the *subdivision* field in GeoLite2 dataset. In the following candidate sentence for the "*per:country_of_birth*" relation: "*Barack Obama was born in Hawaii*", the query entity is "*Barack Obama*" and the potential slot filler is "*Hawaii*". However, "*Hawaii*" is not in the country name list, therefore that candidate is inappropriate.

4.2 Candidate Validation Stage

For each query in the candidate validation stage, we combine outputs from two classifiers, namely CNN-based classifier and bootstrapping-based classifier.

4.2.1 CNN-based Classifier

Piecewise Convolutional Neural Network (PCNN) (Zeng et al., 2015) is a variation of CNN, which adopts a piecewise max pooling. In traditional CNN, max pooling operation is often utilized to capture the most significant features in each feature map. Nonetheless, this idea is insufficient for relation extraction. Piecewise max pooling operation can get more fine-grained features than single max pooling operation. In relation extraction, an input sentence can be divided into three segments based on positions of entities. Therefore, We use the PCNN as our relation classifier.

4.2.2 Bootstrapping-based Classifier

We consider slots in the Slot Filling task as patterns (or relations), and entities as instances in the binary relation extraction task, then we use *Espresso+Word2vec* system introduced by Phi and Matsumoto (2016) to induce more reliable pairs of entities for each type of slot. For the part-whole relation extraction task that deals with 8 fine-grained subtypes, the Espresso+Word2vec system achieved a precision of 84.9% for harvesting instances, and outperformed the original Espresso system (Pantel and Pennacchiotti, 2006). The Espresso+Word2vec system utilizes an additional ranker component, namely *Similarity Ranker*, which uses embedding offset information between instance pairs of specific relations. For each new instance, that ranker calculates the average similarity score between this instance and initial instances in the seed set. The similarity score of an instance i,$SIM(i)$, is defined as:

$$SIM(i) = \frac{\sum_{j \in I_{Previous}} Cos\_sim(i,j)}{|I_{Previous}|}$$

where $Cos\_sim(i,j)$ is the cosine similarity between two instances, and $I_{Previous}$ are instances in the seed set.

[2] http://reverb.cs.washington.edu

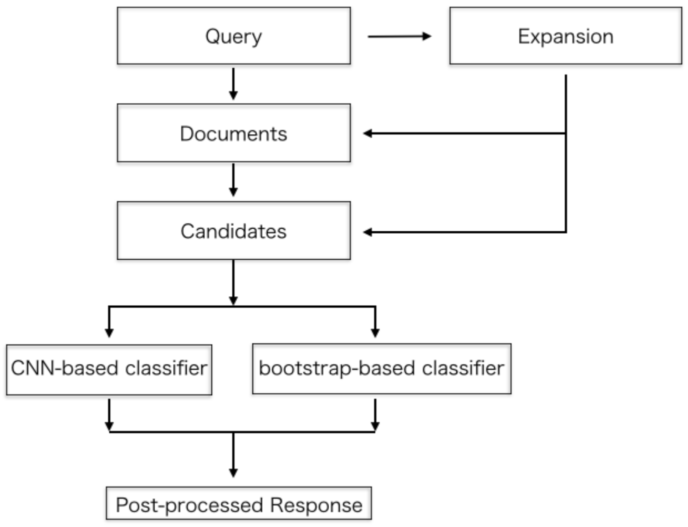

Figure 1: Our overall architecture for the Slot Filling task

Finally, instances are ranked by their similarity score to old instances. The bootstrapping-based classifier discards all but the *top-k* instances as the return results.

5 Experimental Results and Discussion

5.1 Submitted Runs

We show experimental results on the test set in terms of LDC-MAX-micro in Table 1. Our submitted runs are characterized as follows:

Run1 (Recall): We deal with both hop0 and hop1 queries without filtering step.

Run2 (Fast): We deal only with hop0 queries.

Run3 (Precision): Similar to Run1, however, we remove candidates that have confidence scores below our threshold (0.55) for hop1 queries.

For Run3, our precision/recall/F1 scores for hop1 queries are zero because of the high threshold. In terms of run1, the recall for hop1 queries are higher than 0.09. On the contrary, the precision for hop1 queries are less than 0.01, since we did not filter any candidates. This causes lower precisions for all queries in other runs.

5.2 Results for Development Set

We also evaluated our system on the development set (a dataset for TAC-KBP 2013 English Slot Filling task). Because of time constraints, we could not conduct experiments for the combination of two classifiers. Thus, we conducted experiments only for the CNN-based classifier. We show our experimental results and the results for the baseline system(*RelationFactory*) in Table 2[3].

The system we evaluated are:

System (Recall): We removed candidates for queries that have confidence scores below our threshold (0.01) to get the high recall.

System (Precision): We removed candidates for queries that have confidence scores below our threshold (0.6) to get the high precision.

6 Conclusion

This year, we participated in the English Slot Filling task in TAC-KBP 2016. Our system consists of a candidate generation stage and a candidate validation stage. In the candidate generation stage, we

[3]As you can see, the F1 score of our system is much lower than the F1 score of the baseline system. By checking our implementation after the submission, we found that our CNN based classifier had a small bug.

	Hop0			Hop1			All		
Run ID	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Run1	0.0868	0.0376	0.0525	0.0032	0.0974	0.0061	0.0054	0.0576	0.0099
Run2	0.0858	0.0376	0.0523	0.0000	0.0000	0.0000	0.0858	0.0250	0.0387
Run3	0.0861	0.0376	0.0523	0.0000	0.0000	0.0000	0.0801	0.0250	0.0381

Table 1: Experimental results on the test set in terms of LDC-MAX-micro

System	**Precision**	**Recall**	**F1**
Our system(Recall)	0.1213	0.0643	0.0840
Our system(Precision)	0.1430	0.0589	0.0834
Baseline(*RelationFactory*)	0.3276	0.4513	0.3796

Table 2: Experimental results on the development set (official score)

retrieve documents relevant to a given query and select candidates based on the *RelationFactory* system. In the candidate validation stage, we combine results from 2 classifiers, namely CNN-based and bootstrapping-based classifiers. In the future work, we plan to train one CNN-based binary classifier for each kind of relation instead of our current multi-class classifier. This could lead the performance gain for relation classification task. Second, we would like to examine the performance when combining the bootstrapping-based classifier and CNN-based classifier on the development set. Finally, we would like to explore the better way to combine our two classifiers.

References

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2014. *Effective slot filling based on shallow distant supervision methods.*, volume 1. arXiv preprint arXiv:1401.1158.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. *Distant supervision for relation extraction via piecewise convolutional neural networks.* Proceedings of EMNLP.

Fader, Anthony, Stephen Soderland, and Oren Etzioni. 2011. *Identifying relations for open information extraction.* Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

Pantel, Patrick, and Marco Pennacchiotti. 2006. *Espresso: Leveraging generic patterns for automatically harvesting semantic relations.* Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL.

Van-Thuy Phi, and Yuji Matsumoto. 2016. *Integrating Word Embedding Offsets into the Espresso System for Part-Whole Relation Extraction* Proceedings of The 30th Pacific Asia Conference on Language, Information and Computation (PACLIC).