What Does This Acronym Mean? Introducing a New Dataset for Acronym Identification and Disambiguation

Amir Pouran Ben Veyseh[1], **Franck Dernoncourt**[2], **Quan Hung Tran**[2],
and **Thien Huu Nguyen**[1]
[1] Department of Computer and Information Science, University of Oregon, USA
[2] Adobe Research, San Jose, CA, USA
{apouranb,thien}@cs.uoregon.edu,
{franck.dernoncourt,qtran}@adobe.com

Abstract

Acronyms are the short forms of phrases that facilitate conveying lengthy sentences in documents and serve as one of the mainstays of writing. Due to their importance, identifying acronyms and corresponding phrases (i.e., acronym identification (**AI**)) and finding the correct meaning of each acronym (i.e., acronym disambiguation (**AD**)) are crucial for text understanding. Despite the recent progress on this task, there are some limitations in the existing datasets which hinder further improvement. More specifically, limited size of manually annotated AI datasets or noises in the automatically created acronym identification datasets obstruct designing advanced high-performing acronym identification models. Moreover, the existing datasets are mostly limited to the medical domain and ignore other domains. In order to address these two limitations, we first create a manually annotated large AI dataset for scientific domain. This dataset contains 17,506 sentences which is substantially larger than previous scientific AI datasets. Next, we prepare an AD dataset for scientific domain with 62,441 samples which is significantly larger than previous scientific AD dataset. Our experiments show that the existing state-of-the-art models fall far behind human-level performance on both datasets proposed by this work. In addition, we propose a new deep learning model which utilizes the syntactical structure of the sentence to expand an ambiguous acronym in a sentence. The proposed model outperforms the state-of-the-art models on the new AD dataset, providing a strong baseline for future research on this dataset [1].

1 Introduction

Acronyms are shortened forms of a longer phrase. As a running example, in the sentence "*The main key performance indicator, herein referred to as KPI, is the E2E throughput*" there are two acronyms *KPI* and *E2E*. Also, the acronym *KPI* refers to the phrase *key performance indicator* (a.k.a. the long form of the acronym KPI). In written language, acronyms are prevalent in technical documents that helps to avoid the repetition of long and cumbersome terms, thus saving text space. For instance, about 15% of PubMed queries include abbreviations, and about 14.8% of all tokens in a clinical note dataset are abbreviations (Islamaj Dogan et al., 2009; Xu et al., 2007; Jin et al., 2019).

Considering the widespread use of acronyms in texts, a text processing application, such as question answering or document retrieval, should be able to correctly process the acronyms in the text and find their meanings. To this end, two sub-tasks should be solved: **Acronym Identification (AI)**: to find the acronyms and the phrases that have been abbreviated by the acronyms in the document. In the running example, the acronyms *KPI* and *E2E* and the phrase *key performance indicator* should be extracted. **Acronym Disambiguation (AD)**: to find the right meaning for a given acronym in text. In the running example, the systems should be able to find the right meanings of the two acronyms *KPI* and *E2E*. Note that while the meaning of *KPI* is found in the sentence, the meaning of *E2E* should be inferred from its occurrences either in the previous text of the same document or external resources (e.g., dictionaries).

[1]Dataset for AI is available at https://github.com/amirveyseh/AAAI-21-SDU-shared-task-1-AI and dataset for AD is available at https://github.com/amirveyseh/AAAI-21-SDU-shared-task-2-AD

Proceedings of the 28th International Conference on Computational Linguistics, pages 3285–3301
Barcelona, Spain (Online), December 8-13, 2020

Acronym Identification and disambiguation can be used in many downstream applications including slot-filling (Veyseh et al., 2020), definition extraction (Kang et al., 2020; Veyseh et al., 2020), and question answering (Ackermann et al., 2020; Veyseh, 2016)

Over the past two decades, several approaches and resources have been proposed to solve the two sub-tasks for acronyms. These approaches extend from rule-based methods for AI and feature-based models for AD (i.e., SVM and Naive Bayes) (Schwartz and Hearst, 2002; Nadeau and Turney, 2005; Okazaki and Ananiadou, 2006; Yu et al., 2007) to the recent deep learning methods (Li et al., 2015; Charbonnier and Wartena, 2018; Wu et al., 2015; Ciosici et al., 2019; Jin et al., 2019; Li et al., 2019). While the prior work has made substantial progress on this task by providing new approaches and datasets, there are some limitations in the existing datasets which hinder further improvement. First, most of the existing datasets for AI are either limited in their sizes or created using simple rule-based methods (i.e., not humnan-annotated). For instance, Ciosici (2019) exploits the rules proposed by Schwartz (2002) to generate a corpus for acronym disambiguation from Wikipedia. This is unfortunate as rules are in general not able to capture all the diverse forms to express acronyms and their long forms in text (Harris and Srinivasan, 2019). This limitation not only restricts the coverage of the previous methods and datasets for AI, but also imposes some bias to evaluation of AI using these datasets. Second, most of the existing datasets are in the medical domain, ignoring the challenges in other scientific domains. While there are a few datasets for general domain (e.g., Wikipedia), online forums, news and scientific documents (Thakker et al., 2017; Li et al., 2018; Charbonnier and Wartena, 2018; Liu et al., 2011; Harris and Srinivasan, 2019), they still suffer from either noisy examples inevitable in the heuristically generated datasets (Charbonnier and Wartena, 2018; Ciosici et al., 2019; Thakker et al., 2017) or their small size, which makes them inappropriate for training advanced methods (e.g., deep neural networks) (Prokofyev et al., 2013; Harris and Srinivasan, 2019; Nautial et al., 2014).

In order to address the limitations of the existing resources and push forward the research on acronym identification and disambiguation, especially in scientific domain, this paper introduces two new datasets for AI and AD. Notably, our datasets are annotated by human to achieve high quality and have substantially larger numbers of examples than the existing AI datasets in the non-medical domain (see Appendix B.3). Moreover, based on acronyms and long forms obtained by AI annotations, we introduce a new AD dataset that is also substantially larger than the existing AD datasets for the scientific domain. For more details about the datasets, see Appendices B.2 and B.3. Finally, we conduct extensive experiments to evaluate the state-of-the-art methods for AI and AD on the proposed datasets. The experiments show that the existing models fail to match the human-level performance on the proposed datasets.

Motivated by the unsatisfactory performance of the existing AD models, we introduce a new deep learning model for acronym disambiguation. In particular, one of the limitations of the prior AD models is that they fail to efficiently encode the context of the ambiguous acronyms in the given sentences. On the one hand, the traditional feature-based models for AD are limited in their ability to effective representations for the contexts due to the use of hand-designed features. On the other hand, the current deep learning models for AD, based on either language models or sentence encoders (e.g., LSTMs) to capture the contextual information, cannot effectively encode the long dependencies between words in the sentences. These long dependencies are important for acronym disambiguation as the meanings of the acronyms might depend on some words that are sequentially far from the acronyms of interest. To address the issue of long dependencies in sentences, prior work on other natural language processing (NLP) applications has shown that dependency trees can be employed to capture dependencies between words that are sequentially far from each other (Veyseh et al., 2020). As none of the recent AD models employs the dependency trees to encode the sentence context for acronyms, we propose a novel deep learning model for AD that exploits the syntactic dependency structures of the sentences using graph convolutional neural networks (GCN) (Kipf and Welling, 2016). In this work, the structure information induced by the GCN model would be employed as the structural context for acronym disambiguation. Our experiments show that the proposed model outperforms the existing state-of-the-art AD models, providing a strong baseline for the future work on the proposed dataset for AD.

In summary, in this paper, we make the following contributions:

- We release the first publicly available and the largest manually annotated acronym identification dataset in scientific domain. Moreover, we also release the largest acronym disambiguation dataset in scientific domain which is created from humanly curated dictionary of ambiguous acronyms
- We conduct extensive experiments on the proposed datasets and compare the performance of the state-of-the-art acronym identification and disambiguation systems.
- We propose a new graph-based acronym disambiguation method that employs the syntactic structure of the sentence to predict the expanded form of the ambiguous acronym. This baseline outperforms existing state-of-the-art models for acronym disambiguation.

2 Data Collection and Annotation

In order to prepare a corpus for acronym annotation, we collect a corpus of 6,786 English papers from arXiv. These papers consist of 2,031,592 sentences that would be used for data annotation for AI and AD in this work.

2.1 AI Dataset

AI aims to identify acronyms (i.e., short forms) and phrases that have been abbreviated by the acronyms in text (i.e., long forms). To create a dataset for this task, we manually annotate the sentences from the collected papers. In order to improve the efficiency of the data annotation, we seek to annotate only a set of sentences $S = \{s_1, s_2, ..., s_n\}$ that have a high chance to host short forms or long forms for acronyms. As such, each sentence s_i for annotation needs to contain at least one word w_a in which more than half of the characters in w_a are capital letters (i.e., acronym candidates). Afterward, for each sentence s_i in S, we search for a sub-sequence of words $W_t = \{w_t, w_{t+1}, ..., w_{t+k}\}$ in s_i in which the concatenation of the first one, two or three characters of the words $w_j \in W_t$ (in the order of the words in the sub-sequence W_t) could form an acronym candidate w_a in s_i[2]. We call W_t as long form candidate. If we cannot find any long form candidate W_t in sentence s_i, we remove sentence s_i from S. Using this process, we end up with 17,506 sentences in S to be annotated manually by the annotators from Amazon Mechanical Turk (MTurk). In particular, we create a HIT for each sentence and ask the workers to annotate the short forms and the long forms in the sentence. Also, we ask them to map each long form to its corresponding short form. In order to work on these HITs, workers should have an approval rate of more than 99% and pass a qualification test. Moreover, we remove the annotations from workers who submit the HIT sooner than a preset time or have more than 10% of their annotations different from with the others'. We hire three workers per HIT and pay each of them \$0.05. In case of disagreements, if two out of three workers agree on an annotation, we use majority voting to decide the correct annotation. Otherwise, a fourth annotator is hired to resolve the conflict. The inter-annotator agreement (IAA) using Krippendorff's alpha (Krippendorff, 2011) with the MASI distance metric (Passonneau, 2006) for short-forms (i.e., acronyms) is 0.80 and for long-forms (i.e., phrases) is 0.86. Out of the 17,506 annotated sentences, 1% and 24% of the sentences do not contain any short form or long form, respectively. We call this dataset **SciAI** (scientific acronym identification). In this work, the AI task is formulated as sequence labeling where we provide the boundaries for the acronyms and long forms in the sentence using BIO format (i.e., label set includes *B-acronym*, *I-acronym*, *B-long*, *I-long* and *O*).

2.2 AD dataset

Using the AI dataset annotated in the previous section, we first create a dictionary of acronym meanings (i.e., a mapping from short forms to all of the possible long forms). First, we create a meaning candidate set[3] for each acronym based on the annotations in AI dataset. As some of the long forms in the meaning set might be different variants of a single meaning, we need to normalize the long forms of each acronym to find the canonical set of long forms of the acronym. To achieve this goal, we use the normalization

[2]Note that we also allow some words $w_j \in W$ to be omitted when we make the concatenation of the characters. Also, note that w_a could be any acronym candidate in s_i

[3]In this paper, we use *meaning* and *long form* interchangeably

approach proposed by the prior work (Ciosici et al., 2019). The main idea is to compute the Levenshtein string edit distance between any pair of long forms, and if it is smaller than a threshold, we replace the less common long form with the more frequent one. After creating the canonical set of long forms for the acronyms, we remove unambiguous acronyms from the mapping/dictionary (i.e., acronyms with only one possible long form), leading to a dictionary $\mathcal{D}$ of ambiguous acronyms. Furthermore, in order to improve the quality of the mappings in $\mathcal{D}$, a graduate student curates the dictionary $\mathcal{D}$ to verify the canonical forms and resolve any potential error. Afterwards, we employ $\mathcal{D}$ to create an AD dataset. To this end, we assume that if an acronym is defined in a paper, its meaning will not change throughout that paper, i.e., the one-sense-per-discourse assumption. Using this criterion, for each acronym a in the dictionary $\mathcal{D}$, we look up all sentences s_i in S, i.e., the set of sentences annotated for AI, that contains a and its corresponding long form l (indicated by the annotation for SciAI). Note that here l, or its more frequent variant, is already in the canonical long form set of a in $\mathcal{D}$. Then, we assign the long form l to all the occurrences of a in the document that contains s_i. This approach leads to a dataset of 62,441 samples where each sample involves a sentence, an ambiguous acronym, and its correct meaning (i.e., one of the meanings of the acronym recorded by the dictionary $\mathcal{D}$). We call this dataset **SciAD** (scientific acronym disambiguation). Consequently, AD is formulated as a classification problem where given a sentence and an acronym, we need to predict the long form of the acronym in a given candidate set.

3 Model

In this section we describe the details of the proposed model for AD. As mentioned in section 2.2, we formulate this task as sequence classification problem. Formally, given an input sentence $W = w_1, w_2, ..., w_n$ and the position of the acronym, i.e., p, the goal is to disambiguate the acronym w_p, that is, predicting the long form l from all the possible long forms of w_p specified in the dictionary $\mathcal{D}$. The proposed model consists of three major components: (1) **Sentence Encoder**: This component encodes the input sentence into a sequence of representation vectors for the words, (2) **Context Encoder**: In order to augment the representation of the acronym w_p, the context encoder component is designed to leverage the syntactical structure of the sentence to generate the context representation vector for w_p, and (3) **Prediction**: This component consumes the representations obtained from previous components to predict the long form l' for the acronym w_p in the sentence. In this section, we describes these components in details.

3.1 Sentence Encoder

We represent the word w_i of W using its corresponding pre-trained word embedding. Furthermore, the word embedding w_i is concatenated with the POS tag embedding of w_i to obtain the representation vector e_i for w_i. In order to create more abstract representations of the words and to encode the sequential order of the words in the sentence, we exploit the popular recurrent architecture encoder (i.e., Bidirectional Long Short-Term Memory (BiLSTM)) to consume the word representations $E = e_1, e_2, \ldots, e_n$ to generate abstract representations $H = h_1, h_2, \ldots, h_n$. More specifically, the abstract representation h_i is obtained by concatenating the corresponding hidden states of the forward and backward LSTMs: $h_i = [\overrightarrow{h_i} : \overleftarrow{h_i}]$. The abstract representation H will be consumed by the next components in our model.

3.2 Context Encoder

While the representation generated by recurrent encoders such as BiLSTM could potentially capture the context of the entire sentence to represent each word w_i, in practice, these models fail to encode long dependencies due to vanishing gradient issue. This is problematic as the actual meaning of the words, especially the acronym w_p, might depends on some words appeared in far distances from the word itself. In order to address this limitation, prior work in other NLP tasks has shown that the syntactical structure (e.g., dependency trees) could substantially improve the range of contextual information encoded in word representations. Unfortunately, none of the existing deep learning models for AD exploits the dependency trees of the sentences to enrich the contextual information encoded in the word representations. Thus, in this work, we propose to leverage the syntactical structure available in the dependency

	SciAD	UAD	SciUAD
Number of acronyms	732	567	538
average number of long form per acronym	3.1	2.3	1.8
overlap between sentence and long forms	0.32	0.01	0.31
average sentence length	30.7	18.55	50.30

Table 1: Comparison of different AD datasets. Note that the third row shows the ratio of sentences that have at least one word in common with the long forms of the acronyms appearing in the sentence.

tree to augment the representations of the words with syntactical contextual information. More specifically, the dependency tree of W is modeled as an undirected graph with adjacency matrix A where $A_{i,j}$ is 1 if the word w_i is the head or one of the children of the word w_j in the dependency tree. Note that we add self-loops to the dependency tree by adding the identity matrix I to A: $\hat{A} = A + I$. Afterwards, in order to encode the structure A into the node (i.e., word) representations, we employ graph convolutional neural networks (GCN) (Kipf and Welling, 2016; Pouran Ben Veyseh et al., 2019). Concretely, we first initialize the node representations with the vectors H generated by BiLSTM and then at the m-th layer of the GCN, we update the representation of i-th node by: $h_i^m = \sigma(W_m \cdot \frac{1}{\deg(i)} \Sigma_{j \in \mathcal{N}(i)} h_j^{m-1})$, where W_m is the weight matrix at layer m, σ is a non-linear activation function and $\mathcal{N}(i)$ and $\deg(i)$ are the set of neighbors and the degree of the i-th node in the adjacency matrix $\hat{A}$, respectively. The word representations at the final layer of the GCN, i.e., $H^s = h_1^s, h_2^s, \ldots, h_n^s$, are then employed as the structure-aware representations to be consumed by the next component.

3.3 Prediction

The prediction component is the last layer in our model which employs the representation obtained from the BiLSTM, i.e., H, and the GCN, i.e., H^s, to predict the true long form of the acronym w_p in sentence W. To this end, the representation of the acronym w_p from BiLSTM, i.e., h_i, and GCN, i.e., h_i^c, are concatenated to capture both the sequential and structural context of the word w_p. In addition, we also concatenate the representations of the entire sentence obtained from BiLSTM and GCN to enrich the acronym representation. To create the sentence representations, we employ max pooling over all words: $H_{sent} = MAX\_POOL(h_1, h_2, ..., h_n)$ and $H^s_{sent} = MAX\_POOL(h^s_1, h^s_2, ..., h^s_n)$. Thus, the final representation vector for prediction will be: $V = [h_p : h^s_p : H_{sent} : H^s_{sent}]$, where ":" indicates concatenation. Finally, a two-layer feed forward classifier is employed to predict long form l'. Note that the number of neurons in the last layer of the feed forward classifier is equal to the total number of long forms of all acronyms in $\mathcal{D}$. We use negative log-likelihood as the loss function to train the model: $L = -P(l|W, p)$. We call this model graph-based acronym disambiguation (GAD).

4 Experiments

Datasets & Evaluation Metrics: We evaluate the performance of the state-of-the-art acronym identification and disambiguation models on SciAI and SciAD, respectively. For AD, in addition to SciAD, we evaluate the performance of the models on the UAD Wikipedia dataset proposed by Ciosici (2019). UAD consists of sentences in general domain (i.e., Wikipedia). As the domain difference between SciAD and UAD (i.e., scientific papers vs Wikipedia articles) could effect on the direct comparison between the performances of the models, we also prepare SciUAD dataset for acronym disambiguation[4]. SciUAD employs the same corpus as we use to create SciAD but the acronyms, long forms, their mappings, and the ambiguous use of acronyms in corpus are all extracted using the unsupervised method proposed by UAD (Ciosici et al., 2019). Table 1 compares all AD datasets using different criteria. There are several observations from this table. First, SciAD supports more ambiguous acronyms and the level of ambiguity is higher in SciAD as there are more long forms per acronym. This is significant, specially considering the fact that SciAD is prepared from a smaller corpus than UAD. Also, comparing the level of ambiguity

[4]Note that we do not use NOA dataset (Charbonnier and Wartena, 2018) as it has only 4 samples per each long form on average and it is not suitable for supervised baselines. So, it cannot be comparable with the other datasets in our experiments.

Model	Acronym			Long Form			
	P	R	F1	P	R	F1	Macro F1
NOA	80.31	18.08	29.51	88.97	14.01	24.20	26.85
ADE	79.28	86.13	82.57	98.36	57.34	72.45	79.37
UAD	86.11	91.48	88.72	96.51	64.38	77.24	84.09
BIOADI	83.11	87.21	85.11	90.43	73.79	77.49	82.35
LNCRF	84.51	90.45	87.37	95.13	69.18	80.10	83.73
LSTM-CRF	88.58	86.93	87.75	85.33	85.38	85.36	86.55
Human Performance	98.51	94.33	96.37	96.89	94.79	95.82	96.09

Table 2: Performance of models in acronym identification (AI)

Model	SciAD			UAD			SciUAD		
	P	R	F1	P	R	F1	P	R	F1
MF	89.03	42.2	57.26	76.37	46.34	57.68	91.32	45.21	60.47
ADE	86.74	43.25	57.72	83.56	44.01	57.65	85.90	42.57	56.92
NOA	78.14	35.06	48.40	76.93	42.79	54.99	79.23	36.76	50.21
UAD	89.01	70.08	78.37	90.82	92.33	91.03	90.23	72.43	80.35
BEM	86.75	35.94	50.82	75.33	44.52	55.96	85.99	37.24	51.97
DECBAE	88.67	74.32	80.86	95.23	93.74	94.48	90.11	75.13	81.94
GAD	89.27	76.66	81.90	96.06	94.37	95.21	91.12	77.08	83.51
Human Performance	97.82	94.45	96.10	98.13	96.32	97.21	98.02	95.43	96.70

Table 3: Performance of models in acronym disambiguation (AD)

in SciAD and SciUAD indicates that our dataset preparation is more effective to capture acronym ambiguity in the given corpus than UAD. Second, comparing the ratio of sentences that have overlap with the long forms emphasizes the difference between scientific domain and general domain. The higher overlap ratio of SciAD and SciUAD compared to UAD could be attributed to two characteristics of the scientific domain: 1) Sentences in scientific papers are normally longer than sentences of general articles and it is corroborated by comparison between average sentence length in the three datasets shown in Table 1. 2) Long forms in scientific domain are more semantically related to each other, therefore they share more vocabulary in the context of their acronyms. This shows that AD could be more challenging on scientific domain than general domain. For more statistics, see Appendix B.2.

To create training and testing splits of the datasets, we randomly divide SciAI, SciAD and UAD into training, development and test data using 80:10:10 ratio. Regarding the evaluation metrics, for AI, a prediction of acronym or long form is counted as true if the boundaries of the prediction matches with the boundary of the ground-truth acronym or long form in the sentence, respectively. We report the macro-averaged precision, recall and F1 score computed for acronym and long form prediction. For AD, similar to prior work (Ciosici et al., 2019), we report the performance of the models using macro-averaged precision, recall and F1 score computed for each long form.

Baselines: For acronym identification, we compare the performance of the following baselines on SciAI: (1) **Rule-based methods**: These models employ manually designed rules and regular expressions to extract acronyms and long forms in text; namely we report the performance of NOA (Charbonnier and Wartena, 2018), UAD (Ciosici et al., 2019) [5] and ADE (Li et al., 2018), (2) **Feature-based models**: These models define a set of features for acronym and long form predictions, then they train a classifier (e.g., SVM or CRF) using these features, namely we report the performance of BIOADI (Kuo et al., 2009) and LNCRF (Liu et al., 2017), (3) **Deep learning models**: Since to the best of our knowledge, none of the existing work leverage pre-trained word embeddings with deep architectures for AI, we implement an LSTM-CRF baseline for this task. For more details on the LSTM-CRF baseline and hyper parameters, see Appendices A and B.

For acronym disambiguation, we compare the performance of the proposed model, i.e., GAD, with the following baselines: (1) **Non-deep learning models**: This category includes two models: (a) "most frequent" (MF) which takes the long form with the highest frequency among all possible meanings of an acronym as the expanded form of the acronym, and (b) a feature-based model that employs hand crafted

[5]Note that UAD employs the rules proposed by (Schwartz and Hearst, 2002)

Error Type	Share	Examples
Popular Acronyms	16%	... perspective projection or **3D** Thin Plate Spline (TPS) ...
		... the temporal resolution (TR) is 720 **ms**
Meaningful Acronyms	8%	... the **Cost** library and the **Sense** simulator ...
		For the **Home** dataset, we compare simplified ...
Embedded Acronyms	18%	... translation on **Europarl** (EP) and News domains ...
		... linear **SVM** (LSVM) and Radial Basis **SVM** (RSVM) classifiers ...
Lack of Expertise	5%	... current **malware landscape** and allow re-training the **ML** systems ...
Multiple Acronyms	24%	Since online **API** access could ... in the **API** access

Table 4: Errors in Annotations. Words shown in bold are missed from annotation except for *Lack of Expertise* error, where the acronym is mapped to a wrong long form

features from the context of the acronyms to train a disambiguation classifier; namely ADE (Li et al., 2018), (2) **Deep learning models**: In this category, we report the performance of (1) language-model-based baselines that train the word embeddings using the training corpus, namely NOA (Charbonnier and Wartena, 2018) and UAD (Ciosici et al., 2019), and (2) models employing deep architectures (e.g., LSTM), namely we report performance of DECBAE (Jin et al., 2019) and BEM (Blevins and Zettlemoyer, 2020)[6].

Table 2 and 3 show the results. There are several important observations from these tables. First, for AI, all rule-based and feature-based methods have higher precision for long form prediction than those for acronym prediction. This is due to the conservative nature of the rules/features exploited for finding long forms. On the other hand, these rules/features fail to capture all patterns of expressing the meanings of the acronym, resulting in poorer recall on long forms compared to acronyms. This is in contrast to the deep learning model LSTM-CRF as it has comparable recall on long forms and acronyms, showing the importance of pre-trained word embeddings and deep architectures for AI. Moreover, the performance of all AI baselines fall far behind human level performance, indicating that more research is required to fill this gap. Second, for Acronym Disambiguation task, except for MF and ADE, all baselines perform substantially better on UAD than SciAD; showing that SciAD is more challenging than UAD. In addition, while the better performance of the baselines on UAD than those on SciUAD corroborates the more challenging nature of AD in the scientific domain, the better performance of the baselines on ScieUAD than those on SciAD shows that the manual annotation for the AD dataset in this paper is more effective than those in UAD. Third, among all the baselines, the proposed GAD achieves the best results on all three datasets, showing the importance of syntactic structure for AD. However, all baselines are still far less effective than human on SciAD, thereby providing many research opportunities on this dataset.

5 Analysis

This sections provides further analysis on the annotation process and the proposed datasets and model. We first discuss the common errors in annotations that result in conflicts. Afterwards, we provide a sample complexity analysis of the AD task. Case study is presented in the end.

5.1 Annotation Errors

In order to prevent conflicts in annotations, we first conduct a pilot study to analyze the common errors that annotators could make during the annotations, then we update the annotation instructions and interface accordingly to prevent these errors. More specifically, we create HITs for 5% of the sentences in the prepared corpus and we manually analyze the conflicts. Table 4 shows the most frequent errors by the annotators in the annotation process that create the conflicts. Note that these errors represent 75% of the conflicts and the other 25% of the conflicts are due to mis-annotations with other errors (e.g., careless annotation). Among all errors, multiple acronyms (i.e., failing to annotate all occurrences of an acronym

[6]Consider that BEM is originally proposed for word sense disambiguation. As there is no glossary for the long forms, we use the words of the long form as the its glossary

Model	SciAD			UAD			UAD_{small}		
	P	R	F1	P	R	F1	P	R	F1
DECBAE	88.67	74.32	80.86	95.23	93.74	94.48	89.01	93.66	91.28
GAD	89.27	76.66	81.90	96.06	94.37	95.21	90.78	94.59	92.64

Table 5: Performance of models on the small version of UAD compared to the original UAD and SciAD daataset in AD task

Sentence	Long Form extracted by Rules	Model
It uses **fast and factual exploration** (**FA2E**) to perform ...	factual exploration	UAD
results from **overloaded homogeneous preparation** (**OMP**) for ...	homogeneous preparation	UAD
we first use **complete polar evaluation** (**CARE**) as ...	N/A	ADE
which uses **local rewriting pattern** (**LWP**) features ...	N/A	ADE
... where **genetic algorithms** come into picture, denoted by **GA** in ...	N/A	UAD/ADE

Table 6: Failures of rule-based methods for extracting acronyms and their long forms from text

in a sentence), embedded acronyms (acronyms that are part of a long form), and popular acronyms (i.e., acronyms such as 3D (3 dimensions) or ms (millisecond)) contribute the most to the conflicts. To reduce these errors and prevent conflicts, we update the instructions in the annotation interface to explicitly address confusions that result in these errors and provide examples to the annotators. In addition to that, we include test cases prone to each of these errors in the qualification test. Based on our evaluations, compared to the pilot study, these modifications result in 43% reduction in conflict rates during the main annotation process. Details of the annotation instructions and interface are provided in Appendix C.

5.2 Sample Complexity Analysis

As it is shown in Table 7 in Appendix B.2, UAD dataset is almost one order of magnitude larger than SciAD. This larger size results in more training samples per meaning. More specifically, in SciAD, there are 17 training samples per each long form, while this number is 126 for UAD. So, in this analysis, we tend to answer the question that whether the better results of the baselines on UAD is due to its larger size or its less challenging nature than SciAD. To this end, we prepare a small version of the UAD dataset by keeping only 17 training samples for each long form. Using this criterion, the size of the UAD training set reduces to 22,522. We name this datase as UAD_{small}. Note that we do not change the size of development or test set. Afterwards, we retrain the DECBAE and GAD models on the new small UAD training set. The results of evaluation of re-trained models on UAD test set are shown in Table 5. For convenience, we also report the results on the original UAD and SciAD datasets. This table shows that training on the UAD_{small} results in performance loss on UAD test set, however, both models still perform considerably better on UAD_{small} than SciAD. It empirically proves that SciAD is more challenging than UAD as the same models trained on datasets with comparable size achieve higher performance on UAD_{small} than SciAD. In addition, this table also shows that in the small version of UAD, i.e., UAD_{small}, GAD again outperforms DECBAE, indicating its superiority for AD.

5.3 Case Study

In this section we study the cases in which the acronym identification baselines fail to correctly extract the long form of the identified acronyms. Studying this type of failure is important because it contributes the most to the low recall of the baselines for long form identification. Note that most of the recent work employs rule-based methods for long form identification, so we analyze the failures of two baselines UAD (Ciosici et al., 2019) and ADE (Li et al., 2018). The results are shown in Table 6. Note that UAD looks for a sequence of letters in the words preceding the acronym which can make the capital letters in the acronym. Thus, it fails in the first two examples. The failures of the ADE are mainly due to the use of characters in the middle or end of the words in the long forms to create the acronyms. In the last example, both methods fail as the long form is far away from the acronym so they do not capture it.

In order to provide more insight into the acronym disambiguation challenges in SciAD, we study the failures and successes of the proposed model GAD. For the success cases, we study sentences in which

the other baselines fail to correctly disambiguate the given acronym but GAD can successfully predict the long form. Consider this sentence: "*Words that are not compatible with our pre-defined rules are excluded from* ***SDP***". In this example, the true long form of "*SDP*" is "*shortest dependency path*" and the token "*Words*" in the beginning of the sentence is a clue to disambiguate this acronym. Due to the long distance between "*SDP*" and "*Words*", the baselines that encode the sentence sequentially fail to capture their connection. On the other hand, as GAD is equipped with the dependency tree and it benefits from the short dependency path between "*SDP*" and "*Words*". As another example, consider this sentence: "***SDP**, SP, and GT together with the models proposed in section 4 and the simplified version of SLD are the major optimization baselines in our experiments*". In this example, the expanded form of "*SDP*" is "*semi-definite programming*". While the other baselines fail to correctly predict the true long form, GAD successfully predicts it. The success of the GAD could be attributed to the short distance between "*SDP*" and the word "*optimization*" in the dependency tree (see Figure 6). On the other hand, due to the long sequential distance between "*SDP*" and "*optimization*" in the sentence, none of the other baselines is able to capture the connection between these two words.

Despite the improvement obtained from utilizing dependency tree for acronym disambiguation, there are still cases that some misleading words in the sentence could obfuscate disambiguation. Note that in these examples both GAD and the baselines fail, indicating the necessity of more advanced models for this task. For instance consider this example: "θ *represents the parameters that determine the optimal splitting node of* ***RF*** *in our regression model*". In this sentence the true long form of the acronym "*RF*" is "*Random Forest*" but the GAD model wrongly predicts the long form "*Regression Functions*". This failure could be attributed to the existence of the misleading word "*regression*" in the sentence that highly correlates to the vocabulary of the contexts of the long form "*Regression Functions*" This high correlation downgrades the importance of the related word "*node*" which is useful to disambiguate this acronym.

6 Related Work

In the last two decades, several work has been proposed for acronym identification and disambiguation. For AI, most of the prior work employs rule-based methods (Park and Byrd, 2001; Wren and Garner, 2002; Schwartz and Hearst, 2002; Adar, 2004; Nadeau and Turney, 2005; Ao and Takagi, 2005; Kirchhoff and Turner, 2016) or feature engineering (Kuo et al., 2009; Liu et al., 2017). Majority of the recent work leverages the rules proposed by (Schwartz and Hearst, 2002) to extract long forms from the sentences containing acronyms. Recently, it has been shown that the rule-based methods fall far behind human level performance, especially in scientific domain (Harris and Srinivasan, 2019). In addition to the rule-based methods, some work utilizes users' search experience to identify the meanings of acronyms (Jain et al., 2007; Nadeau and Turney, 2005; Taneva et al., 2013). However, this method cannot be applied to non-web-based resources (e..g, scientific papers). For acronym disambiguation, prior work employs either feature-based models (Wang et al., 2016; Li et al., 2018) or deep learning models. While some of the deep learning models directly employ word embeddings for disambiguation (Wu et al., 2015; Antunes and Matos, 2017; Charbonnier and Wartena, 2018; Ciosici et al., 2019), some of them employ deep architectures to encode the context of the acronym (Jin et al., 2019; Li et al., 2019). Moreover, acronym disambiguation has been also modeled as the more general tasks Word Sense Disambiguation (WSD) (Henry et al., 2017; Tulkens et al., 2016) or Entity Linking (EL) (Cheng and Roth, 2013; Li et al., 2015). While the majority of the prior work studies AD in medical domain (Okazaki and Ananiadou, 2006; Vo et al., 2016; Wu et al., 2017), recently some work proposes acronym disambiguation in general (Ciosici et al., 2019), enterprise (Li et al., 2018), or scientific domain (Charbonnier and Wartena, 2018).

7 Conclusion

We introduce new datasets and strong baselines for AI and AD sub-tasks. These datasets are created based on human annotation and shown to be more challenging than their previous counterparts. While the baselines achieve good performance, especially the proposed GAD model which outperforms the prior models for AD, more research is required to reach to human level performance for the new datasets.

References

Christopher F Ackermann, Charles E Beller, Stephen A Boxwell, Edward G Katz, and Kristen M Summers. 2020. Resolution of acronyms in question answering systems, February 25. US Patent 10,572,597.

Eytan Adar. 2004. Sarad: A simple and robust abbreviation dictionary. *Bioinformatics*, 20(4):527–533.

Rui Antunes and Sérgio Matos. 2017. Biomedical word sense disambiguation with word embeddings. In *International Conference on Practical Applications of Computational Biology & Bioinformatics*, pages 273–279. Springer.

Hiroko Ao and Toshihisa Takagi. 2005. Alice: an algorithm to extract abbreviations from medline. *Journal of the American Medical Informatics Association*, 12(5):576–586.

Terra Blevins and Luke Zettlemoyer. 2020. Moving down the long tail of word sense disambiguation with gloss informed bi-encoders. *ACL*.

Jean Charbonnier and Christian Wartena. 2018. Using word embeddings for unsupervised acronym disambiguation. In *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796.

Manuel R Ciosici, Tobias Sommer, and Ira Assent. 2019. Unsupervised abbreviation disambiguation. *arXiv preprint arXiv:1904.00929*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Christopher G Harris and Padmini Srinivasan. 2019. My word! machine versus human computation methods for identifying and resolving acronyms. *Computación y Sistemas*, 23(3).

Sam Henry, Clint Cuffy, and Bridget McInnes. 2017. Evaluating feature extraction methods for knowledge-based biomedical word sense disambiguation. In *BioNLP 2017*, pages 272–281.

Rezarta Islamaj Dogan, G Craig Murray, Aurélie Névéol, and Zhiyong Lu. 2009. Understanding pubmed® user search behavior through log analysis. *Database*, 2009.

Alpa Jain, Silviu Cucerzan, and Saliha Azzam. 2007. Acronym-expansion recognition and ranking on the web. In *2007 IEEE International Conference on Information Reuse and Integration*, pages 209–214. IEEE.

Qiao Jin, Jinling Liu, and Xinghua Lu. 2019. Deep contextualized biomedical abbreviation expansion. *arXiv preprint arXiv:1906.03360*.

Dongyeop Kang, Andrew Head, Risham Sidhu, Kyle Lo, Daniel S Weld, and Marti A Hearst. 2020. Document-level definition detection in scholarly documents: Existing models, error analyses, and future directions. *arXiv preprint arXiv:2010.05129*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Katrin Kirchhoff and Anne M Turner. 2016. Unsupervised resolution of acronyms and abbreviations in nursing notes using document-level context models. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*, pages 52–60.

Klaus Krippendorff. 2011. Computing krippendorff's alpha-reliability.

Cheng-Ju Kuo, Maurice HT Ling, Kuan-Ting Lin, and Chun-Nan Hsu. 2009. Bioadi: a machine learning approach to identifying abbreviations and definitions in biological literature. In *BMC bioinformatics*, volume 10, page S7. Springer.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Chao Li, Lei Ji, and Jun Yan. 2015. Acronym disambiguation using word embedding. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Yang Li, Bo Zhao, Ariel Fuxman, and Fangbo Tao. 2018. Guess me if you can: Acronym disambiguation for enterprises. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1308–1317, Melbourne, Australia, July. Association for Computational Linguistics.

Irene Li, Michihiro Yasunaga, Muhammed Yavuz Nuzumlalı, Cesar Caraballo, Shiwani Mahajan, Harlan Krumholz, and Dragomir Radev. 2019. A neural topic-attention model for medical term abbreviation disambiguation. *arXiv preprint arXiv:1910.14076*.

Jie Liu, Jimeng Chen, Yi Zhang, and Yalou Huang. 2011. Learning conditional random fields with latent sparse features for acronym expansion finding. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 867–872.

Jie Liu, Caihua Liu, and Yalou Huang. 2017. Multi-granularity sequence labeling model for acronym expansion identification. *Information Sciences*, 378:462–474.

David Nadeau and Peter D Turney. 2005. A supervised learning approach to acronym identification. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 319–329. Springer.

Ankit Nautial, Nagesh Bhattu Sristy, and Durvasula VLN Somayajulu. 2014. Finding acronym expansion using semi-markov conditional random fields. In *Proceedings of the 7th ACM India Computing Conference*, pages 1–6.

Naoaki Okazaki and Sophia Ananiadou. 2006. Building an abbreviation dictionary using a term recognition approach. *Bioinformatics*, 22(24):3089–3095.

Youngja Park and Roy J Byrd. 2001. Hybrid text mining for finding abbreviations and their definitions. In *Proceedings of the 2001 conference on empirical methods in natural language processing*.

Rebecca Passonneau. 2006. Measuring agreement on set-valued items (masi) for semantic and pragmatic annotation. In *LREC*.

Amir Pouran Ben Veyseh, Thien Huu Nguyen, and Dejing Dou. 2019. Graph based neural networks for event factuality prediction using syntactic and semantic structures. In *ACL*.

Roman Prokofyev, Gianluca Demartini, Alexey Boyarsky, Oleg Ruchayskiy, and Philippe Cudré-Mauroux. 2013. Ontology-based word sense disambiguation for scientific literature. In *European conference on information retrieval*, pages 594–605. Springer.

Ariel S Schwartz and Marti A Hearst. 2002. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Biocomputing 2003*, pages 451–462. World Scientific.

Bilyana Taneva, Tao Cheng, Kaushik Chakrabarti, and Yeye He. 2013. Mining acronym expansions and their meanings using query click log. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1261–1272.

Aditya Thakker, Suhail Barot, and Sudhir Bagul. 2017. Acronym disambiguation: A domain independent approach. *arXiv preprint arXiv:1711.09271*.

Stéphan Tulkens, Simon Šuster, and Walter Daelemans. 2016. Using distributed representations to disambiguate biomedical and clinical concepts. *arXiv preprint arXiv:1608.05605*.

Amir Pouran Ben Veyseh, Franck Dernonrcourt, and Thien Huu Nguyen. 2020. Improving slot filling by utilizing contextual information. In *Workshop on natural language processing for conversational AI, ACL*.

Amir Pouran Ben Veyseh. 2016. Cross-lingual question answering using common semantic space. In *Proceedings of TextGraphs-10: the workshop on graph-based methods for natural language processing*, pages 15–19.

Thi Ngoc Chau Vo, Tru Hoang Cao, and Tu Bao Ho. 2016. Abbreviation identification in clinical notes with level-wise feature engineering and supervised learning. In *Pacific Rim Knowledge Acquisition Workshop*, pages 3–17. Springer.

Yue Wang, Kai Zheng, Hua Xu, and Qiaozhu Mei. 2016. Clinical word sense disambiguation with interactive search and classification. In *AMIA Annual Symposium Proceedings*, volume 2016, page 2062. American Medical Informatics Association.

Jonathan D Wren and Harold R Garner. 2002. Heuristics for identification of acronym-definition patterns within text: towards an automated construction of comprehensive acronym-definition dictionaries. *Methods of information in medicine*, 41(05):426–434.

Yonghui Wu, Jun Xu, Yaoyun Zhang, and Hua Xu. 2015. Clinical abbreviation disambiguation using neural word embeddings. In *Proceedings of BioNLP 15*, pages 171–176.

Yonghui Wu, Joshua C Denny, S Trent Rosenbloom, Randolph A Miller, Dario A Giuse, Lulu Wang, Carmelo Blanquicett, Ergin Soysal, Jun Xu, and Hua Xu. 2017. A long journey to short abbreviations: developing an open-source framework for clinical abbreviation recognition and disambiguation (card). *Journal of the American Medical Informatics Association*, 24(e1):e79–e86.

Hua Xu, Peter D Stetson, and Carol Friedman. 2007. A study of abbreviations in clinical notes. In *AMIA annual symposium proceedings*, volume 2007, page 821. American Medical Informatics Association.

Hong Yu, Won Kim, Vasileios Hatzivassiloglou, and W John Wilbur. 2007. Using medline as a knowledge source for disambiguating abbreviations and acronyms in full-text biomedical journal articles. *Journal of biomedical informatics*, 40(2):150–159.

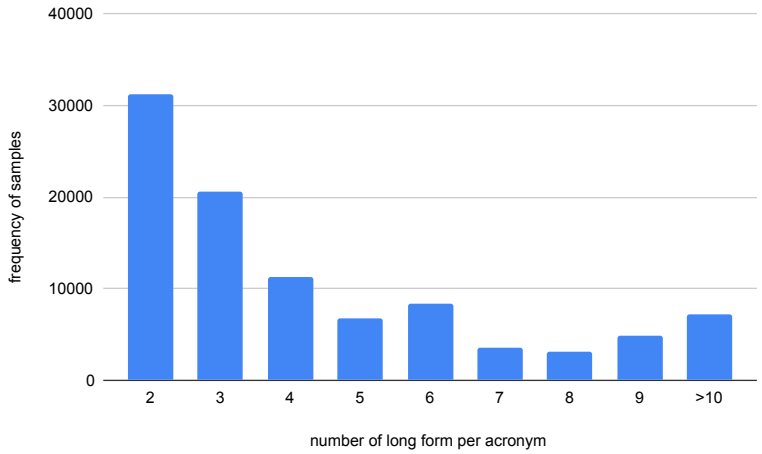

Figure 1: Distribution of samples based on number of long form per acronym

Dataset	Train	Dev.	Test
SciAI	14,006	1,750	1,750
SciAD	49,788	6,233	6,225
UAD	459,869	57,484	57,484
SciUAD	15,982	1,997	1,999

Table 7: Dataset Statistics. Numbers of samples per each data splits

A LSTM-CRF Baseline

This section describes the details of the LSTM-CRF employed in acronym identification experiments. Formally, given the input sentence $W = w_1, w_2, ..., w_n$, each word w_i is represented by e_i which is the concatenation of (1) the corresponding pre-trained word embedding, and (2) the embedding of the part-of-speech (POS) tag of the word w_i. Afterwards, the word representations $E = e_1, e_2, ..., e_n$ are fed into a bi-directional long short-term memory (BiLSTM) network to obtain more abstract representations $H = h_1, h_2, ..., h_n$. Specifically, the representation h_i is obtained by the concatenation of the hidden states of the forward and backward LSTMs at the corresponding time step. Next, the feature vector h_i is transformed into a score vector v_i whose dimensions correspond to the possible word labels/tags (i.e., the five BIO tags) and quantify the possibility for w_i to receive the corresponding labels: $v_i = W_S h_i$, where W_S is the trainable weight matrix and $|v_i| = 5$. In the next step, to capture the sequential dependencies between word labels, we exploit conditional random field (CRF) layer. Concretely, the score for a possible label sequence $\hat{L} = \hat{l}_1, \hat{l}_2, ..., \hat{l}_N$ for W would be:

$$Score(\hat{l}_1, \hat{l}_2, ..., \hat{l}_N | W) = \Sigma_{j=1}^{N} \left(v_{\hat{l}_j} + t_{\hat{l}_{j-1}, \hat{l}_j} \right) \quad (1)$$

where $t_{\hat{l}_{j-1}, \hat{l}_j}$ is the trainable transition score from label $\hat{l}_{j-1}$ to label $\hat{l}_j$. Note that the probability of the possible labeling $\hat{L}$, i.e., $P(\hat{l}_1, \hat{l}_2, ..., \hat{l}_N | W)$, is computed using dynamic programming (Lafferty et al., 2001). We use the negative log-likelihood $\mathcal{L}$ of the input example as the objective function to train the model:

$$\mathcal{L} = -\log P(l_1, l_2, ..., l_N | W) \quad (2)$$

where $L = l_1, l_2, ..., l_N$ is the golden label sequence for W. Finally, the Viterbi decoder is employed to infer the sequence of labels with highest score for the input sentence.

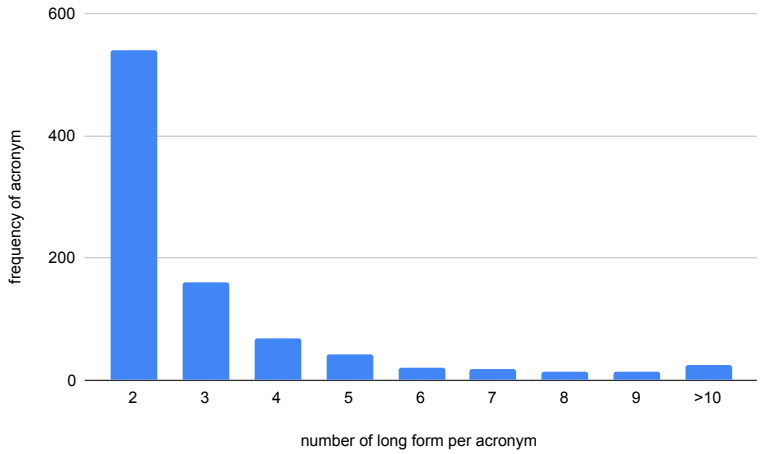

Figure 2: Distribution of acronyms based on number of long form per acronym

B Hyper Parameters and Dataset Statistics

B.1 Hyper Parameters

In our experiments, for the rule-based, feature-based and language-model-based models, i.e., NOA, ADE, UAD, BIOADI and LNCRF, we use the same hyper parameters and features reported in the original papers. For the deep learning models, i.e., LSTM-CRF, BEM, DECBAE and GAD, we fine tune the hyper parameters using the performance on the development set for each task. More specifically, we find the following hyper parameters for the deep learning models: 50 samples per each mini-batch; 200 hidden dimensions for all feed forward and BiLSTM layers; dropout rate 0.2; two layers of BiLSTM and GCN; and Adam optimizer with learning rate 0.3. Note that for pre-trained word embeddings we use the uncased version of BERT$_{base}$ (Devlin et al., 2018) with 768 dimensions.

B.2 Dataset Statistics

For statistics of the datasets used in our experiments, see Table 7. This table shows the number of samples per different data splits of all datasets used in our experiments. In addition, Figures 1, 2 and 3 demonstrate more statistics of SciAD dataset. More specifically, Figure 1 shows the distribution of number of samples based on number of long form per acronym. The distribution shown in this figure is consistent with the same distribution presented in the prior work (Charbonnier and Wartena, 2018) in which in both distributions, acronyms with 2 or 3 meanings have the highest number of samples in the dataset. Figure 2 shows the distribution of number of acronyms based on number of long forms per acronym. Again, this distribution is compatible with the distribution of samples, i.e., Figure 1, as acronyms with 2 or 3 meanings have the highest frequency in the dataset. Finally, Figure 3, depicts the number of long forms with either less or more than 10 samples in the dataset. This figure shows that the number of high-frequent and low-frequent long forms are on par with each other in our dataset, so it provides opportunity for future research to train and evaluate acronym disambiguation models on both categories of long forms (i.e., high-frequent and low-frequent long forms)

B.3 Comparison with the Existing AI and AD Datasets

This sections compares SciAI and SciAD with the existing acronym identification and disambiguation datasets. Table 8 compares SciAI with the manually labeled non-medical acronym identification datasets proposed and studied by the prior work. This table shows that our dataset is substantially larger than the existing datasets in various criteria, including the number of sentences present in the dataset and the number of distinct acronyms and long forms. Moreover, we publicly release SciAI to provide more research opportunity for acronym identification in scientific domain. For acronym disambiguation, we compare our model with the existing AD datasets in scientific domain. The results are shown in Table 9.

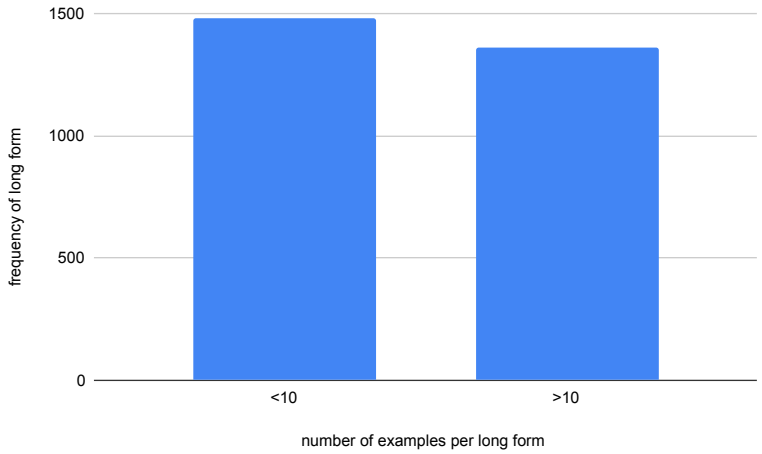

Figure 3: Distribution of long forms based on number of samples per long form

This table shows that our proposed dataset is considerably larger than the existing datasets. Especially considering the average number of samples per long form, our dataset is more suitable for advanced deep learning models that is also corroborated in our experiments.

Dataset	Size	# Unique Acronyms	# Unique Meaning	# Documents	Publicly Available	Domain
LSAEF (Liu et al., 2011)	6,185	255	1,372	N/A	No	Wikipedia
AESM (Nautial et al., 2014)	355	N/A	N/A	N/A	No	Wikipedia
MHIR (Harris and Srinivasan, 2019)	N/A	N/A	N/A	50	No	Scientific Papers
MHIR (Harris and Srinivasan, 2019)	N/A	N/A	N/A	50	No	Patent
MHIR (Harris and Srinivasan, 2019)	N/A	N/A	N/A	50	No	News
SciAI (ours)	17,506	7,964	9,775	6,786	yes	Scientific Papers

Table 8: Comparison of non-medical manually annotated acronym identification datasets. Note that size refers to the number of sentences in the dataset.

Dataset	Size	Annotation	Avg. Number of Samples per Long Form
Science WISE (Prokofyev et al., 2013)	5,217	Disambiguation mannualy annotated	N/A
NOA (Charbonnier and Wartena, 2018)	19,954	No manual annotation	4
SciAD (ours)	62,441	Acronym identification manually annotated	22

Table 9: Comparison of scientific acronym disambiguation (AD) datasets. Note that size refers to the number of sentences in the dataset.

C Annotation Instructions and Interface

Annotation instruction is shown in Figure 4. This instruction is shown directly above the annotation interface so that the annotators could access it before attempting to annotate. Note that while we ask the annotators to annotate embedded acronyms (e.g., "*svm*" in "*linear svm (LSVM)*"), but in the released dataset we keep the long form label for the embedded acronyms to make it consistent with sequence labeling framework for AI. In addition to the instruction shown to the annotators, we also require the annotators to pass a qualification test with the same interface. Only the annotators with 100% score from the qualification test are allowed to accept the actual HITs.

An example of annotation interface is shown in Figure 5. In this example the annotator has selected "*high level protocol specification language*" as the long form and "*HLPSL*" as the short form (i.e., shown at the bottom of the page). In addition, the annotator has mapped the short form "*HLPSL*" to the long form "*high level protocol specification language*" (shown in the middle section). Note that when the annotators switch the button from "*short forms (acronyms)*" to "*Long Forms (phrases)*" (i.e., the green buttons at the top of the page), instead of the short forms, the long forms will be highlighted in the text box.

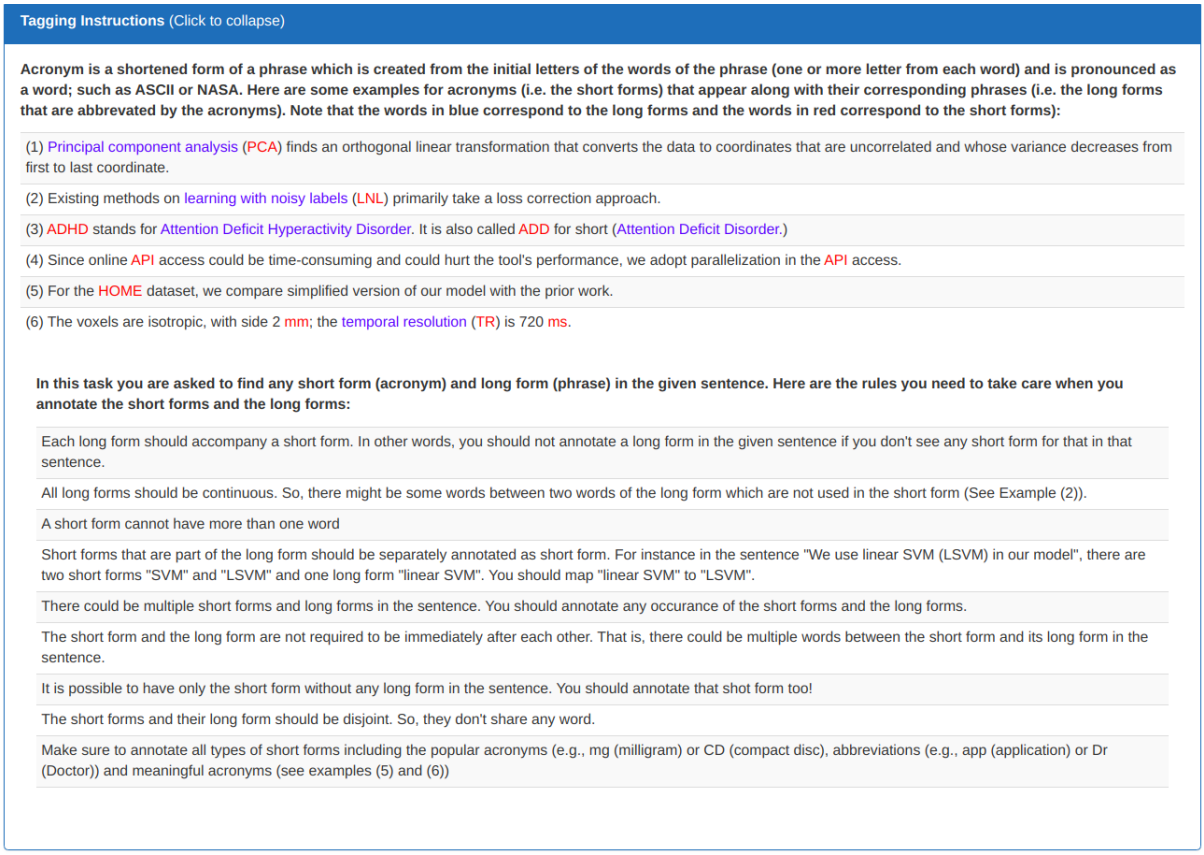

Figure 4: Annotation instructions shown to the annotators

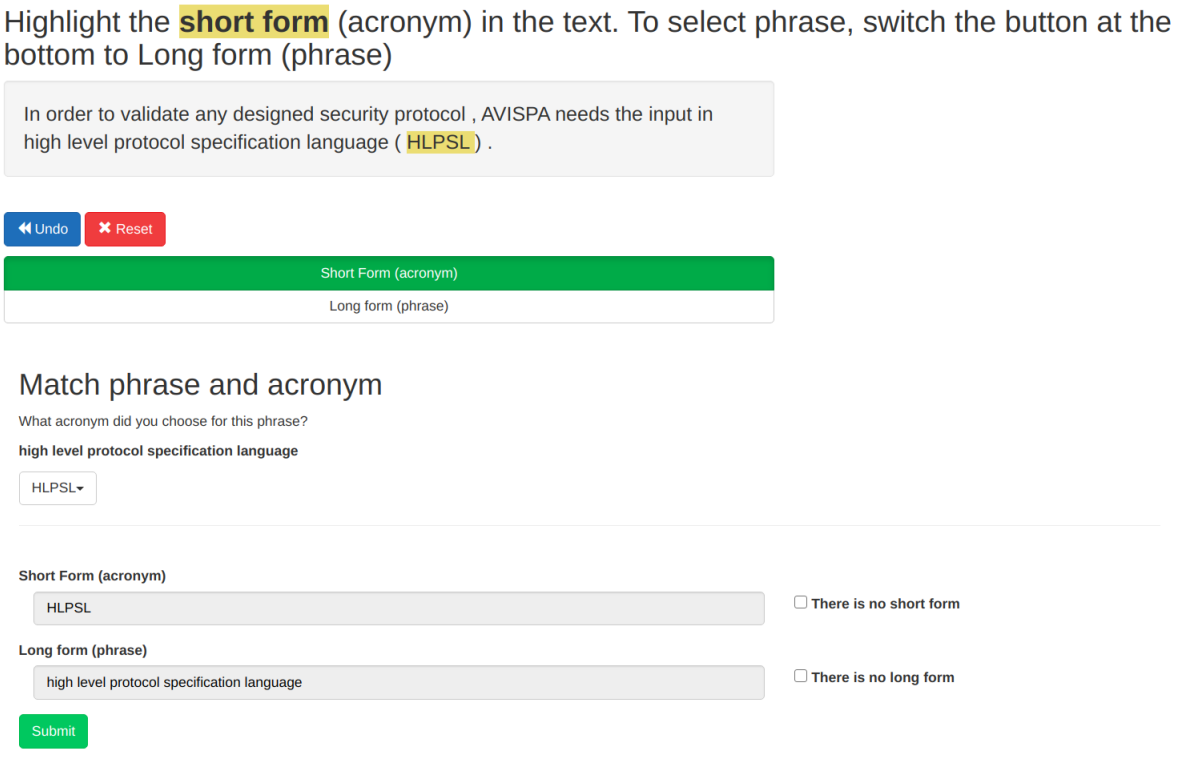

Figure 5: An example of annotation interface.

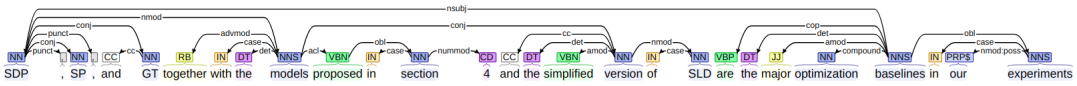

Figure 6: Dependency tree of the sentence "*SDP, SP, and GT together with the models proposed in section 4 and the simplified version of SLD are the major optimization baselines in our experiments*"