IBM Research AI Systems for TAC2020 KBP: RUFES Track

Parul Awasthy and **Ken Barker** and **Jian Ni** and **Taesun Moon** and **Radu Florian**
IBM Research AI
Yorktown Heights, NY 10598
{awasthyp, kjbarker, nij, tsmoon, raduf}@us.ibm.com

Abstract

Fine-grained entity typing extracts entity mentions from text and classifies them into types drawn from a relatively large and detailed type system. The TAC 2020 KBP RUFES task (Recognizing Ultra Fine-grained Entities) aims to drive research on techniques for fine-grained entity typing with limited manually annotated training data. We explore three approaches with varying dependence on supplied annotations: a baseline supervised learning system trained only on the available data, a weakly supervised system that adapts data from a different type system and corpus genre, and a synthetic data augmentation system that uses supplied annotations to generate similar data automatically. All three systems show above-median performance on the fine-grained typing metric, with the fully-supervised system performing best overall.[1]

1 Introduction

Fine-grained entity typing has been found useful for many NLP applications such as Question Answering, Natural Language Understanding, Event Extraction, and Knowledge Graph Construction (Aliod et al., 2006; Lewis et al., 2019; Wang et al., 2020; Nguyen and Nguyen, 2019). This task is more challenging than traditional coarse-grained entity typing, in which mentions of entities in text are to be labeled with one of a small number of unambiguous types, such as *Person*, *Organization*, or *Location*. It is also less amenable to the usual supervised learning approaches: having a large number of fine-grained classes requires considerably more training data, and mentions are often more ambiguous and contextual with respect to those types. Existing benchmarks (Ling and Weld, 2012) have little gold annotated data, and rely on silver annotations. These benchmarks usually also focus on named entities only.

[1]Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)

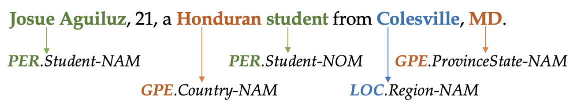

Figure 1: Example of text with RUFES fine-grained labels.

1.1 RUFES Task

The TAC 2020 KBP: Recognizing Ultra Fine-grained Entities (RUFES) track goes beyond previous benchmarks in breadth and depth, challenging participants to explore novel approaches for extracting and classifying entities, given a large type system and limited training data. The shared task defines a shallow (three-level) hierarchical taxonomy of 265 fine-grained entity types that are appropriate for information extraction from news text. There are fourteen coarse-grained, level 1 types (such as *ORG*, *PER*, and *Pathogen*) with 251 fine-grained subtypes (such as *ORG.EducationalInstitution*, *PER.Artist.Musician*, and *Pathogen.Virus.Coronavirus*). The task extends the target mentions to named, nominal and pronominal mentions (*NAM*, *NOM*, and *PRO*). Moreover, it requires within-document coreference of entity mentions. Figure 1 shows an example with RUFES annotations.

The task is made even more challenging (and representative of real-world applications) by having a type system that mixes entities and role types without distinction and providing little information on their semantics. For example, "Brookfield High School" may be of type *ORG.EducationalInstitution.SecondarySchool* or of type *FAC.Building.School* (or both). A mention of "corn" may be of type *ConsumerGoods.Food*, but only if it is intended to be bought and eaten by a consumer, not if it is bought by a bakery to make corn bread.

The RUFES evaluation has two phases. In the first phase (Evaluation Window 1), participants submit entity extractions over a large (100,000 document) development corpus. The RUFES organizers return feedback on ten documents, including error logs and gold annotations for the first 40 system errors. Participants use the feedback to improve their systems for the second evaluation phase, for which they submit new system output on the development corpus. The phase-2 output is scored to produce the official task results.

2 Data

We use data from different sources in developing our systems for the RUFES task, including annotated and unannotated data distributed by the RUFES organizers, in-house data from other projects, and publicly available structured and unstructured data.

RUFES organizers shared 50 sample documents with gold-standard annotations as well as a large, unannotated development corpus of 100,000 documents of the same genre. All documents are Washington Post news articles. In this paper we refer to these datasets as "the 50 sample documents" and "the 100,000-document development corpus".

From the feedback gold annotations returned after the first evaluation phase, we produced a small, gold-standard dataset to be used as a held-out test set in our experiments, allowing us to use the full 50 sample documents for training and tuning. In this paper we refer to this extra, held-out set as "the 10 feedback documents". We note that these documents are not complete documents, since the gold annotations in the feedback stop after the first 40 errors. As such, the documents may not be representative of the distribution of the complete documents in the evaluation corpus.

2.1 Data Transfer

One approach for working in a domain with little training data is to adapt existing datasets from similar domains.

2.1.1 KLUE Data

Our in-house KLUE dataset (Florian et al., 2004) uses an entity type system similar enough to the RUFES coarse-grained types that KLUE-annotated data can be used without adaptation. Twelve of the KLUE entity types are direct matches with RUFES coarse-grained types. We remove all mentions from the KLUE data that do not correspond to RUFES types. For the two RUFES coarse types not in the KLUE type system (*Document*, *Pathogen*), we extend the KLUE dataset with examples generated from the DBpedia Abstract corpus (section 2.1.2).

The KLUE dataset cannot be used for fine-grained entity typing, since its type system contains only coarse types. But we can use this data for training a coreference resolution model (section 4) as well as for a coarse-grained entity detection and classification model that can be cascaded with a fine-grained typing model (section 3.2.1).

2.1.2 The DBpedia Abstract Corpus

The DBpedia Abstract corpus (DBPA) consists of Wikipedia abstracts (the introductory section of Wikipedia articles) annotated with DBpedia resources. The hyperlinks in Wikipedia articles are automatically converted to DBpedia URIs, giving a large, broad-domain corpus in seven languages annotated with entity links. The English DBpedia Abstract corpus contains 4,415,993 abstracts with 39,650,948 linked entities.

Since DBpedia resources cover similar semantic territory to the fine-grained RUFES types, our hypothesis is that it should be possible to adapt a corpus with DBpedia annotations for training a model to detect RUFES types. The most significant challenge in adapting the corpus is how to map the roughly three million unique DBpedia resources linked in the corpus to RUFES types.

Rather than try to map the huge DBpedia vocabulary to RUFES types, we concentrate on populating the RUFES type system with instances, and mapping those instances automatically to DBpedia resources. We devise four procedures for populating RUFES types with instances.

1. *Map the RUFES types to DBpedia ontology classes.* The majority of DBpedia resources (more than 90%) are classified into a shallow, top-level ontology of 685 classes (DBO). Many of the classes are mappable to RUFES types. DBpedia resources that belong to DBpedia ontology classes mappable to RUFES types are candidate instances of those types.
2. *Mine Wikipedia Categories.* Wikipedia Categories noisily group related Wikipedia pages. Many RUFES types can be mapped to Wikipedia Categories, and the pages within those categories used as instances of the

type. The resulting Wikipedia page names can be converted automatically to DBpedia resources.

3. *Generate labels from RUFES type names and examples, and expand them through WordNet.* Words and phrases associated the RUFES types (including type names naïvely converted to phrases) are expanded to synonyms and hyponyms with WordNet (Miller, 1995). These names are mapped to DBpedia resources by string matching against DBpedia labels.
4. *Populate RUFES types with instances from high-precision lists.* We generate web queries from type names (and their synonyms and hyponyms) to find lists of instances. In many cases, Wikipedia includes such lists, which can be scraped for Wikipedia titles, which can be converted automatically to DBpedia resources.

Using these techniques we generate a map with 1,981,985 unique, canonicalized DBpedia resources mapped to 261 RUFES types. (We found no mappings for *FAC.GardenPark*, *LOC.GeographicPoint*, *ORG.Court.LocalCourt*, or *PER.CivilServant.PolicyAdvisor*). The type with the most instances is *PER.Professional.Athlete* (430,062). The median number of instances per type is 251.

We use the map to replace entity links the DBpedia Abstract corpus with RUFES types, giving a corpus with 12,437,702 RUFES type annotations. This corpus can be used directly for training a fine-grained entity typing model.

3 Fine-Grained Entity Typing

We explore different approaches for extracting and typing fine-grained entities with each of our three submission systems. Building a supervised system usually works well for entity typing when there is sufficient training data. Since the goal of the RUFES task is to extract entities given limited training data, we also explore a weakly supervised approach that transfers data from another domain, and a synthetic approach that automatically augments the small amount of supplied data. In this section we describe each of our three systems in detail.

3.1 Supervised System

Following the work of (Devlin et al., 2019), we train a supervised token classification model using Hugging Face transformers (Wolf et al., 2019). We randomly divide the 50 sample documents into 39 train and 11 dev documents and train a model on this data. The architecture for this model (with an example) is illustrated in figure 2. An input sentence is tokenized and passed through a transformer-based model to generate a contextualized vector representation for each word token. The vectors from the final transformer layer are then passed to a linear classifier layer mapped to the RUFES label space to get the RUFES label predictions for each token.

3.2 Weakly Supervised System

Our weakly supervised system is a two-step cascaded system similar to (Awasthy et al., 2020). The first step is a coarse-grained entity detection and typing model. The second is a fine-grained model trained on the adapted DBpedia Abstract corpus (section 2.1).

3.2.1 Coarse-Grained Entity Extractor

The DBpedia Abstract corpus has entity annotations for only those text spans with hyperlinks in the original Wikipedia articles. Many spans that could reasonably be annotated are not. This limits the usefulness of the corpus for training span detection. The weakly supervised model, therefore, is expected to predict fine-grained types only, given mention spans and top-level (coarse-grained) types detected by a coarse-grained model. We train the coarse-grained model on the extended KLUE data (2.1.1), using the same transformer architecture as our fully supervised system (section 3.1).

3.2.2 Fine-Grained Entity Classifier

For predicting the fine-grained RUFES types, we train a weakly supervised model using the adapted DBpedia Abstract corpus (section 2.1) as training data. The model is based on the same transformer architecture for token classification as our supervised system (section 3.1), with three significant extensions. First, mentions spans are provided with the input sentence by surrounding mention span tokens with special boundary tokens. Second, the top-level RUFES type is provided as a feature by using a different special span start token for each of the types. Finally, we add dictionary features to the output vector for each token within a span to be used as input to the linear classifier layer.

The dictionary features are generated from the same data used to produce the DBpedia-to-RUFES map (section 2.1). Each phrase that was used to

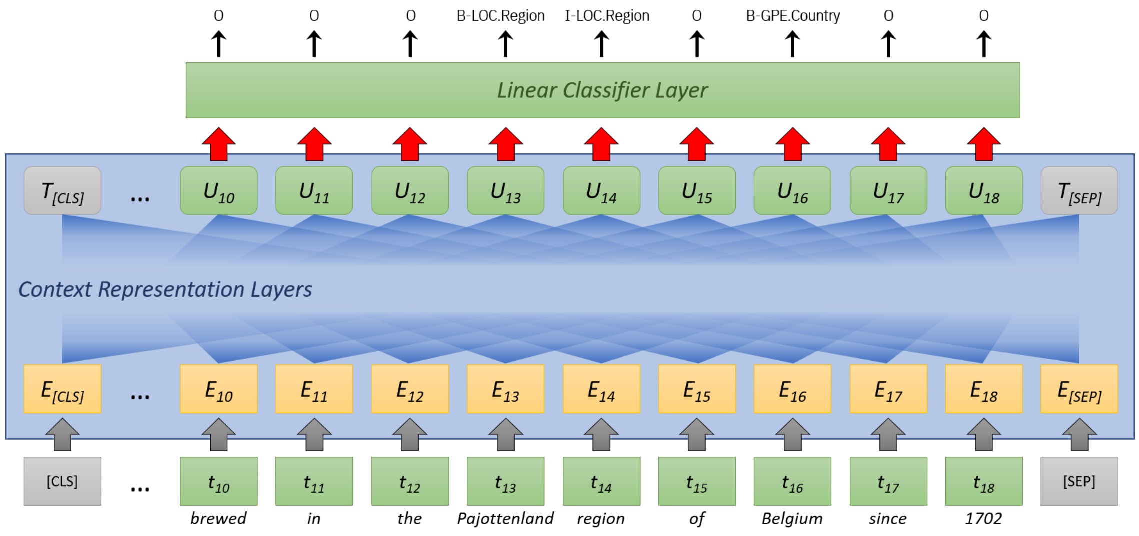

Figure 2: Architecture of a transformer-based token classification model

find a DBpedia resource for the map is normalized and associated with a vector. Each position in the vector corresponds to a RUFES type, with the value equal to the frequency with which the normalized phrase maps to the type. For example, the phrases "Saint Sixtus" and "saint-sixtus" may have been discovered through different instance populating approaches to map to *PER.ReligiousLeader*, *ORG.ReligiousOrganization*, and *Consumer-Goods.Food*, one or more times each. The dictionary will contain an entry for "saintsixtus" with a vector containing frequencies for the three fine types, discounted frequencies for their parent types, and zeros elsewhere. The vectors are then normalized to sum to one. When a token span that normalizes to "saintsixtus" is seen in the input, the transformer output vector for each token in the span is extended with the dictionary vector. This vector provides additional signal to the linear classifier even for phrases without examples in the training corpus.

Once the weakly supervised model is trained, we further tune the system in three ways:

1. Fine-tune the model using the 50 provided sample documents as further training data.
2. Apply a confidence threshold to the model output.
3. Adjust the types and entity identifier for pronominal spans (section 4.2).

We compare performance of a single, global confidence threshold (*constant threshold*) and multiple thresholds (*multi-threshold*) that are tuned separately for each top-level type.

Experiments showing the contribution of the different kinds of tuning appear in section 5.1.2.

3.3 Synthetic Data Augmentation

Synthetic data augmentation has proven effective for tasks such as Question Answering (Alberti et al., 2019) and Event Extraction (Hong et al., 2018; Tong et al., 2020). A common approach to generating synthetic data is to train a model on existing data and use it to label more data for further training Zhu (2008); Kingma et al. (2014). Unfortunately, when there is little training data available, this approach heavily biases the model to the training data. To mitigate the bias, we use a two-step approach.

We generate synthetic data from the unannotated development corpus to augment the available data using a cascaded approach as in Section 3.2: extract mention spans and tentative mention labels using a coarse-grained model and then specialize the mention type using a secondary model. The secondary model we train is not a standard parametric model, but a k-Nearest Neighbor Graph (k-NNG) model (Fritzler et al., 2019). We store the representations of gold mentions in a k-NNG as a training step, and at decode time retrieve the top k matching mentions from the graph. We then use a weighted vote to pick the best possible fine-grained label from the retrieved labels and the tentative mention label produced by the first-level model. Once we have annotations on 100 randomly selected documents we train a full entity detection model on these 100 documents. We now provide a detailed description of each stage.

k-NNG: Similar to the idea of few-shot learning proposed in (Fritzler et al., 2019) we create a k-NNG with gold annotations of mentions. Unlike Fritzler et al., who compute and save a representation of each token in a mention, we compute a representation for each mention to store in the graph. For each mention vector *j*, from sentence *i*, represented by *n* token vectors $\{e_1, e_2, ..., e_n\}$, the representation m_{i_j} is computed as:

$$m_{i_j} = concat(e_1; e_n; sen_i) \tag{1}$$

where sen_i is the sentence vector for sentence *i*. This is obtained by using the <CLS> token from the transformer. Using this sentence representation along with the mention representation provides another lens to look at the whole sentence to determine the fine-grained type.

Using the same train-dev split as in Section 3.1, we insert each of the mention representations into a k-Nearest Neighbor graph.

Mention Extraction: Given an unannotated document sampled from the 100,000 document development corpus, we again use a token classification model (section 3.1) to extract mention spans and a candidate type label l^1.

Fine-Grained Typing using k-NNG: For each of the mentions extracted, we compute the mention representation using equation (1). We query the k-NNG with this mention representation and retrieve the k nearest mentions and labels associated with them $l^2 = \{l^2_1, l^2_2 ... l^2_k\}$. We then select the fine-grained type using a threshold t to vote between l^1 and all l^2 types.

$$\hat{l} = \begin{cases} \text{argmax}(\text{count}(l)) & \text{max}(\text{count}(l)\ \forall l \in l^2) \geq t \\ l^1 & \text{otherwise} \end{cases} \tag{2}$$

Synthetic Data Entity Extraction: We now use these documents to train a fine-grained entity extraction model, following the same method described in Section 3.1.

4 Coreference Resolution

We run a coreference model after entity extraction for all of our systems. This means the coreference model runs last for supervised and synthetic data models, and it runs after coarse-grained mention extraction for the weakly supervised model.

4.1 Coreference Model

Our coreference model is an implementation of the Bell-Tree mention synchronous coreference algorithm described in (Luo et al., 2004). The model is trained on our in-house KLUE data, as we didn't have data annotated with the RUFES type system. However, because the system is a statistical coreference model and the mention types are simply features that are provided to it, the model is able to perform coreference on new types.

4.2 Coreference Post-processing

We consider two modifications to the output of the base coreference model.

First, the coreference model by default generates unique, opaque identifiers for entity clusters. But it also generates a canonical text string, which is the most specific phrase appropriate to the mentions within the entity cluster. In our systems' outputs, we replace the opaque identifier with the canonical text. This may result in merged entity clusters for distinct clusters having the same canonical text identifier, which is similar to what appears in the gold data seen in the 50 sample documents.

Second, we have observed that the base coreference model is weaker with pronominal referring expressions. In our systems' outputs, for mentions of type *PRO* whose mention text is a known personal pronoun, we replace the entity identifier (canonical text) and the predicted types list with the identifier and types list of the most recent mention whose top-level type is *PER*. For non-personal-pronoun *PRO* mentions, we use the identifier and types list of the most recent non-*PER* mention. This pronoun post-processing significantly improves all three systems' performance on the 10 feedback documents (see section 5).

5 Experiments

5.1 System Experiments

5.1.1 Supervised System

We use Hugging Face transformers toolkit (Wolf et al., 2019) to train our supervised model. We train for 30 epochs with learning rate of 3e-5. We train five models with different random seeds and select the model with best performance on the development set. We use *xlm-roberta-large* as the pre-trained language model.

For the phase 1 submission we use 39 sample documents for training and 11 for development.

For the phase 2 submission we use all 50 sample documents for training and use the 10 feedback documents for testing.

Pronoun post-processing (section 4.2 results in a 4-5 point improvement in F1 score for all test sets.

5.1.2 Weakly Supervised System

The initial model for the weakly supervised system uses Hugging Face transformers for token classification, with *bert-base-cased* for the pre-trained language model. Training for three epochs with learning rate 5e-5 gives an F1 score of 0.935 on a held-out test set. The final configuration is based on ablation experiments for different configuration parameters or components. We experiment with:

1. Training data size (maximum number of examples per type)
2. Confidence threshold type (global vs. per-type)
3. Confidence threshold pruning (reject or back-off)
4. Pronoun post-processing
5. Model fine-tuning on the 50 sample documents

Increasing the number of training examples per class uniformly improves model performance on the 50 sample documents by a small margin. For example, the F1 score (as calculated by the RUFES score_submission script) is consistently roughly 0.3 percentage points higher for a model trained on (maximum) 10,000 examples per class vs. 5,000 examples per class, independent of the other configuration elements. Smaller training sizes are consistently worse.

The system can be configured either with a constant confidence threshold or individual thresholds for each top-level RUFES type. For both configurations, we tune the thresholds using the 50 sample documents and test on the 10 feedback documents. Annotations below threshold can either be rejected or "backed off" to the top-level type. In our experiments, backing off gives higher F1 scores.

Table 1 shows the effect of configuration elements in isolation and combination on Precision, Recall, and F1 score as calculated by the RUFES scorer on the 10 feedback documents.

The results show that tuned, per-type thresholds outperform a single, global (constant) tuned threshold. Pronoun post-processing helps in isolation and in combination with multi-thresholding. Fine-tuning the model using the 50 sample documents gives significant improvements across all other configuration combinations.

configuration	**P**	**R**	**F1**
10k base model	0.3100	0.3525	0.3171
+ct	0.3450	0.3442	0.3310
+mt	0.3880	0.3337	0.3428
+pp	0.3502	0.3891	0.3535
+mt +pp	0.4308	0.3735	0.3825
+sft	0.3802	0.3773	0.3700
+ct +sft	0.3857	0.3754	0.3715
+mt +sft	0.3866	0.3754	0.3720
+pp +sft	0.4171	**0.4211**	0.4107
+mt +pp +sft	**0.4253**	0.4195	**0.4144**

Table 1: Precision, Recall, F1 score on the 10 feedback documents for the weakly supervised model trained on a maximum of 10,000 examples per class. +ct: constant threshold on all L1 types; +mt: multi-threshold tuned per L1 type; +pp: pronoun post-processing; +sft: additional model fine tuning on the sample 50 documents.

Based on these experiments, we set the final configuration of this system to the 10,000 example model, fine-tuned on the 50 sample documents, with tuned multi-thresholding and pronoun post-processing.

5.1.3 Synthetic Data System

We randomly sample 100 documents from the development corpus for phase 1 and 1000 documents for phase 2. We use the phase 1 supervised model to annotate the first-step mentions. We use the 39 training documents to build the k-NNG. Based on experiments with different threshold values, we set t at 5. Experiments with different distance metrics for k-NNG show that a Cosine distance with $k = 10$ is effective. The k-NNG mention representations are computed using the final layer of the *xlm-roberta-large* model. We have experimented using fine-tuned models to compute the mention representations but the out-of-the-box pre-trained *xlm-roberta-large* with no fine-tuning performs the best.

Once we have silver annotations on this data, we train an entity detection model using these synthetic documents as the training set, the 39 sample documents as the development set and the remaining 11 documents as the test set. We use a setup similar to the supervised system (section 5.1.1) and train 5 seeds. We select the model with best performance on the test set.

System	Fine-Grain F1
Supervised	**0.4453**
Weak Super.	0.4144
Synthetic	0.4189

Table 2: The final fine-grained F1 scores for our three systems on the 10 feedback documents.

5.2 Cross-System Comparisons

Each system uses the development corpus and the 50 sample documents in some way for training and tuning. Table 2 shows results comparing the three systems on the 10 feedback documents that we saved as a held-out test set. The F1 scores for the systems correlate to the scores on the final evaluation data, as reported by RUFES organizers (see section 6).

5.3 Ensembling

Given the three very different approaches of our systems, an obvious next step is exploring ways to combine them in an ensemble.

We explore naïve ensembling of our synthetic data system and our weakly supervised system as follows:

1. Measure the performance of each system on different kinds of mentions.
2. Where both systems agree there is a mention, use the types predicted by the system with better performance for the kind of mention.
3. If the systems disagree on the kind of mention, and the disagreement means there is a conflict on which system is better for this mention, use the types predicted by system 1 as default.

The kinds of mention we consider are mention type (NAM, NOM, or PRO), top-level RUFES type, and a combination of the two. For example, if we are considering mention type, then for mentions of type NOM, the ensemble prefers types predicted by the system whose performance on NOM mentions in the 50 sample documents was better, and so on. If we are considering top-level RUFES type, then for mentions of type PER, the ensemble prefers types predicted by the system better at PER mentions, and so on.

We experiment with both systems as "system 1" (the default system for conflicts).

Unfortunately, all experiments show insignificant improvement over individual system performance. We defer more sophisticated ensembling approaches to future work.

System	StrongMention	CEAF	Fine-Grain
Supervised	0.838	0.594	**0.4173**
Weak Super.	0.789	0.570	0.3521
Synthetic	0.825	0.584	0.3936
Median	0.812	0.577	0.3226
Max	**0.867**	**0.687**	**0.4173**

Table 3: Phase 2 scores for our three systems on the three metrics. Max and Median scores are calculated on the best-performing submission from each participant, as shared by RUFES organizers.

6 Final Results and Discussion

The submitted outputs of systems are evaluated by RUFES organizers using three metrics:

- *Strong Mention Match*: F1 score that measures the performance of the mention detection system on capturing the correct mention boundaries, ignoring the types of the mentions.
- *Mention CEAF*: "Constrained Entity-Alignment F-measure" metric proposed by (Luo, 2005) to score Coreference output.
- *Fine-Grain F1*: a new metric proposed for this task to measure the fine-grained mention output. This combined metric evaluates mention span detection, coreference, and predicted fine-grained types of mentions.

The results of the RUFES phase 2 evaluation for each of our three systems on the 104 test documents, including the feedback documents, are shown in Table 3. All three of our systems perform significantly better than the median on the fine-grained typing score. Our supervised system scores more than 18 F1 points above median; our three systems score on average 15 F1 points above median. Two of our systems perform slightly above median on the mention detection (*StrongMention*) and coreference (*CEAF*) metrics, with our third model scoring slightly below median.

We note that the performance of our three systems is directly related to the extent to which the system relies on the gold sample annotations. The supervised system uses *only* the manually annotated 50 sample documents for training. The synthetic system generates new training data from unannotated documents, optimizing for similarity to the 50 sample documents. The base version of the weakly supervised system does not use the sample gold annotations at all, and trains on a noisily adapted corpus of a different genre. The competitive performance of the final version, however, is due in part to fine tuning the base model on the 50 sample documents. This is a somewhat disappointing result. One of the explicit goals of the RUFES task is to explore techniques beyond simple supervision using manual gold annotations, which is expected to work poorly for such a large, complex type system with limited training data. Yet even with a very small amount of gold data, the simple, supervised system performs best. Clearly the unseen test data is a close match to the sample data.

We argue that perhaps the superiority of the supervised system over our other systems is not an indictment of the low-data methods we propose. Rather, it emphasizes the difficult challenges in creating evaluation frameworks that focus precisely on the behavior we target in tasks such as these.

7 Conclusion

We have described three approaches to the fine-grained entity extraction task that vary in their dependence on within-domain, manually annotated training data. A simple, fully-supervised model is trained on the supplied annotations only. A synthetic data approach automatically generates new data from an unlabeled corpus, intended to match the properties of supplied annotations. A weakly supervised approach adapts annotations from a different type system on an unrelated corpus for training, based on detailed analysis of the target types. In our own experiments and in the official RUFES task results, the performance of the systems is correlated to their dependence on the manually annotated data, with the fully-supervised system performing best.

8 Acknowledgements

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-19-C-0206. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Defense Advanced Research Projects Agency (DARPA)

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Diego Mollá Aliod, M. Zaanen, and Daniel Smith. 2006. Named entity recognition for question answering. In *ALTA*.

Parul Awasthy, Taesun Moon, Jian Ni, and Radu Florian. 2020. Cascaded models for better fine-grained named entity recognition. *arXiv preprint arXiv:2009.07317*.

DBO. The dbpedia ontology. `https://wiki.dbpedia.org/services-resources/ontology`. Accessed: 02/12/2021.

DBPA. The dbpedia abstract corpus. `http://downloads.dbpedia.org/2015-04/ext/nlp/abstracts`. Accessed: 02/12/2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, H Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. Technical report, IBM Thomas J. Watson Research Center, Yorktown Heights, NY.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 993–1000.

Yu Hong, Wenxuan Zhou, Jingli Zhang, Guodong Zhou, and Qiaoming Zhu. 2018. Self-regulation: Employing a generative adversarial network to improve event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 515–526.

Diederik P Kingma, Danilo J Rezende, Shakir Mohamed, and Max Welling. 2014. Semi-supervised learning with deep generative models. *arXiv preprint arXiv:1406.5298*.

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. *arXiv preprint arXiv:1906.04980*.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, page 94–100. AAAI Press.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 135–142, Barcelona, Spain.

G. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6851–6858.

Meihan Tong, Bin Xu, Shuai Wang, Yixin Cao, Lei Hou, Juanzi Li, and Jun Xie. 2020. Improving event detection via open-domain trigger knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5887–5897, Online. Association for Computational Linguistics.

Xuan Wang, Xiangchen Song, Yingjun Guan, Bangzheng Li, and Jiawei Han. 2020. Comprehensive named entity recognition on cord-19 with distant or weak supervision. *arXiv preprint arXiv:2003.12218*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Xiaojin Zhu. 2008. Semi-supervised learning literature survey. *Comput Sci, University of Wisconsin-Madison*, 2.