Data Processing Techniques for the TAC-KBP 2019 EDL Task

Qihui Feng, Weiyue Wang, Evgeniia Tokarchuk, Christian Dugast, Hermann Ney

Human Language Technology and Pattern Recognition, Computer Science Department
RWTH Aachen University, 52056 Aachen, Germany

1. Data Analysis

The following corpora were used in our systems:

- OntoNotes [1]
- TAC EDL 2015-2017 [2, 3]
- RPI silver-standard annotation from Wikipedia and YAGO [4, 5]
- AIDA Seedling Corpora (LDC2018E01, LDC2018E52, LDC2018E53) annotated by the community effort

Although these corpora have been annotated using different ontologies, we combine them using different data processing strategies in order to generate a training corpus that was mapped to the AIDA ontology. We firstly list the specificity of each corpus before going further describing the methodology used to select data and to homogenize annotations.

1.1. RPI silver-standard annotation from Wikipedia and YAGO

This corpus contains more than 100 million sentences and is labeled with more than 7.3 thousand entity types. As this corpus is by far the largest from all corpora available to us, we mainly use this corpus to generate our training and development sets.

The difficulties in using this corpus are:

- Low annotation coverage: there is a large set of missed annotated mentions, whose volume seems to be significant.
- Type ambiguity: a mention is annotated with on average 7.6 YAGO entity types
- Existence of irrational mentions: words like a single conjunction `and` or a single article `the` can be annotated

Therefore, the main work for this corpus is to reduce the ambiguity of entity types or mention extents, and to filter out the sentences with irrational mentions.

1.2. TAC EDL 2015-2017 and OntoNotes

Compared to the YAGO data set, the TAC and OntoNotes data sets are much smaller. Each training set contains less than one hundred thousand sentences, but they are cleaner and contain no nested entities nor mentions of several entity types. A good reason to make use of the TAC corpus is that the five entity types in TAC EDL 2015-2017 (`PER`, `ORG`, `GPE`, `LOC` and `FAC`) match five of the first level entity types in AIDA, our target annotation system. OntoNotes uses another ontology and contains 18 entity types. In our previous experiments, however, we found that incorporating re-tagged OntoNotes data into the TAC EDL 2015 training data improves the system performance on the TAC EDL 2017 test set. Thus, we use these corpora as a seed for training a model on TAC ontology, and apply this model to generate synthetic annotations on sentences from YAGO.

1.3. AIDA Seedling Corpora

These corpora are manually annotated by community members and supposed to be clean. However, they contain only 1533 sentences that have at least one annotated mention. Thus, they could hardly be used for independent training. We use these corpora first to evaluate the quality of our system, and in our final submission they are used to fine-tune our system (domain adaptation). Alternatively they are up-sampled and then concatenated to our filtered training data.

For simplicity, from now on corpora in Section 1.1, 1.2 and 1.3 are referred to as YAGO, TAC and AIDA data, respectively.

2. Data Processing and Filtering

This section describes the steps that are used to convert the original YAGO data into the training data used for our final submission. The aim of the data processing is to eliminate the annotation inconsistencies and the formatting issues and to improve the annotation accuracy of the training data by re-tagging and filtering.

2.1. Ontology Transformation

First, the data must be converted from YAGO to AIDA ontology. The YAGO entity types specified in the partial mapping are converted to the corresponding AIDA types, while other types are deleted. Then we also remove all sentences that, according to AIDA ontology, contain no mentions.

2.2. Sentence-level Merging of YAGO Corpus

We notice that many YAGO sentences appear several times in the corpus, each time with an additional annotation not seen in the other identical sentences. To address this problem, our first step is to merge identical sentences and their entities. A test on a subset of 1 million randomly selected sentences from YAGO data shows that after the merge, the total number of entities on this subset increases by 32% (from 2.15 millions to 2.83 millions).

After step 2.1 and step 2.2, the corpus consists of 29.1 million sentences.

2.3. Elimination of Nested Mentions

Our goal is to obtain a training set without nested mentions and each mention should be labeled with only a single entity type. However, mentions in the YAGO corpus are originally multi-type (7.6 YAGO types per mention, and 2.3 AIDA types per mention). Besides, 9.9 thousands sentences in the corpus after step 2.2 contains nested mentions. The proportion of these sentences is low, but formatting issues still need to be eliminated.

We use the following rules to resolve the overlap of mention extents: For each sentence, we keep a list of mention extents and initialize the list as empty. The mention candidates are sorted by length. We start from the longest candidate and add the mention extent to the list if it does not overlap with an existing entry in the list. All mentions whose extent is not in the list are deleted. After this step we obtain a corpus without nested entities.

2.4. Disambiguation of Entity Types

The next step is to select the entity type for each mention. Here, a context-aware, count-based strategy is used to make a decision. We maintain a vote result for each “mention head string with context” that designates the head string of a entity name mention concatenated with its predecessor and the following words. For each mention in the YAGO corpus on the AIDA ontology, each entity type of that mention votes for itself for the corresponding “mention head string with context”. The decision is based on the entity type that receives the most votes.

2.5. Re-tagging

After the above steps, the corpus no longer contains format inconsistencies and can be used for training. However, due to the noise in YAGO data, the quality of the corpus is still low: There are still obvious issues, such as mention deletion, mention insertion or entity type substitution. Our idea for solving these problems is to introduce a well-trained named entity recognition (NER) model and use this model to disambiguate the mention extents and entity types as an alternative to the steps in subsection 2.3 and 2.4.

We note that the five TAC entity types of previous years (`PER`, `ORG`, `GPE`, `LOC` and `FAC`) match five of the first level entity types defined in AIDA annotation. Therefore, we apply the TAC 2015 training and evaluation sets and the re-tagged OntoNotes data to train an NER model. We then use this model to re-tag the YAGO corpus after step 2.2. Because the estimated mentions of our model (further details in Section 3) are non-nested, the mention overlapping problem can be solved by the re-tagging. The way to disambiguate entity types is similar to the procedure in 2.4. The difference is the restriction that the highest level types of the mention should match the model estimate.

2.6. Combination and Solving Conflicts

Compared with the steps 2.3 and 2.4, the method in 2.5 not only solves the format inconsistency problem but also improves the quality of the corpus, since the model used for re-tagging is cleaner than the YAGO corpus. However, this model does not cover all entity types of TAC-KBP 2019 EDL Track. Therefore, we need to combine the result of step 2.4 and 2.5.

For the five TAC entity types and their subordinate types, the mentions found with the re-tagging method are more accurate (according to a test on TAC 2017) and thus are used. For types `WEA` and `VEH` we use the mentions obtained from 2.4 if they do not overlap with the mentions estimated by the re-tagging method.

2.7. Data Filtering

We define our training corpus by selecting sentences with high quality of annotation, while keeping the variety of the YAGO data.

To reduce the proportion of omitted mentions, we first select those sentences of which at least 30% of the tokens are annotated. About 4.7 million sentences pass this filter.

Then we set two types of threshold for further filtering the data. The first type of threshold is called "extent similarity", which compares the extent of the original YAGO annotation with the re-tagged corpus. For each sentence, the extent similarity score is calculated as follows:

$$\text{simscore} = \frac{\#(E_{\text{yago}} \cap E_{\text{retag}})}{\#(E_{\text{yago}} \cup E_{\text{retag}})} \tag{1}$$

where E_{yago} is the set of mention extents of the sentence with the YAGO annotation, and E_{retag} is the set of mention extents of the same sentence with the re-tagged annotation. This score can help us select sentences whose mention extents with the YAGO annotation is similar to the extents with the re-tagged annotation.

The second type of threshold is the confidence score. When using our model to re-tag sentences from YAGO, we calculate the average confidence score of the tokens and use it as the confidence score of the sentence.

$$\text{confscore}(S) = \frac{\sum_{t \in \text{Token(S)}} \text{confidence}(t)}{\#\text{Token(S)}} \tag{2}$$

Where Token(S) is the sequence of tokens in sentence S, and confidence(t) is the confidence score of token t. By selecting sentences with higher confidence score we get sentences with a higher probability that the annotation is correct.

To combine the two types of threshold, we select sentences of different threshold values with different sampling rates (Table 1). The reason why we set a lower threshold for each criterion is that we hope the selected sample is representative. We want to avoid selecting only certain types of sentences.

Table 1: Sampling rates of different portions of the corpus.

	confscore ≥ 0.96	$0.88 \leq$ confscore < 0.96	otherwise
simscore ≥ 0.7	1.0	0.5	0
$0.5 \leq$ simscore < 0.7	0.5	0	0
otherwise	0	0	0

By applying these filtering techniques, we finally obtain a processed corpus with around 1.4 million sentences. Due to time and memory limitation, we further downsize the training data. Although experiments show that increasing the volume of training data has a slight positive impact on performance, taking into account resource and time constraints, we decide to limit the training volume to 100k.

3. System Design

We use the LSTM-CRF architecture implemented in the Flair framework [6]. The pre-trained Flair embeddings [7] are also included. In the default configuration, the hidden dimension of the LSTM is 256, we increase it to 512 as it improves the performance.

3.1. Training Details

We train four models to build our submissions for the second evaluation window (Table 2). Models 1-3 are first trained using the 100k sentences (YAGO-100K) for 75 epochs, and then fine-tuned using the concatenation of 25k YAGO sentences and tenfold up-sampled AIDA data for 15 epochs. The difference between Model 1,2 and 3 is that they are respectively trained on type, subtype and sub-subtype level of annotation. Model 4 is trained directly using the concatenation of the YAGO and the up-sampled AIDA corpus.

Table 2: Details of different models. YAGO-100k: 100k sentences of processed YAGO Corpus. usp-AIDA: tenfold up-sampled and shuffled AIDA data. YAGO-25K: 25k sentences of processed YAGO Corpus.

Model ID	Training Data	Annotation Level	Fine-tuning Data
1	YAGO-100k	sub-subtype	YAGO-25k + usp-AIDA
2		subtype	
3		type	
4	YAGO-100k + usp-AIDA	sub-subtype	-

3.2. System Combinations

Models 1-4 in Section 3.1 are used to decode the test data what results in 4 independent annotations of the test corpus. We then use three different ways to combine the decoding results by Models 1-4 which allow us to generate three submissions.

- RWTH_1: This is a back-off system. We take all mentions from Model 1, then insert mentions from Model 2 which do not conflict with entries from Model 1 (conflicts are mention extent overlap or multi-types problems). We then analogously insert mentions from Model 1. At last, we add mentions from Model 4 whose entity types only exist in AIDA data (such as `VAL`, `LAW`) to the submission file.
- RWTH_2: This system is inspired by the human feedback of the first evaluation window. We note that a large proportion of the error is "wrong type" (41%). However, some of the incorrect types are correct at type level (highest level) or subtype level (23% of the feedback errors). In addition, experimental results on TAC 2017 and AIDA show that model 3 has more accurate estimate on the type level compared with model 2 and 1. Thus for this run, we add the mentions in a top-down manner rather than back-off. We take all the mentions from model 3. If an estimated mention by model 2 matches the prediction of model 3 (model 3 annotate the same mention extent, and for this mention, the entity type in model 3 is its super-type in model 2), then we will overwrite the entity type of the mention. Analogously we overwrite estimated mentions by model 1. At last, we add mentions from model 4 whose entity types only exist in AIDA data (such as `VAL`, `LAW`) to the submission file.
- RWTH_3: This system is a simplified version of RWTH_1. We use model 4 as an alternative to model 3. Then the mention list is extended by model 2 and model 1 similarly as in RWTH_1.

4. Experimental Results

Table 3 shows the official test results and the results that we subsequently calculate. According to the test result on the core subset of TAC-KBP 2019 EDL Track (excluding the 10 documents that were used to give feedback from the first evaluation window), our system combination slightly increases the recall rate of the prediction. However, it significantly increases the number of false positives. Compared to the combined system we have submitted, our original system (Model 1) without a system combination performs much better (the F1-score increases from about 39.4% to 44.2%).

A tokenization error we did not notice prior the evaluation also lead to a severe deterioration. By simply ignoring the suffix `'s` at the end of the entities, we achieve a further improvement on the F1-score of 2%.

For extraneous entity types that exist only in AIDA data, the test set contains only 92 entities. However, our systems estimate more than two thousand such entities. We will further investigate the possible reasons.

Table 3: Test results on the TAC 2019 EDL core collection. The prefix "token_" represents the result when the tokenization problem is resolved.

	System	Precision[%]	Recall[%]	F1-score[%]
Systems with Combinations	RWTH_1	34.4	44.0	38.6
	RWTH_2	35.0	43.3	38.7
	RWTH_3	35.0	45.1	39.4
	token_RWTH_3	37.0	**47.6**	41.6
Systems without Combinations	Model_1	48.1	40.9	44.2
	token_Model_1	**50.6**	43.1	**46.5**
	Model_4	40.9	43.3	42.1
	token_Model_4	42.9	45.4	44.1
Other Teams	Median system	-	-	42.3
	Best system	-	-	61.4

References

[1] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini, M. El-Bachouti, R. Belvin, A. Houston, Ontonotes release 5.0, Tech. rep., Linguistic Data Consortium (2012).

[2] H. Ji, J. Nothman, B. Hachey, R. Florian, Overview of tac-kbp2015 tri-lingual entity discovery and linking, in: TAC, 2015.

[3] H. Ji, J. Nothman, Overview of tac-kbp2016 tri-lingual edl and its impact on end-to-end kbp, in: TAC, 2016.

[4] F. Suchanek, G. M. Kasneci, G. M. Weikum, Yago: A Core of Semantic KnowledgeUnifying WordNet and Wikipedia, in: 16th international conference on World Wide Web, Proceedings of the 16th international conference on World Wide Web, Banff, Canada, 2007, pp. 697 – 697. doi:10.1145/1242572.1242667.
URL https://hal.archives-ouvertes.fr/hal-01472497

[5] F. Mahdisoltani, J. A. Biega, F. M. Suchanek, Yago3: A knowledge base from multilingual wikipedias, in: CIDR, 2014.

[6] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, R. Vollgraf, FLAIR: An easy-to-use framework for state-of-the-art NLP, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 54–59. doi:10.18653/v1/N19-4010.
URL https://www.aclweb.org/anthology/N19-4010

[7] A. Akbik, D. Blythe, R. Vollgraf, Contextual string embeddings for sequence labeling, in: Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 1638–1649.
URL https://www.aclweb.org/anthology/C18-1139