AcX: system, techniques, and experiments for Acronym eXpansion [Experiments, Analysis & Benchmarks]

João L. M. Pereira
INESC-ID, IST Universidade de Lisboa, and
University of Amsterdam
joaoplmpereira@tecnico.ulisboa.pt

João Casanova
INESC-ID, IST Universidade de Lisboa, and
Hitachi Vantara
joao.casanova@tecnico.ulisboa.pt

Helena Galhardas
INESC-ID and IST Universidade de Lisboa
hig@inesc-id.pt

Dennis Shasha
Courant Institute, New York University
shasha@cs.nyu.edu

Abstract

In this information-accumulating world, each of us must learn continuously. To participate in a new field, or even a sub-field, one must be aware of the terminology including the acronyms that specialists know so well, but newcomers do not.

This paper describes an end-to-end acronym expander system called *AcX* that takes a document, identifies its acronyms, and suggests expansions that are either found in the document or appropriate given the subject matter of the document. As far as we know, AcX is the first open source and extensible system for acronym expansion that allows mixing and matching of different inference modules. As of now, AcX works for English, French, and Portuguese with other languages in progress.

In addition to describing the design and implementation of AcX, this paper proposes *three new acronym expansion benchmarks* (for in-expansion techniques, for out-expansion techniques and for end-to-end systems). This paper applies state-of-the-art techniques to the proposed benchmarks to find a set of promising approaches for acronym expansion. Finally, the paper evaluates the performance of AcX in end-to-end experiments (using the best techniques based on the benchmark measurements) on a human-annotated dataset of Wikipedia documents. When this paper is published, we will make all our (reproducible) code and data public domain and put a github link in the abstract.

Our experiments show that human performance is still better than the best automated approaches. Thus, achieving Acronym Expansion at a human level is still a rich and open challenge.

PVLDB Reference Format:
João L. M. Pereira, João Casanova, Helena Galhardas, and Dennis Shasha. AcX: system, techniques, and experiments for Acronym eXpansion [Experiments, Analysis & Benchmarks]. PVLDB, 15(1): XXX-XXX, 2022.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at http://web.tecnico.ulisboa.pt/ist164790/acx.

Proceedings of the VLDB Endowment, Vol. 15, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

1 Introduction

Take a great historical literary figure of any culture and put him or her in the present. That person might barely understand a newspaper headline partly because of the acronyms that have no expansion. For example, the headlines of the New York Times on the balmy and benign August 20, 2019 include "ISIS is regaining strength...", "Under mounting pressure, top C.E.Os", "S&P 500...". Even Shakespeare would be at a loss.

Contemporary scholars encounter similar challenges when entering a new field or a sister field. A typical document about wireless communication is practically unintelligible to a computer scientist with its talk of 3gPP, 5G. BS etc. Documents written for specialists often neglect even to define the acronyms they use [6].

Further, the proper expansion of an acronym depends on context. For example, "ISBN" can mean International Standard Book Number in a publishing context, Integrated Satellite Business Network in a satellite communication context, and International Society for Behavioral Neuroscience in a cognitive scientific context. Thus, any system that hopes to help readers understand the intended meaning of an undefined acronym in an document must expand that acronym using its context.

An *end-to-end acronym expander system* comprises the following two steps: (i) Extraction of each acronym and (when present) its expansion within a text. For example, if a given text has "ISBN (Integrated Satellite Business Network)" then ISBN would be the acronym and Integrated Satellite Business Network would be the expansion. We call this *in-expansion* because this can be done for a particular document based solely on its own text. (ii) In the case that an acronym is not expanded in the text of a document, *out-expansion* chooses an expansion from a large parsed corpus (training corpus) of other documents (e.g., Wikipedia).

This paper makes the following contributions:

- The **end-to-end Acronym eXpander (AcX) system** that accepts a text document as input and outputs a list of acronym-expansion pairs for the acronyms found in the document, whether or not the expansions are in the document.
- A **benchmark of in-expansion techniques (in-expansion benchmark)**. We make use of four biomedical datasets previously proposed in the literature (i.e., Medstract [24], BIOADI [24], Schwartz and Hearst [24], and Ab3p [24]) and one of the biggest sentence based datasets from the

scientific domain (i.e., SciAI [60]). Additionally, we have created a new dataset composed of Wikipedia documents from the *Computing* category. We calculate the precision, recall, and F1-measure for the extraction of both acronyms and acronym-expansion pairs. In addition, we measure training and execution times.

- A **benchmark of out-expansion techniques (out-expansion benchmark)**. We evaluate out-expansion techniques on three datasets from different domains previously used in related work that contain documents (i.e., MSH [47], SciWISE [47], and CS Wiki [57]) and one that is constructed from independent sentences from the scientific domain (i.e., SciAD [59] revised by Egan and Bohannon [17]).
  Some of the out-expansion techniques we consider have already been proposed in related work: Classic Context Vector [2, 35, 47], Surrounding Based Embedding [35], Thakker et al. [57], and Unsupervised Abbreviation Disambiguation [14]. Most recently, competitors of the SDU@AAAI competition [59] mainly used pre-trained language models based on Transformer neural networks like BERT [16] and SciBERT [7]. For purposes of out-expansion, we have adapted and evaluated (i) the out-expander of the MadDog system [58], (ii) SciDr [53] (which was originally hard coded to apply to the sentences of the SDU@AAAI competition, but we have extended it to multiple datasets of documents) (iii) techniques from Natural Language Processing (i.e., Term Frequency–Inverse Document Frequency [29], Latent Dirichlet Allocation [9], and Doc2Vec [34]).
  We have also embedded the outputs of a variety of out-expansion techniques as features for machine learning classifiers. The net result is that AcX is an extensible system able to create a new set of out-expansion pipelines out of combinations of user-chosen techniques.
- A **benchmark of end-to-end acronym expander systems (end-to-end benchmark)**. We create the first end-to-end dataset of human-annotated documents that includes both in- and out-expansions. We use this dataset to test the AcX end-to-end system. We use all English Wikipedia documents to train it. We compare the MadDog system [58] against different pipelines of AcX. We also compare AcX's performance to that of human annotators.
- **A link follower component** within AcX system that infers expansions based on the links of unexpanded acronyms in input documents. Link following improves the F1-measure of the best AcX pipeline identified by our end-to-end benchmark.

This paper is organized as follows: Section 2 presents related work, particularly techniques for in-expansion extraction and techniques for acronym out-expansion. Section 3 describes the AcX system and its components. The next three sections (Sections 4, 5 6) describe the proposed benchmarks and the corresponding analyses of the results obtained from the experiments performed in the context of each benchmark. Finally, in Section 7, we draw the main conclusions and ideas for future work.

2 Related Work

This section describes the work that is closely relevant to acronym expansion. Specifically, we start with in-expansion (acronym expansion extraction within documents) in Section 2.1. Then, we move on to out-expansion, which expands acronyms based on other documents and subject matter, in Section 2.2. Finally, we describe the few known end-to-end acronym expander systems in Section 2.3.

2.1 In-expansion

Pustejovsky et al. [48] present a technique that uses a robust parsing of the input text in order to reduce the context within which to search for a candidate expansion. Schwartz and Hearst [51] describe a technique that considers two possible placements of expansions and acronyms in text (before or after), and chooses the correct expansion by matching acronym characters with potential expansion characters.

MadDog [58] in-expander introduces minimal variations of the Schwartz and Hearst technique [51] and refines the candidate expansions using a sequence of rules where each one yields more precise expansions than the previous one. Nabeesath and Nazeer [50] suggest new pattern heuristics as well as space reduction heuristics. The technique of Azimi et al. [5] uses the same patterns as Schwartz and Hearst [51] but relaxes the heuristics for acronym and expansion extraction: an acronym simply needs to be a token composed of capital letters and an expansion composed of n tokens, with n corresponding to acronym length in characters.

The technique proposed by Yarygina and Vassilieva [63] makes use of user feedback and two decision tree classifiers in order to filter candidate acronym-expansion pairs. Glass et al. [20] proposed a technique that focuses on several languages other than English, and scores candidate pairs by using word embeddings in order to measure the similarity between candidate acronyms and expansions.

In the techniques of Liu et al. [37] and Veyseh et al. [60], the task of finding expansions for an acronym is formalized as a sequence labeling problem solved by Conditional Random Fields (CRFs) [33] based techniques. SciDr [53] in-expander and Zhu et al. [65] also interpret acronym and expansion extraction as a sequence labeling task and make use of pre-trained BERT-based models coupled with ensemble techniques to achieve higher model performance than previous techniques. SciBERT is a language model based on Transformers and pre-trained on research papers from Semantic Scholar [1]. SciBERT is fine-tuned in SciDr [53] with training data for the sequence labeling task. SciDr [53] in-expander uses a blending process [52]. It splits the training data into train and validation sets. Five different SciBERT models (e.g., number of epochs and learning rate values) are constructed based on the training set. Predictions on the validation set are then stored. The models, the predictions, and additional syntactic features extracted from the word-to-tag mapping are used to train 5 Conditional Random Fields (CRFs) [33] in a 5-fold cross-validation setting. The resulting CRF models are ensembled using hard voting.

The technique of Chopard and Spasić [12] also makes use of word embeddings and calculates the *Word Mover's Distance* [32] in

[1] https://www.semanticscholar.org/

order to select the correct expansion from the candidate expansions of an acronym. Jacobs et al. [25] propose a technique that focuses on Modern Hebrew and makes use of a Support Vector Machine (SVM) to select the correct expansion from several candidate expansions for an acronym. Similarly, to select the correct expansion for biomedical documents, in the technique of Kuo et al. [31], an SVM is implemented as well as Logistic Regression and Naïve Bayes models.

Another line of work extracts acronyms not from text but from Web Data like query click logs [26, 56].

2.2 Out-expansion

Li et al. [35] propose two techniques based on word embeddings from Word2Vec [38] to address the out-expansion problem. Their best technique, called Surrounding Based Embedding, combines the Word2Vec embeddings of the words surrounding the acronym or the expansion. Similarly to Surrounding Based Embedding, Ciosici et al. [14] propose Unsupervised Acronym Disambiguation that replaces each expansion occurrence in the collection of text documents by a normalized token and retrains the Word2Vec google news model [38] on that collection. The resulting model produces an embedding for each normalized token, i.e., an expansion embedding.

Thakker et al. [57] creates document vector embeddings, using Doc2Vec, for each document. For each set of documents D containing an expansion for an acronym A, the system trains a Doc2Vec model on D which is used to infer the embedding for an input document i containing an undefined acronym A.

Charbonnier and Wartena [11] proposed an out-expansion technique based on Word2Vec embeddings weighted by Term Frequency-Inverse Document Frequency scores to find out-expansions for acronyms in scientific document captions.

MadDog [58] proposes a sequential model to encode context in sentences followed by a feedforward network to classify the input sentence with an expansion. SciDr [53] formulates the out-expansion problem as a substring prediction task. Given a list of expansions concatenated with a sentence as input, it uses the pre-trained language model SciBERT [7] and retrains that model in the dataset sentences to predict the substring, i.e., start and end word indices corresponding to the predicted expansion. The authors also assemble additional models trained on external data. The external data considered is constituted by Wikipedia pages that contain an expansion found in the training data.

A related line of work explored the expansion of acronyms in enterprise texts [19, 36]. For instance, in Li et al. [36], enterprise textual documents as well as Wikipedia documents are used as training data. Other works explored acronym disambiguation in biomedical domains [39, 40, 43, 48, 55, 61, 62, 64]. In our work, we explore the general acronym expansion problem where the input document domain or source is not previously known.

Less directly related, but insightful, is the literature on Word Sense Disambiguation (WSD) [41, 42] because that work also must make use of the context around a token (in our case, an acronym; in the word sense literature, a word). Raganato et al. [49] proposed a benchmark for word sense disambiguation.

2.3 End-to-end Acronym Expanders

To our knowledge, systems that expand acronyms use a predefined dictionary of acronym-expansions [1, 21] as opposed to trying to discover the proper expansion based on context.

Only two end-to-end systems use context for out-expansion. First, Ciosici and Assent [13] propose an end-to-end abbreviation/acronym expansion system architecture that performs out-expansion. Unfortunately, their demo paper does not provide enough technical details and their code is proprietary.

Most recently, Veyseh et al. [58] proposes an end-to-end acronym expansion system, called MadDog, which contains a rule-based in-expander technique that improves on [51] and an out-expander based on neural networks: a sequential model to encode context followed by a feedforward network to classify the input with an expansion. They also trained their models on a large corpus of sentences.

None of the these two systems provides a framework with easy plug-in for different in and out-expansions techniques nor uses other data sources. Moreover, none of them were evaluated on an end-to-end acronym expander benchmark.

3 AcX: an End-to-end Acronym eXpander System

The AcX system (see Figure 1) consists of: (i) A *Database Creation* process which generates an *Expansion Database* that contains documents, acronyms and their corresponding in-expansions. The Expansion Database also associates each <acronym, in-expansion> pair with a *representation* of the document where that pair was found in some summary form to characterize its content[2]. (ii) The *Acronym Expander Server* that accepts one document at a time from a user and outputs a list of acronyms found in the input document and the corresponding expansions found by the system (whether as in-expansions or as out-expansions).

For each document with in-expansions, the *Database Creation* process runs the following pipeline:

(1) an *Acronym and In-Expansion Extractor* obtains the <acronym, expansion> pairs from the document using only within-document evidence.
(2) a *Representator* (there are many possible representators e.g., Latent Dirichlet Allocation that output topics) maps the document to a document representation that holds document contextual information.
(3) a fusing component stores the in-expansions, acronyms, and document representations in the *Expansion Database*, currently SQLite [23].

Given a new input document d supplied by a user, the *Acronym Expander Server* executes the following pipeline:

(1) applies the *Acronym and In-Expansion Extractor* used to build the Expansion Database to extract all the acronyms having expansions in the input document d.

[2]When benchmarking, the expansion database will provide us with both a training set and a test set.

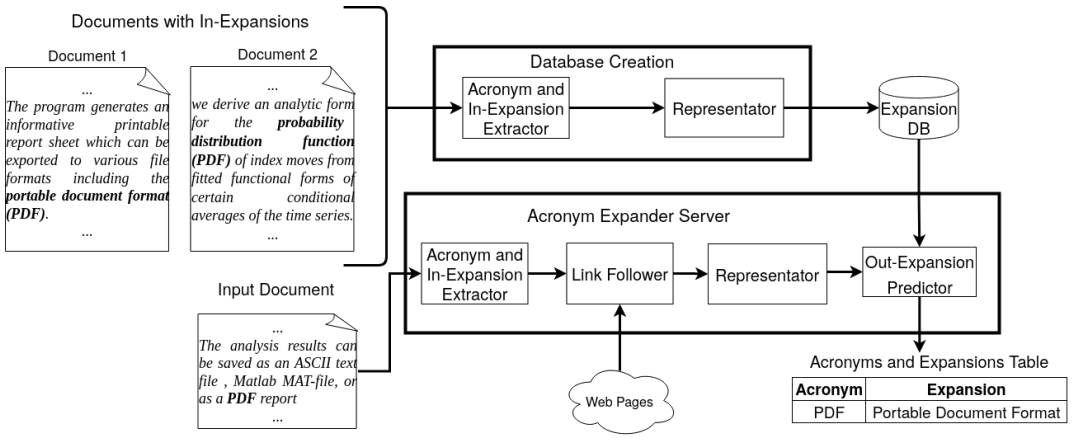

Figure 1: Acronym eXpander (AcX) system. The top stream denotes the creation of the Expansion Database that associates each <acronym, in-expansion> pair with some representation of the document(s) where that pair was found. The bottom stream shows the processing of an input document d by combining acronym in-expansion when possible and a representation of d. For an acronym A with no expansion in d, the representation of d is compared with the representations in the Expansion Database of documents containing A to find the context-appropriate expansion.

(2) when d contains links to web pages then those page texts are extracted and inspected to find the expansion for acronyms whose expansions are not found in d.
(3) utilizes the same *Representator* (say, topics from Latent Dirichlet Allocation) used to characterize each document in the Expansion Database to map d to a document representation.
(4) for each acronym A having no in-expansion in d, the server runs the *Out-Expansion Predictor* to choose a context-appropriate out-expansion. Formally, an expansion E is selected for an acronym A in d if the representations of the documents $doc(A, E)$ with expansion E share more characteristics with the representation of d by some criteria (e.g., closest cosine similarities or labeled by some machine learning classifier for A) than the documents in $doc(A, E')$ for every alternative expansion E'. Intuitively, if the context of d is publishing, then "PDF" should likely expand to "Portable Document Format" but if the context of d is probability or statistics, then "PDF" should expand to "probability distribution function."

3.1 Acronym and In-Expansion Extraction

Acronym and in-expansion extraction can use rule-based or machine learning methods. In our rule-based implementations (i.e. Schwartz and Hearst [51] and MadDog [58]), we used roughly the following three-step process as described in [44]:

(1) *Acronym extraction:* identifies acronyms in a document, e.g., PDF in Figure 1. We modified Schwartz and Hearst [51] to find candidate acronyms even when there is no expansion found in a given document. The method excludes tokens in which all alphabetic characters except the first character are lower case. We also reject acronyms of two characters where the first is a letter and the second is a dot "." to avoid person names.
(2) *Candidate expansion extraction:* builds candidate pairs of acronyms and possible in-expansions <acronym, expansion> from information in the document, e.g., <PDF, formats including the portable document format> from Document 1 in Figure 1.
(3) *Candidate refinement:* evaluates each candidate pair using a variety of heuristics (e.g. find the shortest expansion that matches the acronym) to obtain a final in-expansion for each acronym that has at least one candidate in-expansion within the document, e.g., portable document format from <PDF, formats including the portable document format>.

For the in-expanders of SciBERT and SciDr, the extraction of acronyms and expansions is formalized as a sequence labeling problem consisting of three tags identifying: (i) a word in an acronym, (ii) a word in an expansion and (iii) other word. For example, from Document 1 in Figure 1, PDF would be tagged as a word in an acronym, each word portable, document, and format would be tagged each as a word in an expansion. The remaining words in Document 1 would have the "other word" tag. AcX appropriately tags the acronym-expansion pairs and other words in the training data, then builds a machine learning model on the tagged data. The output of such machine learning models is then converted to acronym-expansion pairs by matching the acronym characters against expansions.

3.1.1 Link Follower For an input document d containing hyperlinks, the *Link Follower* component follows those links to try to find the expansions from documents pointed to by d. For each A acronym having no in-expansion, the Link Follower:

(1) searches in the *title* attribute of the hyperlink an HTML tag for the expansion. For example, given the following tag *LEED* the expansion "Leadership in Energy and Environmental Design" would be extracted for acronym LEED.
(2) executes the same in-expander technique used by the acronym and in-expansion extractor.

3.2 Representator

Representors in the AcX system summarize documents in order to capture knowledge about their semantics.

Some representators assign a topic description to a document. The topics of a document are representative terms (i.e., relevant words) in that document. If two documents have many representative terms in common, then they are considered to be semantically related.

Other representators use embeddings [34] to characterize a document. An *embedding* is a vector of real numbers in a high dimensional space. Embedding techniques map an object encoded in a one-hot representation, a very sparse and high dimensional vector of binary values, into a very dense and lower dimensional vector of real values (i.e., embedding). A small distance between embedding vectors suggests document similarity.

AcX encloses several techniques that can semantically represent an entire set of documents that contain the same expansion for a given acronym. Specifically, let $docs(A, E)$ denote the set of full document texts in which a given acronym A is defined by a single expansion E (e.g., all documents in which acronym PDF is explicitly expanded as portable document format):

Here are some representations of such a collection of documents:

- *Classic Context Vector (CCV)* [2], represents an expansion E by the set of words in $docs(A, E)$ along with their counts.
- *Document Context Vector (DCV)* (our variation of context vector), builds on context vector, however it represents each document $d \in docs(A, E)$ individually by the set of word occurrences in d. For example, the word occurrences corresponding to Document 2 in Figure 1 would contain among others the values {of: 3}, {the: 2}, {derive: 1}, {analytic: 1}, {form: 1}.
- *Term Frequency–Inverse Document Frequency (TF-IDF)* [29], weights each term t in each document $d \in docs(A, E)$ highly if it is found frequently in d and infrequently in the entire document corpus, thus permitting the characterization of each document by its highly weighted terms e.g., the TF-IDF score for the word the in Document 2 in Figure 1 is $\frac{2}{27} \cdot log(\frac{2}{2}) = 0$ because this word appears in both documents.
- *Latent Dirichlet Allocation (LDA)* [9] assigns topics to documents using a Dirichlet probabilistic model. For example, Document 2 in Figure 1 could be represented by the following topics: topic1={{analytics: 0.7}, {series: 0.3}} and topic2={{functional: 0.8}, {form: 0.2}}.
- *Doc2Vec* [34] is a document embedding technique based on Word2Vec [38] which assigns vectors to words in such a way that words that appear in the same context have a high cosine similarity. For example, the words functional and conditional would be assigned similar vectors. Thus, using the principles of Word2Vec, Doc2Vec assigns vectors to entire documents. For example, documents 1 and 2 in Figure 1 would be assigned mutually distant vectors.

3.3 Out-Expansion Predictor

To choose an out-expansion for an acronym A in an input document d having no expansion for A, the Out-Expansion Predictor component considers each candidate out-expansion E for A and compares d to some representation of $docs(A, E)$.

In the case of Classic Context Vector (CCV), we compare d with the vector representation of $docs(A, E)$. For the remaining techniques, we compare d with each document representation of $d \in docs(A, E)$.

Using cosine similarity, the Out-Expansion Predictor will choose an out-expansion E over a different expansion E' if any document $d \in docs(A, E)$ is more similar to d than all $d' \in docs(A, E')$.

The AcX system also supports classification-based approaches that work as follows. Consider all the documents, denoted $alldocs(A)$ containing in-expansions of acronym A. Some documents in $alldocs(A)$ have an in-expansion of $E1$ for A, some have $E2$ for A and so on. Given the representations of documents in $alldocs(A)$ as features and the expansions ($E1$, $E2$, etc) as labels, the out-expansion problem becomes a machine learning classification problem. When a new document d is given to AcX, its representation is passed as features to the classifier which labels d with an expansion.

The classifiers we support so far are:

- *Support Vector Machines (SVMs)* [15] fit a hyper-plane that optimally separates binary labeled data in the feature space. Non-binary classification is performed by a "one-vs-all" technique where a binary SVM classifier predicts with a certain probability if an input document belongs to a particular class (where each class corresponds to a particular expansion). The class (and therefore expansion) with the highest probability is selected. We used the LibLinear [18] implementation included in sckit-learn toolkit [46].
- *Logistic Regression (LR)* [27] fits a logistic function to classify binary classes (again a class corresponds to an expansion). Non binary classification is again performed by a "one-vs-all" technique. We used the LibLinear [18] implementation included in scikit-learn toolkit [46].
- *Random Forests (RF)* [10] fit a particular number of decision trees (default 100) trained on randomly selected samples. There will be one random forest per acronym A. The representation of a document having no expansion for A will be input to the random forest. Each tree will predict one expansion with some probability. The random forest selects the class whose average probability is the highest. We used the scikit-learn [46] implementation.

In addition to these classifiers, for evaluation purposes or for anyone who wants to try other techniques, AcX supports the following techniques from related work Surrounding Based Embedding [35], Thakker et al. [57], Unsupervised Abbreviation Disambiguation (**UAD**) [14], the SciDr out-expander (**SciDr-out**) [53], and the MadDog out-expander (**MadDog-out**) [58]. For UAD, SciDr-out and Maddog-out, AcX performs sentence segmentation and, given the results from each sentence, decides which expansion to assign to the text. For UAD, we select the most frequent predicted expansion among the sentences in the document.

We have extended SciDr-out to consider all the sentences containing the acronym A instead of just one sentence as in SciDr-out's original implementation. SciDr-out associates an acronym with its possible expansions concatenated together. The system then finds

the substring of that concatenated string with the highest probability and outputs that as the expansion. For example, the concatenated expansion of "PDF" might be "probability density function portable document format". Depending on the contents of some input document d containing "PDF", SciDr-out will choose some substring of that concatenated expansion.

MadDog-out processes the last sentence of the document containing acronym A. We have extended MadDog-out to enable training with any new set of documents, instead of just using their original models.

4 In-expansion Benchmark, Evaluation and Results

We describe our benchmark of in-expansion techniques in Section 4.1 and evaluate state-of-the-art techniques on this benchmark in Section 4.2.

4.1 A Benchmark of In-expansion Techniques

This section describes the benchmark we developed to evaluate in-expansion techniques. Section 4.1.1 details the datasets used in this benchmark. Section 4.1.2 lists the in-expansion techniques that we implemented for this benchmark. Section 4.1.3 defines the metrics that we used to evaluate the in-expansion extraction techniques.

4.1.1 Datasets The datasets included in this in-expansion benchmark are:

Medstract: This dataset is composed by 199 randomly selected MEDLINE[3] abstracts from the results of a query on the term "gene". The abstracts were manually annotated and then the annotations were reviewed by Schwartz and Hearst [51], Ao and Takagi [3], Pustejovsky et al. [48], Yarygina and Vassilieva [63] and Doğan et al. [24]. We use the last revised version of Doğan et al. [24] that contains 159 acronym-expansion pairs.

Schwartz and Hearst: This dataset has 1 000 randomly selected MEDLINE abstracts from the results of a query on the term "yeast". The abstracts were manually annotated by Schwartz and Hearst [51] and revised by Doğan et al. [24]. The revised version that we use contains 979 acronym-expansion pairs.

BIOADI: This dataset contains 1 201 abstracts from the BioCreative II gene normalization dataset. The dataset was original annotated by Kuo et al. [31] and revised by Doğan et al. [24]. It contains 1 720 acronym-expansion pairs.

Ab3P: This dataset results from the random selection of MEDLINE 1 250 abstracts. The dataset was manually annotated by Sohn et al. [54]. We use the revised version of Doğan et al. [24] that contains 1 223 acronym-expansion pairs.

SciAI: This dataset results from processing 6 786 English arXiv[4] papers. Those papers were split into sentences and sent to Amazon Mechanical Turk (MTurk) to be annotated by humans, resulting in 9 775 acronym-expansion pairs. This dataset was annotated for both acronyms and acronym-expansion pairs. The final dataset has 17 506 sentences, where 1% do not contain acronyms and 24% do not contain expansions. We use the SDU@AAAI competition [59] version[5] that was initially proposed by Veyseh et al. [60].

User-Generated: We developed this dataset that consists of 163 English Wikipedia documents randomly selected from the *Computing* category[6] in Wikipedia. It contains 1 139 acronym-expansion pairs. Although intended to evaluate an end-to-end acronym-expander system, for this in-expansion benchmark in particular, we consider only the acronym-expansion pairs with expansion in text. Later, in Section 6.1, we use the whole set of acronym-expansions pairs to evaluate end-to-end systems. Each document was annotated by two computer science students that sign for the task. Each student annotated at least two documents. During the annotation process, each student identified each acronym in the document and mapped it to an expansion. Each acronym-expansion pair was labeled by the annotators, indicating whether the expansion was present in text. Any conflict between annotators was manually resolved. The inter-annotator agreement (IAA) between each annotator (excluding the third annotator, the reviewer) using Krippendorff's alpha [30] with the MASI distance metric [45] is 0.68 for in-expansion pairs and 0.33 for out-expansion pairs. In a hypothetical scenario, if both annotators had given the same acronym-expansions, then the score would be 1. In this case, the human annotators disagree on out-expansions more often than on in-expansions. This is expected since out-expansion is a more difficult task because it implies that additional text sources have to be accessed, while for in-expansion the text provided is enough.

4.1.2 In-expansion techniques This benchmark includes the following in-expansion techniques (that are supported by our AcX system described in Section 3):

Rule-based: Schwartz and Hearst (**SH**) [51] technique and the MadDog [58] in-expansion (**MadDog-in**) technique which builds on the Schwartz and Hearst algorithm.

Machine Learning: SciBERT based technique used in [53] and the SciDR [53] in-expansion (**SciDr-in**) technique which ensembles SciBERT models with CRFs. Moreover, we also consider models used by these techniques trained *with external data* besides the individual training sets of each dataset. The external data is composed of all biomedical datasets (i.e., Medstract, Schwartz and Hearst, BIOADI, and Ab3P) if the test set is biomedical. For more general test sets, the external data consists of all datasets (i.e., biomedical datasets, SciAI, and User-Generated).

4.1.3 Performance metrics Our benchmark uses the following metrics. The metrics apply to acronyms alone as well as to acronym-expansion pairs. If the same acronym or pair appears several times in the same document, they are counted only once:

Acronym Pair Precision: the number of correctly extracted acronym pairs divided by the number of acronym pairs extracted by that technique over all documents.

[3] https://www.nlm.nih.gov/bsd/medline.html
[4] https://arxiv.org/
[5] https://github.com/amirveyseh/AAAI-21-SDU-shared-task-1-AI
[6] https://en.wikipedia.org/wiki/Category:Computing

Acronym Pair Recall: the number of correctly extracted acronym pairs divided by the number of distinct acronym pairs present over all documents.

Acronym Pair F1-measure: the harmonic mean of the *precision* and *recall* of the system.

Training time: CPU or GPU time in seconds to train the machine-learning models that are used by the in-expansion technique.

Execution time: CPU or GPU time in seconds that the in-expansion technique takes to extract acronym-expansion pairs from a document in the dataset.

4.2 In-expansion Experimental Evaluation

In this section, we evaluate the in-expansion techniques using the benchmark presented in Section 4.1.

Setup. The in-expansion experiments were performed on a machine with an Intel® Core™ i5-4690K CPU with 4 cores, and 16 GB of RAM and an NVIDIA GeForce GTX 1070. Only SciBERT and SciDr-in used the GPU.

Results. We report the Precision, Recall, and F1-measure values for the average of the biomedical datasets (i.e., Medstract, Schwartz and Hearst, BIOADI and Ab3P), SciAI and User-Generated datasets in Table 1. The additional external data used to train SciBERT and SciDr-in for the biomedical application includes the data of all biomedical datasets excluding the test set (30%). For SciAI and User-Generated data sets, the external data used to train SciBERT and SciDr-in includes all documents in the other datasets (i.e., Medstract, Schwartz and Hearst, BIOADI, Ab3p, SciAI, and User-Generated). We report the fine-grained results per biomedical dataset and execution times per dataset in the extended version of this paper [7].

For the *biomedical domain*, for all acronym and pair extraction measures (precision, recall, and F1-measure), the best acronym and in-expander technique, on average, is SH.

On *SciAI*, the machine-learning techniques SciDr-in and SciBERT outperform the rule-based techniques, SH and MadDog-in, in terms of recall (**91.78%-94.05%** for acronym and **88.24%-90.64%** for pair) and F1-measure (**94.59%-94.05%** for acronym and **90.94%-91.98%** for pair) for both acronym and pair extraction. Furthermore, MadDog-in achieves the best precisions (**98.63%** for acronym and **96.91%** for pair) followed by SciDr-in (**97.47%-97.58%** for acronym and **93.81%-94.47%** for pair).

On the *User-Generated dataset*, MadDog-in achieves the best overall precision (**92.78%** and **88.65%**) and F1-measure (**79.64%** and **76.10%**) for acronym and pair extraction, while SH surpasses MadDog-in in terms of acronym recall (**70.54%**) and matches for pair recall (**66.67%**). Furthermore, among the machine-learning techniques on the User-Generated dataset, SciDr-in surpasses SciBERT on precision (**77.08%** and **68.75%**), recall (**57.36%** and **51.16%**), and F1-measure (**57.36%** and **58.66%**) for both acronym and pair extraction. Increasing the training dataset with External Data for SciBERT yielded an increase in recalls (**58.91%** and **53.48%**) but decreased precisions (**49.67%** and **45.09%**) and F1-measures (**53.90%** and **48.93%**) for both acronym and pair extraction, while for SciDr-in, we observe a general increase in performance.

Execution time analysis. Regarding execution times, we observed from our experiments that the rule-based techniques are much faster than the machine learning techniques. SH is the fastest technique on every single dataset taking less than **0.06** seconds on average to extract acronym-expansion pairs from a document. MadDog-in is the second fastest technique taking up to **16** seconds to process each document. The machine-learning techniques SciDr-in and SciBERT take much more time to extract pairs from the datasets than the rule-based techniques. For instance, on the SciAI dataset, SciBERT takes **687** seconds. Comparing SciDr-in and SciBERT, the execution times per document of SciDr-in are much higher than SciBERT taking **11 106** seconds on SciAI dataset because it is an ensemble based technique. Regarding training times, SciBERT takes **1 700** seconds on SciAI training data, **2 691** seconds on the biomedical datasets, and **5 121** seconds to train with all datasets. SciDR-in takes longer for training: **52 257** seconds on SciAI training data, **28 612** seconds on the biomedical datasets, and **99 024** seconds to train with all datasets.

In summary:

- If document processing needs to be fast and there are hardware limitations, the rule-based techniques **SH** and **MadDog-in** are the best.
- If there are no time constraints and a large volume of data from the same domain is available, the machine learning techniques **SciBert** and **SciDr-in** are marginally better.
- If the dataset is sentence-based and a large amount of data from the same domain is available, use **SciDr-in**, though, in the medical domain, **SH** would likely be a strong contender.

5 Out-expansion Benchmark, Evaluation and Results

We describe our benchmark of out-expansion techniques in Section 5.1 and evaluate state-of-the-art techniques on this benchmark in Section 5.2.

5.1 A Benchmark of Out-expansion Techniques

This section presents our benchmark of out-expansion techniques. Section 5.1.1 describes the datasets used in this benchmark. Section 5.1.2 explains the steps used to prepare those datasets. Section 5.1.3 lists the out-expansion techniques included in the benchmark, grouped by type. Finally, Section 5.1.4 describes the metrics to evaluate those out-expansion techniques.

5.1.1 Datasets The datasets included in our out-expansion benchmark are:

MSH dataset [28] contains biomedical document abstracts from the MEDLINE (Medical Literature Analysis and Retrieval System Online) corpus used in Li et al. [35], Prokofyev et al. [47]. This dataset was automatically annotated using citations from MEDLINE and the ambiguous terms with MeSH headings identified in the Metathesaurus[8]. We use the original texts and the revised labels from Li et al. [35];

SciWISE dataset consists on the Physics dataset used in Li et al. [35] and Prokofyev et al. [47] that consists of document

[7] Temporary location: web.tecnico.ulisboa.pt/ist164790/acx_extended.pdf

[8] https://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus

Acronym and In-expansion Technique	Biomedical Datasets – Avg.						SciAI						User-Generated					
	Acronym			Pair			Acronym			Pair			Acronym			Pair		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SH	**99.31%**	**81.88%**	**89.72%**	**96.33%**	**79.52%**	**87.07%**	96.02%	82.36%	88.67%	92.85%	79.64%	85.74%	91.00%	**70.54%**	79.47%	86.00%	**66.67%**	75.10%
MadDog-in	98.45%	58.65%	73.35%	92.34%	54.97%	68.82%	**98.63%**	86.72%	92.30%	**96.91%**	85.21%	90.68%	**92.78%**	69.76%	**79.64%**	**88.65%**	**66.67%**	**76.10%**
SciBERT	85.22%	68.22%	75.71%	70.01%	56.01%	62.15%	95.69%	**94.05%**	94.86%	92.21%	**90.64%**	91.42%	65.62%	48.83%	55.99%	58.34%	43.41%	49.77%
SciBERT with External data	88.27%	75.98%	81.65%	75.97%	65.38%	70.27%	96.18%	**94.05%**	**95.11%**	92.50%	90.45%	91.46%	49.67%	58.91%	53.90%	45.09%	53.48%	48.93%
SciDr-in	90.56%	63.40%	74.53%	78.13%	54.76%	64.34%	97.47%	92.47%	94.90%	94.47%	89.63%	**91.98%**	77.08%	57.36%	65.77%	68.75%	51.16%	58.66%
SciDr-in with External data	91.91%	76.12%	83.26%	85.55%	70.86%	77.50%	97.58%	91.78%	94.59%	93.81%	88.24%	90.94%	86.36%	58.91%	70.04%	81.81%	55.81%	66.35%

Table 1: In-expansion techniques Precision, Recall, and F1-measures for acronym and pair extraction and for the average of the biomedical datasets, SciAI dataset, and User Generated dataset.

Statistics	SciWISE	MSH	CS Wiki	SciAD
# of articles	4 677	12 053	10 220	39 815
Average # of chars per article	1 193	1 552	9 246	187
# of sentences	40 300	112 456	702 387	51 340
# of distinct acronyms	129	67	630	732
# of distinct ambiguous acronyms	100	64	566	682
Average # of distinct expansions per article	1.09	0.99	1.02	1
# of distinct acronym/expansions	272	139	8 617	2173
Average # of expansions per ambiguous acronym	2.43	2.13	15.11	3.11

Table 2: Statistics of the out-expansion datasets. SciAD is the dataset with the largest number of articles, 39k. However, if we compare the number of sentences found in each dataset, CS Wiki has the largest total with 70K, 11k for MSH, 5k for SciAD and 4k for ScienceWISE. The reason is that SciAD contains the smallest articles in terms of characters (mostly just sentences having fewer than 200 characters), while others have 1k or more. CS Wiki has the biggest set of distinct acronym-expansions pairs (8k). In terms of distinct expansions per ambiguous acronym, CS Wiki has around 15, while the remaining dataset have maximum around 3.

abstracts. This dataset was annotated by human experts, and it includes expansions either containing at least 2 words or a single word with at least 14 characters.

CS Wiki (Computer Science Wikipedia) dataset created in Thakker et al. [57] contains documents from different fields that contain acronyms used in computer science. Expansions were extracted by parsing the content of English Wikipedia disambiguation pages of acronyms used in computer science (e.g., https://en.wikipedia.org/wiki/PDF_(disambiguation)).

SciAD This dataset was prepared for the out-expansion SDU@AAAI-21 competition [59]. It is based on the SciAI in-expansion dataset, described in Section 4.1.1. We use the revised version[9] created by Egan and Bohannon [17] who removed duplicate sentences from the original train and validation sets.

Table 2 presents relevant statistics about each dataset. An ambiguous acronym is an acronym that has more than one expansion available in the dataset.

[9]https://github.com/PrimerAI/sdu-data

5.1.2 Data Preparation The data preparation steps are roughly the same for each out-expansion technique:

(1) **Dataset Splitting:** We split each dataset into *train* and *test* sets (respectively 70% and 30% of the documents of the original dataset). We then apply 5-fold cross validation on the train dataset in order to tune the hyperparameters of each out-expansion technique. The hyperparameter-tuned technique is then tested on the yet unseen 30% of the data.
(2) **Expansion Consolidation:** For the expansions of acronym A in each dataset, we apply an approximate duplicate detection process that groups expansion strings that correspond to the same expansion meaning. For example, *portable document format* and *Portable-Document-Formats* are two distinct strings that refer to the same real expansion. As criteria, we consider that two expansions are equal if their lower case versions without dashes have an edit-distance less than 3 or if the first 4 characters of each word are equal. Then, expansions are consolidated by replacing in text all expansion strings with the same meaning by the most frequent expansion.
(3) **Expansion Removal:** When testing the accuracy of out-expansion techniques on some document d, we associate any acronym A in the document with its in-expansion $In(A)$, if present. Then, we replace all occurrences of the in-expansion $In(A)$ in text by A itself.
(4) **Tokenization:** We apply the word tokenization from the Natural Language Toolkit (NLTK) [8] to obtain only alphanumeric tokens. Additionally, we remove stop words using NLTK and numeric tokens;
(5) **Token Normalization:** We transform each token into its stem, e.g. probable, probability, and probabilities all map to probabl. We use the Porter Stemmer algorithm from NLTK.

The preparation of the MSH and SciWISE datasets follows the preprocessing reported in Li et al. [35], so we apply all the preparation steps above except token normalization. The five steps are consistent with the pre-processing steps used in Thakker et al. [57] for the CS Wiki dataset. For SciDr-out and Maddog-out, we apply only the first three steps, because these techniques replace the last two steps with steps that are compatible with the language models of the neural networks they use.

5.1.3 Out-expansion Techniques This benchmark includes the following groups of out-expansion techniques:

Classical Techniques: We use two baselines: **Random** which randomly assigns a possible expansion to an acronym; and **Most Frequent** which always selects the most frequent expansion found in our training data in terms of occurrences in distinct documents. We use the Cosine similarity (**Cossim**) with the Classic Context Vector (**CCV**) [35], Document Context Vector (**DCV**) - variant of Classic for each document, Surrounding Based Embedding (**SBE**) [35], and **Thakker** et al. [57].

Sentence-oriented Techniques: We include related work techniques that expect a sentence as input (instead of a document) and adapt them as described in the AcX overview (Section 3.3) for: Unsupervised Abbreviation Disambiguation (**UAD**) [14], MadDog [58] out-expander (**MadDog-out**), and SciDr [53] out-expander (**SciDr-out**). We also use **SciDr-out with External Data** consisting of the Wikipedia pages that contain an expansion found in the training data.

Representator Techniques: We include **Cossim** with the document representation techniques described in Section 3.2, that we have adapted from natural language processing: Term Frequency-Inverse Document Frequency (**TF-IDF**), Latent Dirchlet Allocation (**LDA**), and **Doc2Vec**.

Classification Techniques: We created a complete new class of out-expansion techniques that use the outputs of a representator as features for a Machine Learning classifier, specifically, Random Forests (**RF**), Logistic Regression (**LR**), and Support Vector Machines (**SVM**). Each acronym has its own classifier trained with the features of the documents that contain an expansion for the acronym (e.g., acronym PDF will have a random forest RandFor(PDF) based on documents that contain an in-expansion for PDF). Based on the features of a target document d, the classifier will choose the appropriate expansion as explained in Section 3.3.

Combination of Representator Techniques: The final type of out-expansion techniques that we assembled consists of combining two representators' outputs, namely the Doc2Vec with a Context Vector (either Classic or Document), as input to predictors: **CCV + Doc2Vec** and **DCV + Doc2Vec**. Combinations are constructed by concatenating the outputs together into a single feature vector.

5.1.4 Performance Metrics Our benchmark uses the following metrics:

Out-expansion accuracy: The accuracy of predicting the right expansion for a given acronym in a textual document. Intuitively, this is the fraction of acronym-expansions that are correctly predicted. Accuracy is also used in previous out-expansion works [14, 35, 57] and analogous benchmarks, e.g., for Word-Sense-Disambiguation [49]. Note that an acronym may appear many times in the same document and many times across documents. In our measure, if A is in k documents, it is counted k times, but if A is present j times in the same document, it is counted only once in that document.

Out-Expansion macro averages: Recently, Veyseh et al. [58][60] started using a different set of metrics that we have implemented and measured for completeness. Those metrics are macro-averages of Precision, Recall and F1-measures for acronym-expansions pairs. So, we calculate precision, recall, and F1-measure independently for each acronym-expansion in the training data.

Representator execution time: the execution time to create representations of training documents.

Average execution time per document: the average execution time to predict expansions for acronyms in a document.

5.2 Out-expansion Experimental Results

Setup. For out-expansion on the benchmark presented in Section 5.1, we ran the experiments on a GoogleCloud platform[10] machine with the following specifications: Intel Broadwell CPU platform with 8 cores, 30GB to 80GB of RAM (Random Access Memory). For MadDog-out and SciDr-out, half of a Tesla K80 GPU board was used. The code ran in Python 3.7.

To reduce the duration of experiments, we first find the representator's hyperparameters with the cosine similarity because it involves no learning nor hyperparameters of its own, then save the best representator model on disk. Given the best representator hyperparmeters, we find the best out-expansion predictor model hyperparameters.

Results. For each dataset, we report the out-expansion accuracy and macro F1-measure to predict the expansions of acronyms in a document. In Table 3, we present a summary of the best techniques for each dataset. The *Technique Group* column identifies the out-expansion group that the technique belongs, as organized in Section 5.1.3 (e.g., Classical). The *Predictors* column identifies the out-expansion predictor technique (e.g., Cossim or an ML classifier) that takes a given document representation to predict an expansion (e.g., Cossim). The *Representators* column indicates the technique used to generate a document representation (e.g., Doc2Vec). We did not run SciDr-out with External Data on CSWiki dataset because the external data (i.e., Wikipedia data) would overlap with this dataset's documents.

In these out-expansion experiments, we measure the accuracy and macro F1 only on the acronym-expansions pairs whose acronym is ambiguous (i.e., have at least two expansions in the training data) and whose in-expansions are in the training data.

The best overall techniques (average above 89% of accuracy) in descending order are: SciDr-out, Cossim with CCV, Cossim with TF-IDF, Cossim with DCV, Cossim with Doc2Vec alone or with DCV, and SVM with Doc2Vec. Regarding statistical significance, SciDr-out is the best for CSWiki and SciAD. CCV achieves higher accuracy for SciWISE. However, it is not statistically better than: SciDr with External Data, Cossim TF-IDF (which scores higher macro F1 in this dataset), Cossim with Doc2Vec, and Cossim with combinations of Doc2Vec with either CCV or DCV. Finally, for MSH, SVM with Doc2vec scores higher accuracy but not statistically significantly better than: LR with Doc2Vec, SVM with Doc2Vec combined with either CCV or DCV.

Document processing execution times. Among these best techniques, Cossim with CCV is the fastest for all datasets, able to process input documents in less than **0.07** seconds on dataset average. However, SVM with Doc2Vec is the fastest for MSH and SciWISE. The slowest among the best is Cossim with TF-IDF (average **2.5s**),

[10] https://cloud.google.com/

Out-expansion Technique			ScienceWISE		MSH		CSWiki		SciDR		Average		
Technique Group	**Predictors**	**Repre-sentators**	**Acc**	**MaF1**	**Acc**	**MaF1**	**Acc**	**MaF1**	**Acc**	**MaF1**	**Acc**	**MaF1**	**Exec Times**
Classical	Random		47.72%	46.10%	47.04%	45.49%	14.54%	14.21%	33.06%	32.22%	35.59%	34.51%	**0.00**
	Most Frequent		70.52%	49.31%	50.30%	32.32%	47.76%	20.37%	69.08%	37.64%	59.41%	34.91%	**0.00**
	Cossim	CCV	**91.34%**	80.72%	97.65%	97.67%	77.96%	65.00%	92.15%	88.86%	89.77%	83.06%	0.07
		DCV	89.51%	78.69%	96.18%	96.07%	78.59%	65.86%	93.67%	87.08%	89.49%	81.92%	0.15
		SBE	88.07%	75.89%	95.84%	95.24%	74.60%	63.30%	86.50%	80.76%	86.25%	78.80%	0.05
	Thakker		87.77%	77.38%	92.53%	91.68%	73.16%	63.86%	84.36%	73.21%	84.46%	76.53%	3.79
Sentence--Oriented	UAD		43.69%	46.73%	93.92%	92.55%	12.94%	11.60%	34.98%	45.75%	46.38%	49.16%	0.01
	MadDog-out		89.13%	68.84%	94.09%	93.16%	57.03%	47.71%	87.38%	73.23%	81.91%	70.73%	0.37
	SciDr-out		88.22%	77.45%	97.23%	96.76%	**84.19%**	**72.67%**	**95.07%**	**89.01%**	**91.18%**	**83.97%**	1.28
	SciDr-out with External Data		**89.89%**	77.86%	97.58%	97.22%	N/A	N/A	**95.30%**	**89.87%**	N/A	N/A	N/A
Repre-sentator	Cossim	TF-IDF	**91.26%**	**81.82%**	97.62%	97.57%	77.80%	65.36%	91.79%	83.48%	89.62%	82.06%	2.53
		LDA	85.56%	73.94%	93.81%	93.28%	71.89%	60.49%	84.56%	73.39%	83.95%	75.28%	0.02
		Doc2Vec	**90.27%**	79.04%	98.33%	98.07%	77.16%	65.22%	92.02%	82.94%	89.45%	81.32%	0.10
Classi-fication	RF	TFIDF	70.82%	52.03%	84.53%	76.78%	32.11%	23.14%	87.64%	68.90%	68.77%	55.21%	23.79
		LDA	70.75%	54.13%	95.64%	92.84%	67.57%	50.78%	82.32%	61.33%	79.07%	64.77%	1.20
		Doc2Vec	79.18%	61.34%	96.58%	95.37%	66.29%	41.55%	84.39%	62.43%	81.61%	65.17%	1.36
	LR	TFIDF	71.05%	54.47%	93.41%	88.29%	71.89%	45.59%	80.63%	55.06%	79.24%	60.85%	11.84
		LDA	71.13%	51.49%	88.66%	80.02%	71.73%	48.77%	80.08%	55.16%	77.90%	58.86%	0.02
		Doc2Vec	88.83%	78.35%	**98.87%**	98.72%	76.68%	57.97%	90.75%	77.95%	88.78%	78.25%	0.11
	SVM	TFIDF	81.84%	62.13%	94.71%	91.27%	77.16%	53.54%	91.01%	78.24%	86.18%	71.29%	3.23
		LDA	78.88%	59.80%	93.64%	91.16%	71.89%	51.11%	85.59%	70.63%	82.50%	68.18%	0.02
		Doc2Vec	89.59%	79.31%	**98.93%**	98.79%	77.00%	58.70%	91.47%	80.81%	89.24%	79.40%	0.10
Combi-nation of Repre-sentators	Cossim	CCV + Doc2Vec	**90.27%**	79.04%	98.19%	97.95%	77.16%	65.25%	86.92%	82.82%	88.14%	81.27%	0.33
		DCV + Doc2Vec	**90.27%**	79.01%	98.33%	98.10%	77.16%	65.19%	92.05%	82.96%	89.45%	81.32%	1.65
	SVM	CCV + Doc2Vec	89.97%	80.44%	**98.95%**	**98.84%**	77.00%	58.70%	80.73%	76.06%	86.66%	78.51%	0.39
		DCV + Doc2Vec	89.67%	79.35%	**98.95%**	98.83%	77.00%	58.70%	90.20%	75.90%	88.95%	78.20%	1.68

Table 3: Out-expansion accuracy (Acc) and macro F1-measure (MaF1). Values marked as bold indicate the best Acc obtained in that dataset. A method M1 is considered better than M2 if a non-parametric significance test (based on shuffling[22]) indicates that the difference in their means has a p-value < 0.05. Thus, even though each column has a highest mean value for some method H, the value of a method M will be bolded if H is no better than M based on the p-value criterion.

Representators	SciWISE	MSH	CSWiki	SciAD	Average
CCV	**0**	**1**	**5**	**1**	**2**
DCV	**0**	**1**	**5**	**1**	**2**
SBE	6	23	115	9	38
UAD	43	93	331	54	130
TF-IDF	2	10	58	1	18
LDA	155	7 766	8 830	4 247	5 250
Doc2Vec	13	32	212	108	91
SciDr-in	11 227	42 716	147 454	50 282	62 920
SciDr-in External Data	14 299	46 651	N/A	66441	42 464
MadDog-in	566	728	10 008	1 198	3 125

Table 4: Representator execution times in seconds for each dataset.

followed by SciDr-out (**1.3s** for base and **2.3s** with external data). These differences are statistically significant [11].

[11] We report the fine-grained execution times values per dataset in the extended version of this paper temporarily located at: web.tecnico.ulisboa.pt/ist164790/acx_extended.pdf

Analysis of Classical techniques. The baselines Random and Most frequent out-expanders have better accuracies on SciWISE (**48%** and **71%**) followed by SciAD (**33%** and **69%**) and MSH (**47%** and **50%**). The worst scores are obtained on CSWiki (**15%** and **48%**), because that dataset has far more ambiguity.

MadDog-out achieves near the best performance on SciWISE and MSH, is also good on SciAD, and is worst on CSWiki. MadDog-out's input is a sentence at a time like SBE and UAD, so context is also limited to a few words.

Analysis of remaining Representators and Classification techniques. LDA works well in many Natural Language Processing tasks, but less well for our out-expansion task, probably because measuring document similarity by comparing topics is not the best use of LDA.

Techniques with TF-IDF as a representator are surprisingly competitive in SciWISE and CSWiki, but they achieve inferior accuracy for the other two datasets. Pre-processing steps (Section 5.1.2) play an important role in CCV performance, for instance, in the Tokenization step, we remove stop-words which TF-IDF takes into

account thus automatically giving them less relevance due to the IDF score.

Predictors using RF are slower and slightly less accurate than Cossim and SVMs. RFs struggle with the fact that, in this acronym expansion, there are many features but only a small number of samples, e.g., 300 dimensions from Doc2Vec and a few documents per acronym. By contrast, Cossim and SVM are usually the best predictors closely followed by LR. The representator hyperparameter search was performed for TF-IDF, LDA, and Doc2Vec using Cossim as a predictor, hence they are fitted for this predictor. SVMs and LRs are similar classifiers, the first fits a separating hyper-plane, the other a logistic function.

Execution times of representators. The training execution times depend only on the representators and are reported in Table 4. The Doc2Vec models used by Thakker et al. are created during the input document processing hence are not reported in Table 4.

The CCV and DCV representators take the least time (average **2s**) closely followed by TF-IDF (average **18s**) (Table 4). The most expensive models are SciDr-out (**14ks-66ks**) followed by Maddog-out (**566s-10ks**) which use either language models or neural networks.

External data analysis. SciDr-out with External Data improves over the SciDr-out base (Table 3, indicating that adding additional models trained on external data usually helps at roughly double the time cost.

In summary:

- If training and test time is limited, then use **CCV**, which requires almost no training time (less than 5s) and is the fastest in testing time among the best set of techniques.
- If neither training time nor document processing time is of major concern and especially if GPU processing is available, then use **SciDr-out**.
- A pipeline balancing time and accuracy is to use **Doc2vec** as feature inputs for either **cossim** or **SVMs**.

6 End-to-end Benchmark and Evaluation

The end-to-end benchmark described in Section 6.1 uses input documents containing acronyms, where some of those acronyms contain expansions and others do not. The evaluation in Section 6.2 will measure the recall and precision of acronym expansion for both people and AcX pipelines.

6.1 A Benchmark of End-to-End Acronym Expansion

Section 6.1.1 describes the train and test datasets used in this benchmark. Section 6.1.2 lists the end-to-end acronym expander systems included in the benchmark. Finally, Section 6.1.3 presents the metrics that our benchmark uses to evaluate the systems.

6.1.1 Datasets The end-to-end benchmark uses two different datasets: (i) for testing, the user-generated dataset of Section 4.1.1. (ii) The *train* dataset consists of documents from Wikipedia that do not belong to the annotated test set. Those documents came from the Wikipedia dump of March 1, 2020[12] and were processed with WikiExtractor [4].

We preprocessed all the documents using all the steps described in Section 5.1.2 for all out-expansion techniques except Maddog-out which uses its own preprocessing techniques.

6.1.2 End-to-end systems We use: (i) the end-to-end MadDog System (**MadDog-sys**) and (ii) various ***pipelines* of AcX** consisting of an in-expansion technique along with possibly the Link Follower (**LF**) technique followed by an out-expansion technique possibly with machine learning (see Figure 1). An example of a pipeline would be the SH in-expander, followed by the LF component, Doc2Vec, and SVMs, The pipelines we test consist of combinations of the most practical (accurate and fastest) techniques for in-expansion and out-expansion as determined by the benchmarks in Sections 4.2 and 5.2. Specifically, AcX pipelines use either the **MadDog-in** or the **SH** technique as in-expanders to identify acronyms and expansions in input documents. For out-expansion, AcX pipelines include one of the following combinations of out-expansion techniques, i.e., a predictor (Section 3.3) with a representator (Section 3.2): (i) **Cossim** with **CCV**; (ii) **Cossim** with **Doc2vec**; and (iii) **SVM** with **Doc2vec**.

Finally, we compare MadDog-sys and the various pipelines of AcX against the **student annotators** before the reviewer (the third annotator) resolved conflicts to decide on the final annotations in order to measure human performance.

6.1.3 Performance Metrics Similarly to Section 4.1.3, we evaluate MadDog-sys, different pipelines of AcX, and human annotators listed in Section 6.1.2 in terms of Precision (**P**), Recall (**R**) and F1-Measure (**F1**). *Precision* is the number of correct system-found acronym-expansion pairs divided by the total number of system-identified pairs. *Recall* is the number of correct system-found acronym-expansion pairs divided by the total number of human-found pairs. *F1-measure* is the harmonic mean of Precision and Recall. We also measure training and per test document execution times.

6.2 Results on End-to-end Experiments

Setup. For these end-to-end system experiments, we used a virtual machine with the following specifications: AMD EPYC Processor with 16 cores and 256GB of RAM (Random Access Memory). The code ran in Python 3.7.

Results. Table 5 presents the results for the AcX system running each one of the different pipelines mentioned in Section 6.1.2, the MadDog-sys[13], and the results for the student annotators. The AcX pipeline composed by MadDog-in, SVM with Doc2Vec, and the LF component obtains the best results for precision (59.38%) and F1-measure (53.37%). However, based on the F1-measure, this is not statistically significantly better (i.e., P-value above 0.05) than the following system pipelines: (i) with the same out-expander but without LF, or (ii) with SH and Cossim with CCV with or without LF. The best system pipeline (including the FL component) takes **7s** on average to process a document. Our best AcX pipeline obtains better results for all measures than the MadDog-sys (**+20%** of F1) and is faster (**7s** to **1084s**).

[12] https://dumps.wikimedia.org/enwiki

[13] https://archive.org/details/MadDog-models

AcX (pipelines)							
In-exp	**Out-exp Predictor**	**Repre-sentators**	**LF**	**P**	**R**	**F1**	**Exec Time**
SH	Cossim	CCV	No	**51.43%**	45.62%	**48.35%**	21
			Yes	**52.16%**	46.30%	**49.05%**	19
	SVM	Doc2Vec	Yes	**56.05%**	**49.75%**	**52.71%**	2
Mad-Dog-in	Cossim	DCV	No	53.12%	43.15%	47.62%	18
			Yes	54.46%	44.44%	48.95%	19
		Doc2Vec	Yes	56.20%	45.86%	50.51%	7
	SVM	Doc2Vec	No	**59.12%**	48.03%	**53.00%**	**1**
			Yes	**59.38%**	48.46%	**53.37%**	7
MadDog-sys				37.85%	29.14%	32.93%	1084
Student annotators				88.36%	76.41%	81.95%	N/A

Table 5: End-to-end system quality metrics and average execution times to process a document in seconds. Values marked as bold indicate the best obtained in that metric. A method M1 is considered better than M2 if a non-parametric significance test (based on shuffling[22]) indicates that the difference in their means has a p-value < 0.05. Thus, even though each column has a highest mean value for some method *H*, the value of a method *M* will be bolded if *H* is no better than *M* based on the p-value criterion.

Best AcX pipeline analysis. The reason why the precision is low is that our best AcX pipeline considers certain strings to be acronyms even though they are not (245 in total). Some are small words like "and" and "not". Others are codes like ZAB and ZAU that refer to airports. Conversely, the acronym and in-expansion extraction component fails to identify lower case acronyms as acronyms, common measurement units (e.g., m for meter, g for gram, kbit for kilobit) and some common language abbreviations (e.g., Micro, "etc", email) which usually everyone knows. By contrast, AcX provides the correct expansion for the acronyms that newcomers to a field may not know, e.g., CAS - Computer Algebra System; SLS - SoftLanding Linux system; and ILM - Industrial Light & Magic.

In and out expansion analysis. Apart from the metrics in Table 5, we measured independently the performance of in- and out-expansion. When evaluating just the acronym and in-expander extraction component of AcX pipelines, using SH scored an in-expansion F1-measure of **68.88%** and using MadDog-in scored **69.92%**. If we evaluate out-expansion (acronyms left to expand after in-expansion and LF component), our best AcX pipeline (SVM as predictor with Doc2Vec as representator) obtains an F1-measure of **45.42%**.

LF analysis. Overall, the LF component improves the quality metrics. We measured Precision for only the LF predictions: **93.26%** when running with SH and **92.39%** when running with MadDog-in. When combined with the best out-expander techniques (i.e., SVM as predictor with Doc2Vec as representator), following links improves the F1-score by only **+0.37%** (Table 5). Thus, the better the out-expander, the less link following helps.

Execution times of representators. In terms of execution times to create the out-expansion representators for each system pipeline, CCV executes in less time (**389s** with SH and **332s** with MadDog-in) than Doc2Vec (**15 985s** with SH and **13 341s** with MadDog-in).

Comparison with human performance. Compared with human annotators (before review by one of the authors), our best AcX pipeline (MadDog-in, the LF component, and SVMs with Doc2Vec) is around 29% lower in Precision, 28% lower in Recall, and 29% lower in F1-Measure. So, there is a lot of room for improvement. On the other hand, automatic Acronym Expansion is rapid (**7s** per document) and can give at least a good first guess.

An example application of AcX. Consider one of the documents out of the 163 at random whose original page is here https://en.wikipedia.org/wiki/CC_(complexity). Our best AcX pipeline identified the following acronym-expansion pairs: CC - comparator circuits; CCVP - comparator circuit value problem; AC - accumulator; NC - nick's class; and NL - nondeterministic logarithmic. However, it failed to identify CC-complete, and P. We can see that CC, CCVP, NC, and NL are correct and AC is incorrect.

In summary:

- The best AcX pipeline consists of **MadDog-in**, with **SVM** and **Doc2Vec**.
- The **LF** component has high precision but only slightly improves the AcX performance.

7 Conclusions and Future Work

The AcX system synthesizes and extends the best of previous work on acronym expansion. In the process, our major technical findings are:

- In-expansion rule-based techniques (SH and MadDog-in) usually work best and require little execution time.
- For out-expansion, SciDr-out usually works best but may take an impractically long time to train, followed by Cossim and SVMs with either CCV or Doc2Vec.
- There is still a significant gap between the best AcX pipelines and human-level performance.

There are five data and software products of our work that future researchers can either extend or use as a basis of comparison.

(1) The first human-annotated dataset for end-to-end acronym expander systems.
(2) Three benchmarks to evaluate: (i) in-expansion techniques, (ii) out-expansion techniques, (iii) the combination in an end-to-end setting.
(3) The end-to-end AcX system is available publicly and can be applied to arbitrary languages, follows hyperlinks, and can incorporate new in- and out-expansion techniques.

Future Work

Because the automated methods in the state-of-the-art fall well below human-level accuracy levels, we see the need for improvements in both in-expansion (especially acronym identification) and out-expansion. Some avenues for improvements include: (i) more accurate in-expansion (e.g., additional acronym-expansion extraction patterns) and (ii) new context representation techniques.

With respect to the AcX system, we will add an Application Programming Interface (API) so text analytics systems (e.g., sentiment analysis) can benefit from acronym expansion. Finally, because our platform easily extends to other languages (e.g., our Portuguese extension was done by a high school student), we also plan to create AcX pipelines for a variety of natural languages.

References

[1] ABBREX. 2011. ABBREX - The Abbreviation Expander. http://abbrex.com/

[2] Khaled Abdalgader and Andrew Skabar. 2012. Unsupervised Similarity-based Word Sense Disambiguation Using Context Vectors and Sentential Word Importance. *ACM Transactions on Speech and Language Processing* 9, 1 (2012), 2–21.

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

[4] Giusepppe Attardi. 2015. WikiExtractor. https://github.com/attardi/wikiextractor.

[5] S Azimi, H Veisi, and R Amouie. 2019. A method for automatic detection of acronyms in texts and building a dataset for acronym disambiguation. In *Iranian Conference on Signal Processing and Intelligent Systems*. 1–4.

[6] Adrian Barnett and Zoe Doubleday. 2020. Meta-Research: The growth of acronyms in the scientific literature. *eLife* 9 (jul 2020), e60080. https://doi.org/10.7554/eLife.60080

[7] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: Pretrained Language Model for Scientific Text. In *Empirical Methods in Natural Language Processing*.

[8] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python* (1st ed.). O'Reilly Media, Inc.

[9] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3 (March 2003), 993–1022.

[10] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32.

[11] Jean Charbonnier and Christian Wartena. 2018. Using Word Embeddings for Unsupervised Acronym Disambiguation. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2610–2619. https://www.aclweb.org/anthology/C18-1221

[12] Daphné Chopard and Irena Spasić. 2019. A Deep Learning Approach to Self-expansion of Abbreviations Based on Morphology and Context Distance. In *Statistical Language and Speech Processing*. 71–82. https://doi.org/10.1007/978-3-030-31372-2_6

[13] Manuel R. Ciosici and Ira Assent. 2018. Abbreviation Expander - a Web-based System for Easy Reading of Technical Documents. In *Conference on Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 1–4. https://www.aclweb.org/anthology/C18-2001

[14] Manuel R. Ciosici, Tobias Sommer, and Ira Assent. 2019. Unsupervised Abbreviation Disambiguation Contextual disambiguation using word embeddings. *Computing Research Repository* arXiv:1904.00929 (2019). arXiv:1904.00929 http://arxiv.org/abs/1904.00929 version 2.

[15] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Computing Research Repository* arXiv:1810.04805 (2018). https://arxiv.org/abs/1810.04805

[17] Nicholas Egan and John Bohannon. 2021. Primer AI's Systems for Acronym Identification and Disambiguation. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence*. CEUR-WS.org. http://ceur-ws.org/Vol-2831/paper30.pdf

[18] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9 (jun 2008), 1871–1874.

[19] Shicong Feng, Yuhong Xiong, Conglei Yao, Liwei Zheng, and Wei Liu. 2009. Acronym Extraction and Disambiguation in Large-Scale Organizational Web Pages. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (Hong Kong, China). Association for Computing Machinery, New York, NY, USA, 1693–1696. https://doi.org/10.1145/1645953.1646206

[20] Michael R. Glass, Md. Faisal Mahbub Chowdhury, and Alfio Massimiliano Gliozzo. 2017. Language Independent Acquisition of Abbreviations. *Computing Research Repository* arXiv:1709.08074 (2017). arXiv:1709.08074 http://arxiv.org/abs/1709.08074 version 1.

[21] Phil Gooch. 2012. BADREX: In situ expansion and coreference of biomedical abbreviations using dynamic regular expressions. *Computing Research Repository* arXiv:1206.4522 (2012). arXiv:1206.4522 http://arxiv.org/abs/1206.4522 version 1.

[22] Phillip I Good. 2006. *Resampling Methods: A Practical Guide to Data Analysis*. Birkhäuser Basel. https://doi.org/10.1007/0-8176-4444-X

[23] Richard D Hipp. 2020. SQLite. https://www.sqlite.org/

[24] Rezarta Islamaj Doğan, Donald C Comeau, Lana Yeganova, and W John Wilbur. 2014. Finding abbreviations in biomedical literature: three BioC-compatible modules and four BioC-formatted corpora. *Database: the journal of biological databases and curation* 2014 (2014).

[25] Kayla Jacobs, Alon Itai, and Shuly Wintner. 2020. Acronyms: identification, expansion and disambiguation. *Annals of Mathematics and Artificial Intelligence* 88, 5 (2020), 517–532.

[26] A Jain, S Cucerzan, and Saliha Azzam. 2007. Acronym-Expansion Recognition and Ranking on the Web. *IEEE International Conference on Information Reuse and Integration* (2007), 209–214.

[27] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

[28] Antonio J Jimeno-Yepes, Bridget T McInnes, and Alan R Aronson. 2011. Exploiting MeSH indexing in MEDLINE to generate a data set for word sense disambiguation. *BMC Bioinformatics* 12, 1 (2011), 1–14.

[29] Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 28 (1972), 11–21.

[30] Klaus Krippendorff. 2018. *Content analysis: An introduction to its methodology*. Sage publications.

[31] Cheng-Ju Kuo, Maurice HT Ling, Woody Lin, and Chun-Nan Hsu. 2009. BIOADI: A machine learning approach to identifying abbreviations and definitions in biological literature. *BMC bioinformatics* 10 Suppl 15 (12 2009), S7.

[32] Matt J Kusner, Yu Sun, Nicholas I Kolkin, and Kilian Q Weinberger. 2015. From Word Embeddings to Document Distances. In *International Conference on International Conference on Machine Learning*. JMLR.org, 957–966.

[33] John D Lafferty, Andrew McCallum, and Fernando C N Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 282–289. http://dl.acm.org/citation.cfm?id=645530.655813

[34] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning* (Beijing, China), Vol. 32. 1188–1196.

[35] Chao Li, Lei Ji, and Jun Yan. 2015. Acronym Disambiguation Using Word Embedding. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (Austin, Texas). 4178–4179.

[36] 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*, Vol. 1: Long Papers. Association for Computational Linguistics, Melbourne, Australia, 1308–1317. https://doi.org/10.18653/v1/P18-1121

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

[38] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Computing Research Repository* arXiv:1301.3781 (2013). https://arxiv.org/abs/1301.3781 version 3.

[39] Sungrim Moon, Bridget McInnes, and Genevieve B Melton. 2015. Challenges and practical approaches with word sense disambiguation of acronyms and abbreviations in the clinical domain. *Healthcare Informatics Research* 21, 1 (jan 2015), 35–42. https://doi.org/10.4258/hir.2015.21.1.35

[40] Sungrim Moon, Serguei Pakhomov, and Genevieve B Melton. 2012. Automated disambiguation of acronyms and abbreviations in clinical texts: window and training size considerations. *AMIA Annual Symposium proceedings* 2012 (2012), 1310–1319.

[41] Andrea Moro and Roberto Navigli. 2015. SemEval-2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Denver, Colorado, 288–297. https://doi.org/10.18653/v1/S15-2049

[42] Roberto Navigli. 2009. Word Sense Disambiguation: A Survey. *Comput. Surveys* 41, 2, Article 10 (Feb. 2009), 69 pages. https://doi.org/10.1145/1459352.1459355

[43] Sergeui Pakhomov, Ted Pedersen, and Christopher G Chute. 2005. Abbreviation and acronym disambiguation in clinical discourse. *AMIA Annual Symposium proceedings* 2005 (2005), 589–593.

[44] 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*. https://www.aclweb.org/anthology/W01-0516

[45] Rebecca Passonneau. 2006. Measuring Agreement on Set-valued Items (MASI) for Semantic and Pragmatic Annotation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA), Genoa, Italy.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[47] Roman Prokofyev, Gianluca Demartini, Alexey Boyarsky, Oleg Ruchayskiy, and Philippe Cudré-Mauroux. 2013. Ontology-Based Word Sense Disambiguation for Scientific Literature. In *Proceedings of the 35th European Conference on Advances in Information Retrieval* (Moscow, Russia). Springer-Verlag, Berlin, Heidelberg, 594–605. https://doi.org/10.1007/978-3-642-36973-5_50

[48] J Pustejovsky, J Castaño, B Cochran, M Kotecki, and M Morrell. 2001. Automatic extraction of acronym-meaning pairs from MEDLINE databases. *Studies in Health Technology and Informatics* 84, Pt 1 (2001), 371–375.

[49] Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, Vol. 1, Long Papers. Association for Computational Linguistics, Valencia, Spain, 99–110. https://www.aclweb.org/anthology/E17-1010

[50] Saneesh Mohammed N and K A Abdul Nazeer. 2013. An improved method for extracting acronym-definition pairs from biomedical Literature. In *2013 International Conference on Control Communication and Computing (ICCC)*. 194–197.

[51] Ariel S Schwartz and Marti A Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing*. 451–462.

[52] Utpal Kumar Sikdar and Björn Gambäck. 2017. A Feature-based Ensemble Approach to Recognition of Emerging and Rare Named Entities. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Association for Computational Linguistics, Copenhagen, Denmark, 177–181. https://doi.org/10.18653/v1/W17-4424

[53] Aadarsh Singh and Priyanshu Kumar. 2021. SciDr at SDU-2020: IDEAS–Identifying and Disambiguating Everyday Acronyms for Scientific Domain. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence*. CEUR-WS.org. http://ceur-ws.org/Vol-2831/paper31.pdf

[54] Sunghwan Sohn, Donald C Comeau, Won Kim, and W John Wilbur. 2008. Abbreviation definition identification based on automatic precision estimates. *BMC bioinformatics* 9, 1 (2008), 402.

[55] Mark Stevenson, Yikun Guo, Abdulaziz Al Amri, and Robert Gaizauskas. 2009. Disambiguation of Biomedical Abbreviations. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*. Association for Computational Linguistics, USA, 71–79.

[56] 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*. Association for Computing Machinery, New York, NY, USA, 1261–1272. https://doi.org/10.1145/2488388.2488498

[57] Aditya Thakker, Suhail Barot, and Sudhir Bagul. 2017. Acronym Disambiguation: A Domain Independent Approach. *Computing Research Repository* arXiv:1711.09271 (2017). https://arxiv.org/abs/1711.09271 version 3.

[58] Amir Pouran Ben Veyseh, Franck Dernoncourt, Walter Chang, and Thien Huu Nguyen. 2021. MadDog: A Web-based System for Acronym Identification and Disambiguation. In *European Chapter of the Association for Computational Linguistics*.

[59] Amir Pouran Ben Veyseh, Franck Dernoncourt, Thien Huu Nguyen, Walter Chang, and Leo Anthony Celi. 2021. Acronym Identification and Disambiguation Shared Tasks for Scientific Document Understanding. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence*. CEUR-WS.org. http://ceur-ws.org/Vol-2831/paper33.pdf

[60] Amir Pouran Ben Veyseh, Franck Dernoncourt, Quan Hung Tran, and Thien Huu Nguyen. 2020. What Does This Acronym Mean? Introducing a New Dataset for Acronym Identification and Disambiguation. In *International Conference on Computational Linguistics*.

[61] 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 (apr 2017), e79–e86. https://doi.org/10.1093/jamia/ocw109

[62] Yonghui Wu, Jun Xu, Yaoyun Zhang, and Hua Xu. 2015. Clinical Abbreviation Disambiguation Using Neural Word Embeddings. In *Proceedings of the Workshop on Biomedical Natural Language Processing*. Association for Computational Linguistics, Beijing, China, 171–176. https://doi.org/10.18653/v1/W15-3822

[63] Anna Yarygina and Natalia Vassilieva. 2012. High-recall Extraction of Acronym-definition Pairs with Relevance Feedback. In *Joint Extending Database Technology and International Conference on Database Theory Workshops*. ACM, New York, NY, USA, 21–28. https://doi.org/10.1145/2320765.2320781

[64] Hong Yu, Won Kim, Vasileios Hatzivassiloglou, and John Wilbur. 2006. A Large Scale, Corpus-Based Approach for Automatically Disambiguating Biomedical Abbreviations. *ACM Transactions on Information Systems* 24, 3 (jul 2006), 380–404. https://doi.org/10.1145/1165774.1165778

[65] Danqing Zhu, Wangli Lin, Yang Zhang, Qiwei Zhong, Guanxiong Zeng, Weilin Wu, and Jiayu Tang. 2021. AT-BERT: Adversarial Training BERT for Acronym Identification Winning Solution for SDU@ AAAI-21. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence*. CEUR-WS.org. http://ceur-ws.org/Vol-2831/paper28.pdf