Document-Level Definition Detection in Scholarly Documents: Existing Models, Error Analyses, and Future Directions

Dongyeop Kang ♡ Andrew Head ♡ Risham Sidhu ♡ Kyle Lo ♢ Daniel S. Weld ◦ Marti A. Hearst ♡

♡ University of California, Berkeley, ◦ Allen Institute for AI, ♡ University of Washington
{dongyeopk,andrewhead,rishamsidhu,hearst}@berkeley.edu
{kylel,danw}@allenai.org

Abstract

The task of definition detection is important for scholarly papers, because papers often make use of technical terminology that may be unfamiliar to readers. Despite prior work on definition detection, current approaches are far from being accurate enough to use in real-world applications.

In this paper, we first perform in-depth error analysis of the current best performing definition detection system and discover major causes of errors. Based on this analysis, we develop a new definition detection system, HEDDEx, that utilizes syntactic features, transformer encoders, and heuristic filters, and evaluate it on a standard sentence-level benchmark. Because current benchmarks evaluate randomly sampled sentences, we propose an alternative evaluation that assesses every sentence within a document. This allows for evaluating recall in addition to precision.

HEDDEx outperforms the leading system on both the sentence-level and the document-level tasks, by 12.7 F1 points and 14.4 F1 points, respectively. We note that performance on the high-recall document-level task is much lower than in the standard evaluation approach, due to the necessity of incorporation of document structure as features. We discuss remaining challenges in document-level definition detection, ideas for improvements, and potential issues for the development of reading aid applications.

1 Introduction

Automatic definition detection is an important task in natural language processing (NLP). Definitions can be used for a variety of downstream tasks, such as ontology matching and construction (Bovi et al., 2015), paraphrasing (Hashimoto et al., 2011), and word sense disambiguation (Banerjee and Pedersen, 2002; Huang et al., 2019). Prior work in automated definition detection has addressed the domain of scholarly articles (Reiplinger et al., 2012; Jin et al., 2013; Espinosa-Anke and Schockaert, 2018; Vanetik et al., 2020; Veyseh et al., 2020).

Definition detection is especially important for scholarly papers because they often use unfamiliar technical terms that readers must understand to properly comprehend the article.

Despite recent advances in definition detection,
much work remains to be done before models are capable of extracting definitions with an accuracy appropriate for real-world applications. The first challenge is one of recall: existing systems are typically not trained to identify all definitions in a document, but rather to classify individual sentences arbitrarily sampled from a large corpus. The second challenge is one of precision: the state of the art misclassifies upwards of 30% of sentences (Veyseh et al., 2020). This begs the questions of why definition extractors fall short, and how these shortcomings can be overcome.

In this paper, we contribute the following:

• An in-depth error analysis of the current best-performing model. This analysis characterizes the state of the field and illustrates future directions for improvement;

• A new model, Heuristically-Enhanced Deep Definition Extraction (HEDDEx), that extends a state-of-the-art model with improvements designed to address the problems found in the error analysis. An evaluation shows that this improved model outperforms the state of the art by a large margin (+12.7 F1);

• An introduction of the challenging task of full-document definition detection. In this task, models are evaluated based on their ability to identify definitions across an entire document’s sentences. We believe this framing of definition detection is critical to preparing future algorithms for real-world use;

• A preliminary analysis of previous models and our model on the document-level definition detection task using a small test set of scholarly papers where every term and definition has been labeled. This analysis shows that HEDDEx outperforms the state of the art, while revealing opportunities for future improvements.

In summary, this paper draws attention to the work yet to be done in addressing the task of document-level definition detection for scholarly documents. We draw attention to the fact that a seemingly straightforward task like definition detection still poses significant challenges to NLP, and that this is an area that needs more focus in the scholarly document processing community.

2 Related Work

Definition detection has been tackled in several ways in prior research. The traditional rule-based systems (Muresan and Klavans, 2002; Westerhout and Monachesi, 2008; Westerhout, 2009a) used hand-written definition patterns (e.g., “is defined as”) and linguistic features (e.g., pronoun, verb, punctuation), providing high precision but low recall detection. To address the low recall problem, model-driven approaches (Fahmi and Bouma, 2006; Westerhout, 2009b; Navigli and Velardi, 2010; Reiplinger et al., 2012) were developed using statistical and syntactic features such as bag-of-words, sentence position, part-of-speech (POS) tags, and their combination with hand-written rules. Notably, Jin et al. (2013) used conditional random field (CRF) (Lafferty et al., 2001) to predict tags of each token in a sentence such as TERM for term tokens, DEF for definition tokens, and O for neither. Recently, sophisticated neural models such as convolutional networks (Espinosa-Anke and Schockaert, 2018) and graph convolutional networks (Veyseh et al., 2020) have been applied to obtain better sentence representations in combination with syntactic features. However, our analysis found that the state-of-the-art is still far from solving the problem, achieving an F1 score of only 60 points on a standard test set.

3 Error Analysis of the Leading System

In order to inform our efforts to develop a more advanced system, we performed an in-depth error analysis of the results of the current leading approach to definition and term identification, the joint model by Veyseh et al. (2020). We analyzed the models’ predictions on the W00 dataset (Jin et al., 2013) since it matches our target domain of scholarly papers and is the dataset that the joint model was evaluated on. Of the 224 test sentences, the Veyseh et al. (2020) system got 111 correct. The first author annotated the remaining 113 sentences for which the algorithm was partially or fully incorrect to ascertain the root causes of the errors.

We discovered four (for terms) and five (for definitions) major causes for the erroneous predictions, as summarized in Table 3. We illustrate three exam-
[Equal] is open in something of type collection where that collection is a {partition of something}. 

Overgeneralization: technical term bias
- description (is)

A [graph-based operator] defines a transformation on a multi-document graph (MDG) $G$ which processes [some of its properties while reducing the number].

- Complicated sentence structure
  - <term> defines <def> parsing features

The Inductive Logic Programming learning method that we have developed enables us to automatically extract from a corpus N-V pairs whose elements are linked by one of the semantic relations defined in the qualia structure.

- Unfamiliar or unseen vocabulary
- Unseen patterns

<table>
<thead>
<tr>
<th>Sentences</th>
<th>Cause</th>
<th>Patterns</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Equal] is open in something of type collection where that collection is a {partition of something}.</td>
<td>• Overgeneralization: technical term bias</td>
<td>(none applicable)</td>
<td>?</td>
</tr>
<tr>
<td>A [graph-based operator] defines a transformation on a multi-document graph (MDG) $G$ which processes [some of its properties while reducing the number].</td>
<td>• Complicated sentence structure</td>
<td>&lt;term&gt; defines &lt;def&gt; parsing features</td>
<td></td>
</tr>
<tr>
<td>The Inductive Logic Programming learning method that we have developed enables us to automatically extract from a corpus N-V pairs whose elements are linked by one of the semantic relations defined in the qualia structure.</td>
<td>• Unfamiliar or unseen vocabulary</td>
<td>&lt;term&gt; that we have developed enables us to automatically &lt;def&gt; generalize patterns</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Sample annotations from our analysis of errors produced by Veyseh et al.’s (2020) joint model when extracting definitions from the W00 (Jin et al., 2013) dataset. Each row includes a sentence annotated with gold labels for terms and definitions, and the system’s predictions for [terms] and {definitions} (“Sentences”). Also shown is a class of error (“Cause”), surface patterns that we anticipate could be used to correct the detection of the definition (“Patterns”), and classes of improvements to make to the model (“Solutions”). The first row is an example of a false positive; the second row is a partially-correct prediction; and the third row is a false negative. A transcription error (‘axe’ instead of ‘are’) is retained from the dataset.

For each example, we also labeled surface patterns between the term and definition (e.g., “<term> defines <def>”), and potential algorithm improvements to address the underlying problem.

For instance, in the bottom-most example in Table 2, the system did not predict any term or definition, although the sentence includes the term “Inductive Logic Programming Learning method” and the definition “extract from a corpus...”. Our conjecture is that the underlying surface pattern is unseen in the training set and too complicated to be generalized; we annotate a potential solution as pattern generalization.

Top hypothesized causes of error (%)

<table>
<thead>
<tr>
<th>Cause</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overgeneralization: technical term bias</td>
<td>48.6%</td>
</tr>
<tr>
<td>Unfamiliar or unseen vocabulary</td>
<td>25.7%</td>
</tr>
<tr>
<td>Complicated sentence structure</td>
<td>12.9%</td>
</tr>
<tr>
<td>Entity detection</td>
<td>4.3%</td>
</tr>
</tbody>
</table>

Table 3: Top causes of errors for terms (top) and definitions (bottom)

We rank the causes of errors by frequency and summarize the results in Table 3. For detection of terms, nearly half of the error cases fall into overgeneralization of technical terms: overly predicting words like “equal” and “model” as terms (e.g., the top example in Table 2).

Proposed error correction solution types (%)

<table>
<thead>
<tr>
<th>Solution Type</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntactic (POS, parse tree, entity, acronym)</td>
<td>29.2%</td>
</tr>
<tr>
<td>Heuristics</td>
<td>23.6%</td>
</tr>
<tr>
<td>Better encoder/tokenizer, UNK</td>
<td>18.0%</td>
</tr>
<tr>
<td>Rules (surface patterns)</td>
<td>11.3%</td>
</tr>
<tr>
<td>Annotations*</td>
<td>9.4%</td>
</tr>
<tr>
<td>Pattern generalization</td>
<td>5.7%</td>
</tr>
<tr>
<td>Mathematical symbol detection</td>
<td>1.9%</td>
</tr>
<tr>
<td>More context</td>
<td>0.9%</td>
</tr>
</tbody>
</table>

Table 4: Proportions of proposed error correction solution types. Annotations* indicates extremely ambiguous cases even for humans, so additional human annotations are required to disambiguate them.

We again rank the error correction solutions by frequency (Table 4). We predict that 29% of errors can be fixed by informing the system about syntactic features of the sentence such as part-of-speech tags, parse tree annotations, entities, or acronyms for more accurate detection. Surprisingly, simple heuristics (e.g., stitching up discontiguous token spans) seem likely to be highly effective to address the errors in Table 3, such as discarding output that does not successfully predict both a term and a definition. In the next section, we implement the first three solution types on top of the state-of-the-art system and report the resulting performance improvements.
Figure 1: The HEDDEx model. The new modules developed in this work include the incorporation of syntactic features, the addition of pre-trained transformer encoders, and post-processing with heuristics.

4 Definition Sentence Detection Model

To address the errors identified in §3, we designed HEDDEx, a new sentence-level definition detection model. The model incorporates a set of syntactic features, heuristic filters, and encoders. Each of these was designed to address a common class of error revealed in the error analysis. The model achieves superior performance over the state of the art for the task of sentence-level definition detection.

4.1 Proposed Model: Heuristically-Enhanced Deep Definition Extraction (HEDDEx)

HEDDEx extends the joint model proposed by Veyseh et al. (2020). The joint model is comprised of two components. The first component is a CRF-based sequence prediction model for slot tagging. The model assigns each token in a sentence one of five tags: term (“B-TERM”, “I-TERM”), definition (“B-DEF”, “I-DEF”), or other (“O”). The second component is a binary classifier that labels each sentence as containing a definition or not.

HEDDEx has three new modules (Figure 1). First, it encodes input from a transformer encoder fine-tuned on the task of definition extraction, whereas the joint model encodes input from a combination of a graph convolutional network and a BERT encoder without fine-tuning.  

We evaluate several state-of-the-art encoders for this task, including BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and SciBERT (Beltagy et al., 2019).

Second, HEDDEx is provided with additional syntactic features as input. These features include parts of speech, syntactic dependencies, and the token-level labels provided by entity recognizers and abbreviation detectors (Schwartz and Hearst, 2003). The features are extracted using off-the-shelf tools like Spacy and SciSpacy (Neumann et al., 2019).

Third, the output of the CRF and sentence classifier is refined using heuristic rules. The rules clean up the slot tags produced by the CRF, and override predictions made by the sentence classifier. The rules include, among other rules:

- Do not classify a sentence as a definition if it only contains a term without a definition, or a definition without a term.
- Stitch up discontiguous token spans for terms and definitions by assigning all contiguous tokens between two term or definition labels the same label.

These three enhancements developed in HEDDEx were selected specifically to suit the shortcomings of the models identified in the error analysis (§3), leading to significant improvements for definition detection in our experiments.

4.2 Baseline Models

To evaluate the impact of these improvements on the definition detection task, HEDDEx was compared to four baseline systems: (1) DefMiner (Jin et al., 2013), a CRF-based sequence prediction model that makes use of hand-written features; (2) Li et al.’s (2016) model comprised of a CRF with an LSTM encoder; (3) GCDT (Liu et al., 2019a), a global and local context encoder; (4) Veyseh et al.’s (2020) joint model described above. The experimental setup for the models followed the setup described by Veyseh et al. (2020).

However, we note that we were unable to replicate the accuracy reported by Veyseh et al. (2020) when using the code provided by the authors.

https://spacy.io/
4.3 Metrics

The models were compared using a set of metrics for both slot tagging and sentence classification on the W00 test set. To evaluate the slot tagger, macro-averaged precision, recall, and F1 score were measured (column “Macro P/R/F” in Table 5). However, the Macro scores do not show performance specific to terms or definitions. Also, macro-averaging over the position tags (B, I) makes it difficult to interpret general performance. Therefore, we measured these three metrics only for term tags (“TERM P/R/F”); B-TERM and I-TERM, and definition tags (“DEF P/R/F”); B-DEF and I-DEF. To evaluate the precision of the bounds of term and definition spans, we also evaluated the degree of overlap between each detected term or definition span and the corresponding span in the gold dataset (“Partial F”). Furthermore, the accuracy of sentence classification was measured (column “Classification”). For each of these metrics, a higher score indicated superior performance. We averaged each score across 10-fold cross validation.

4.4 Setup

Due to computing limitations, we chose the best hyper-parameter set through parameter sweeping with HEDDE x with the BERT encoder only, and use the best hyperparameters for all other models. Here is the ranges of each parameter we tuned: batch sizes in {8, 16, 32}, number of training epochs [30, 50, 100] maximum length of sentences in {80, 256, 512}, learning rates in [{2,5}e-{4,5}].

The other parameters used as defaults in our experiments were as follows: the dropout ratio was 10%, the layer size for POS embeddings was 50, and the hidden size for slot prediction was 512. We follow the default hyper-parameters for each transformer model of each size (base or large) using HuggingFace’s transformer libraries.4

4.5 Results

Outcomes of the evaluation for all measurements are presented in Table 5. The pre-trained language model encoders (BERT, RoBERTa, SciBERT) achieve comparable performance to more complex neural architectures like the graph convolutional networks used in Veyseh et al.’s (2020). Models that included SciBERT (Beltagy et al., 2019), rather than BERT or ROBERTa, achieved higher accuracy on most measurements. We attribute this to the domain similarity between the scholarly documents that SciBERT was trained on, and those used in our evaluation.

With SciBERT as the base encoder, the incorporation of syntactic features led to further accuracy gains. Of particular note are the improvements in recall in term spans. During our evaluation, we observed that the gains from syntactic features were more pronounced for encoders with a small mode size (i.e., the “-base” models). We conjecture that this is because the larger encoder models were capable of learning comparable linguistic patterns to those captured by the syntactic features.

The addition of heuristic rules led to significant improvement (+11.8 Macro F1) over the combination of Joint and SciBERT. Given the modest improvement in term and definition tagging, we suspect that much of this improvement can be accounted for by the correction of position markers in the slot tags (i.e., distinguishing between B and I in the tag assignments).

In the following experiments, we call HEDDE x the combination of three components: the encoder (SciBERT or RoBOTa), syntactic features, and heuristic filters.

5 Document-Level Definition Detection

Although HEDDE x attains reasonable performance on individual sentences, it faces new challenges when applied to the scenario of document-level analysis. In this section, we evaluate sentence detection for full papers in two novel ways. First, we assess the precision of the HEDDE x model across all of the sentences of 50 documents in §5.1. Second, we assess both the precision and the recall of the algorithm across all of the sentences of 2 full documents (§5.2, S5.3).

5.1 Error Analysis on Predicted Definitions

To assess how well HEDDE x works at the document level, we randomly sampled 50 ACL papers from the S2ORC dataset (Lo et al., 2020), a large corpus of 81.1M English-language academic papers spanning many academic disciplines. We ran the pretrained HEDDE x model on every sentence of every document; if the model detected a term/definition pair, the corresponding sentence was output for assessment. (Note that this analysis can estimate precision but not recall, as false negatives are not detected.)
Table 5: Comparison of the accuracy of recent models and the HEDDEx model for the definition detection task. Asterisks (*) indicate that we reimplemented the model from the authors’ specification. Models were evaluated on the W00 (Jin et al., 2013) test set. Each score is averaged across 10-fold cross validation. Accuracy measurements include Precision, Recall, and F1-score. Each of these measurements is macro-averaged.

Table 6: Analysis of HEDDEx output on 50 ACL papers, ordered by frequency. N = 923.

Table 7: Analysis of span length of HEDDEx output on 50 ACL papers, ordered by frequency. N = 923.

5.2 Full Document Definition Annotation

Prior definition annotation collections select unrelated sentences from across a document collection. As mentioned in the introduction, we are interested in annotating full papers, which requires finding every definition within a given paper. Therefore, we created a new collection in which we annotate every sentence within a document, allowing assessment of recall as well as precision. Two annotators annotated two full papers using an annotation scheme similar to that used in DEFT (Spala et al., 2019) except for omitting cross-sentence links.

We chose to annotate two award-winning ACL papers: ELMo (Peters et al., 2018) and LISA (Strubell et al., 2018) resulting in 485 total sentences from which we identified 98 definitional and 387 non-definitional sentences. Similar to DEFT (Spala et al., 2019), we measured inter-annotator agreement using Krippendorff’s alpha (Krippendorff, 2011) with the MASI distance metric (Pas- sonneau, 2006). We obtained 0.626 for terms and
0.527 for definitions, where the agreement score for terms is lower than those in DEFT annotations (0.80). This may be because our annotations for terms include various types such as textual terms, acronyms, and math symbols, while terms in DEFT are only textual terms. The task was quite difficult: each annotator takes two and a half hours to annotate a single paper. Future work will include refining the annotation scheme to ensure more consistency among annotators and to annotate more documents.

5.3 Evaluation on Document-level Definitions
We evaluated document-level performance using the same metrics used in §4.3. All metrics were averaged over scores from 10-fold validation models. The ensemble model aggregates ten system predictions from the 10-fold validation models and choose the final label via majority voting. We use the best single system; HEDDEx but with RoBERTa, for model ensembling.

<table>
<thead>
<tr>
<th></th>
<th>Macro</th>
<th>TERM</th>
<th>DEF</th>
<th>Partial Clf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint model</td>
<td>36.0</td>
<td>30.1</td>
<td>38.1</td>
<td>34.1</td>
</tr>
<tr>
<td>HEDDEx w/ BERT</td>
<td>45.3</td>
<td>32.4</td>
<td>39.0</td>
<td>34.6</td>
</tr>
<tr>
<td>HEDDEx w/ RoBERTa</td>
<td>47.7</td>
<td>36.4</td>
<td>47.2</td>
<td>37.2</td>
</tr>
<tr>
<td>HEDDEx ensemble</td>
<td>50.4</td>
<td>38.7</td>
<td>49.5</td>
<td>39.0</td>
</tr>
</tbody>
</table>

Table 8:  Document-level evaluation on our annotated documents. F1 score is measured for every metric except for classification (Cf.), which uses accuracy.

Compared to the joint model by Veyseh et al. (2020), HEDDEx showed significant improvements on every evaluation metric, which is slightly larger than that of the sentence-level evaluation (Table 8). With model ensembling, compared to the state-of-the-art system, HEDDEx achieved gains by +14.4 Macro F1 points, +8.7 TERM F1 points, +11.4 DEF F1 points, +4.9 Partial Matching F1 points, and +3.0 classification accuracy scores.

Table 9: Low recall problem in document-level definition detection. We report precision, recall, and f1 scores on three metrics; Macro, TERM, and DEF, using our best system; HEDDEx ensemble.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro</td>
<td>55.3</td>
<td>46.7</td>
<td>50.4</td>
</tr>
<tr>
<td>TERM</td>
<td>44.8</td>
<td>34.0</td>
<td>38.7</td>
</tr>
<tr>
<td>DEF</td>
<td>55.6</td>
<td>44.7</td>
<td>49.5</td>
</tr>
</tbody>
</table>

However, document-level definition detection is a much harder task than sentence-level detection. Compared to the sentence-level task in Table 5, the document-level task showed relatively lower performance (73.4 Macro F1 in sentence-level versus 50.4 Macro F1 in document-level). In particular, recall is much lower than precision in the document-level task (Table 9), whereas in the sentence-level task, precision and recall are almost the same, indicating the necessity of incorporation of document structure as additional features (See further discussion in §6).

Table 10 shows the predicted terms and definitions as well as annotated gold labels. Acronym patterns (e.g., “biLM,” “WSD”), definition of newly-proposed terms (e.g., “LISA”), re-definition of prior work (e.g., “SQuAD,” “SRL,” Coreference resolution) and some of mathematical symbols were detected well. However, as sentences get more complex, the system made incorrect predictions. Additionally, sub-words or parentheses in abbreviations are sometimes partially predicted (e.g., the beginning of the word “pretrained” is cut off in the definition of “semi-supervised learning” in example 8 of Table 10).

However, the aforementioned problem of low recall is severe for this task, particularly since the model often fails to detect mathematical symbols or a combination of textual terms and mathematical symbols (e.g., “L-layer biLM”). Moreover, when a sentence contains multiple terms and/or multiple symbols together, the system only ever detects one of them.

6 Discussion
Detecting definitions is a very challenging task, and it is far from solved. Here we discuss remaining challenges and ideas for improvements, and motivate the need for high-precision, high-recall definition detection in an academic document reading aid application.

Outstanding technical challenges include:

- **Poor recognition of mathematical symbols:** As shown in our experiment, our system is less successful at detecting math symbols than textual terms. This is mainly because the lack of coverage of mathematical symbols in our training dataset (W00).

- **Contextual disambiguation of symbols:** In our study, we observe that some symbols are used with multiple meanings. For example,
<table>
<thead>
<tr>
<th>Predicted definition sentences</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Our word vectors are learned functions of the internal states of a [deep bidirectional language model] (biLM), which is pre-trained on a large text corpus.</td>
<td>term</td>
</tr>
<tr>
<td>2 We use vectors derived from a bidirectional LSTM that is trained with a coupled [language model] (LM) objective on a large text corpus.</td>
<td>term</td>
</tr>
<tr>
<td>3 Using intrinsic evaluations, we show that the higher-level [LSTM states] capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised word sense disambiguation tasks) while lower-level states [model aspects of syntax].</td>
<td>term</td>
</tr>
<tr>
<td>4 We first show that they can be easily added to existing models for six diverse and challenging language understanding problems, including textual entailment, question answering and sentiment analysis.</td>
<td>term</td>
</tr>
<tr>
<td>5 For tasks where direct comparisons are possible, outperforms [CoVe] CITATION, which [computes contextualized representations using a neural machine translation encoder].</td>
<td>term</td>
</tr>
<tr>
<td>6 context2vec CITATION uses a bidirectional Long Short Term Memory LSTM ; CITATION to encode the context around a pivot word .</td>
<td>term</td>
</tr>
<tr>
<td>7 Unlike most widely used word embeddings CITATION, word representations are functions of the entire input sentence, as described in this section.</td>
<td>term</td>
</tr>
<tr>
<td>8 This setup allows us to do [semi-supervised learning], where the biLM is pretrained at a large scale} (Sec. SECTION) and easily incorporated into a wide range of existing neural NLP architectures (Sec. SECTION).</td>
<td>term</td>
</tr>
<tr>
<td>9 Given a sequence of ( N ) tokens, ((t_1, t_2, \ldots, t_N)), a forward language model computes the probability of the sequence by modeling the probability of token ( t_k ) given the history ((t_1, \ldots, t_{k-1})).</td>
<td>term</td>
</tr>
<tr>
<td>10 A [backward LM] is [similar to a forward LM, except it runs over the sequence in reverse, predicting the previous token given the future context].</td>
<td>term</td>
</tr>
<tr>
<td>11 A [biLM] [combines both a forward and backward LM].</td>
<td>term</td>
</tr>
<tr>
<td>12 where ( \text{[LM]} ) is {the token layer} and ( h_k^{LM} = [H_k^{LM}, B_k^{LM}] ), for each biLM layer.</td>
<td>symbol</td>
</tr>
<tr>
<td>13 In (EQUATION), ( s^{\text{task}} ) are {softmax-normalized weights} and the scalar parameter ( \gamma^{\text{task}} ) allows the task model to scale the entire vector.</td>
<td>symbol</td>
</tr>
<tr>
<td>14 For each token ( t_k ), a ( L )-layer biLM computes a set of ( 2L + 1 ) representations EQUATION where ( h_0^{LM} ) is the token layer and ( h_k^{LM} = [H_k^{LM}, B_k^{LM}] ), for each biLM layer.</td>
<td>symbol, term</td>
</tr>
<tr>
<td>15 In (EQUATION), ( s^{\text{task}} ) are softmax-normalized weights and the {scalar parameter ( \gamma^{\text{task}} ) } allows the task model to scale the entire vector.</td>
<td>symbol, term</td>
</tr>
<tr>
<td>16 The [Stanford Question Answering Dataset (SQuAD) CITATION] [contains 100K+ crowd sourced question-answer pairs where the answer is a span in a given Wikipedia paragraph].</td>
<td>term</td>
</tr>
<tr>
<td>17 [Textual entailment] is [the task of determining whether a &quot;hypothesis&quot; is true, given a &quot;premise&quot;].</td>
<td>term</td>
</tr>
<tr>
<td>18 The [Stanford Natural Language Inference (SNLI) corpus CITATION] [provides approximately 550K hypothesis/premise pairs].</td>
<td>term</td>
</tr>
<tr>
<td>19 A [semantic role labeling (SRL) system] [models the predicate-argument structure of a sentence], and is often described as answering.</td>
<td>term</td>
</tr>
<tr>
<td>20 CITATION modeled [SRL] [as a BIO tagging problem and used an 8-layer deep biLSTM with forward and backward directions interleaved], following CITATION.</td>
<td>term</td>
</tr>
<tr>
<td>21 [Coreference resolution] is [the task of clustering mentions in text that refer to the same underlying real world entities].</td>
<td>term</td>
</tr>
<tr>
<td>22 The [CoNLL] 2003 NER task CITATION [consists of newswire from the Reuters RCV1 corpus tagged with four different entity types (PER, LOC, ORG, MISC)].</td>
<td>term</td>
</tr>
<tr>
<td>23 The [fine-grained sentiment classification] task in the Stanford Sentiment Treebank SST-5 [involves selecting one of five labels (from very negative to very positive) to describe a sentence from a movie review].</td>
<td>term</td>
</tr>
<tr>
<td>24 The sentences contain diverse linguistic phenomena such as idioms and complex syntactic constructions such as negations that are difficult for models to learn.</td>
<td>multi-term</td>
</tr>
<tr>
<td>25 Intuitively, the [biLM] must be [disambiguating the meaning of words using their context].</td>
<td>term</td>
</tr>
<tr>
<td>26 a fine grained [word sense disambiguation] ([WSD]) task and a POS tagging task.</td>
<td>term</td>
</tr>
</tbody>
</table>

Table 10: All predicted and gold label terms and definitions for the ELMo paper (Peters et al., 2018). Gold labels for terms are underlined and for definitions are dashed. System-predicted [terms] are placed in [boldfaced brackets] and [definitions] are placed in [italicized braces]. “CITATION,” “SECTION,” and “EQUATION” are placeholders inserted for citations, section numbers, and display equations. “Type” means term type.
symbol $T$ in the LISA paper is used for *token representation* as well as *matrix transpose*. Disambiguating terms based on context of use will be an interesting future direction.

- **Description vs Definition**: In our annotation and error analysis, the most difficult distinction was between definitions and descriptions — they have quite similar surface patterns, although they refer to entirely different meanings. For instance, a definition is the exact denotation of a word, while a description is more detailed so it can change from person to person. Training a model that distinguishes these types should lead to better and more useful results.

Potential ideas for improvements of the system include:

- **Annotation of mathematical definitions**: A solution for poor math symbol detection is to annotate math symbols and use them for our training. One option is to add span information to the binary judgements of the math definition collection of Vanetik et al. (2020).

- **Utilization of document-level features**: Document structure and positional information may improve detection. For instance, the section information of a term would be an important feature to recognize whether a term is first introduced or not.

- **Data augmentation or domain-specific fine-tuning for high-recall system**: Existing definition training sets are small (W00 contains only 731 definitional sentences). To obtain more data, the data can be augmented via seed patterns or fine-tuning with existing language models such as SciBERT.

Lastly, as the performance of definition detection systems increases, these systems can be applied to real-world reading or writing aids. We discuss potential issues of our system in the realistic settings:

- **Metrics for usefulness**: Currently, we measure precision, recall, and F1 scores with the document-level annotations. However, we have not explored the usefulness of the predicted definitions for readability, when they are used in real-world applications like ScholarPhi (Head et al., 2020). Deciding when and where to show definitions based on context and information density still remains an important future direction.

- **Categorization of definitions**: We observe that in fact, terms and definitions can be grouped into multiple categories: short names, acronyms, textual definitions, formula definitions, and more. Automatically categorizing these and showing structured definitions might be helpful for organizing and ranking definitions in a user interface.

- **Repeated definitions and terms within documents**: We observed a pattern in which the same term is referred to multiple times in slightly different ways. Newly proposed terms are especially likely to exhibit this pattern. Grouping and summarizing these in a *glossary table* would be helpful for an academic document reader application.

### 7 Conclusion

This work sets the stage for bridging the gap between a well-known NLP task; definition detection, and real-world applications of the technique that requires both high precision and high recall. To achieve the goal, we proposed a more realistic setup for definition detection task called *document-level definition detection* that requires high recall, mathematical symbol recognition, and document-level feature engineering. Our proposed definition detection system HEDDEx achieved significant gains in both sentence-level and document-level tasks. Yet, the problem is far from being solved. We suggest that better coverage of variability of expression, recognition of mathematical symbols and notation, and other nuances of the task must still be addressed.

### Acknowledgements

We like to thank Amir Pouran Ben Veyseh for his help in sharing his code, preprocessed data, and general advice on replication of his work. We also thank Raymond Fok, Vivek Aithal, Hearst lab members at UC Berkeley and anonymous reviewers at SDP 2020 for their helpful comments. This research receives funding from the Alfred P. Sloan Foundation, the Allen Institute for AI, Office of Naval Research grant N00014-15-1-2774, NSF Convergence Accelerator award 1936940, NSF RAPID award 2040196, and the University of Washington Washington Research Foundation/Thomas J. Cable Professorship.
References


Claudio Delli Bovi, Luca Telesca, and Roberto Navigli. 2015. Large-scale information extraction from textual definitions through deep syntactic and semantic analysis. Transactions of the Association for Computational Linguistics, 3:529–543.


Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. Scispacy: Fast and robust models for

Rebecca J 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 (LREC’06)*.


