Learning from Limited Labels for Long Legal Dialogue

Jenny Hong
Stanford University
jennyhong@cs.stanford.edu

Derek Chong
Stanford University
derekch@stanford.edu

Christopher D. Manning
Stanford University
manning@cs.stanford.edu

Abstract

We study attempting to achieve high accuracy information extraction of case factors from a challenging dataset of parole hearings, which, compared to other legal NLP datasets, has longer texts, with fewer labels. On this corpus, existing work directly applying pretrained neural models has failed to extract all but a few relatively basic items with little improvement over rule-based extraction. We address two challenges posed by existing work: training on long documents and reasoning over complex speech patterns. We use a similar approach to the two-step open-domain question answering approach by using a Reducer to extract relevant text segments and a Producer to generate both extractive answers and non-extractive classifications. In a context like ours, with limited labeled data, we show that a superior approach for strong performance within limited development time is to use a combination of a rule-based Reducer and a neural Producer. We study four representative tasks from the parole dataset. On all four, we improve extraction from the previous benchmark of 0.41–0.63 to 0.83–0.89 F1.

1 Introduction

In many judicial processes such as legal hearings and criminal trials, decisions are made as a result of lengthy dialogues, in which case factors are discussed in great detail. To study such dialogues, scholars typically invest immense effort to hand label a small number of transcripts with some case factors; the factors are then used in downstream analysis. In most cases, the sheer length of transcribed conversational text all but prohibits any large-scale analysis of the process. Information extraction over dialogues can assist in identifying the underlying factors of a case from transcripts.

The benefits of information extraction are twofold. Automating the extraction of case factors means that a historical legal analysis can now be comprehensive, containing all available transcripts, rather than being limited to the several dozen or hundred transcripts that a single researcher can label by hand. The second advantage is to open the door to *counterdata* applications in law (D'ignazio and Klein, 2020). To date, most machine learning applications in the law have been predictive: given case factors up front, make a prediction of an outcome. In domains where case factors cannot or should not be known prior to the hearing, information extraction can produce case factors *after* a hearing, which enables machine learning to play an alternative role to the role of prediction, the role of oversight (Bell et al., 2021). In our application, information extraction allows the public to audit the parole process, whose case records are otherwise locked away in a filing cabinet.

To be useful for such downstream research, the consensus in legal domain NLP is that information extraction should produce labels that achieve an F1 of at least 0.80 (Hendrycks et al., 2021; Hong et al., 2021). Our corpus, a set of historical California parole hearings, is a particularly difficult application, but also representative of many challenges in criminal law: (1) Parole hearings are longer than documents in existing benchmarks. (2) Existing benchmarks source from written text; parole documents are loosely-structured dialogue. (3) Existing benchmarks contain at least an order of magnitude more labels. (4) Information extraction from formal written documents centers around named entities and relation extraction. By contrast, much of the text in the criminal context serves the purpose of surfacing, discussing, and correcting case factors, which are not necessarily relational. This means parole hearings pose both extractive and abstractive tasks, often across multiple sentences, which is known to be challenging even in more structured settings (Wang et al., 2021).

The scarcity of labels and specificity of the domain suggest that subject matter experts (SMEs) can be helpful. On the parole corpus, weak supervision-based data programming approaches (Ratner et al., 2016; Zheng et al., 2019) achieve F1 scores of only 0.41–0.63 (Hong et al., 2021). We propose an alternative way to involve SMEs, in which we split the problem into two components: a Reducer model which extracts relevant text segments from a hearing, and a Producer model which generates answers from the text segments selected by the Reducer. Our methods effectively achieve extraction at 0.83–0.89 F1.

We show that using an approach with a rule-based Reducer and neural Producer outperforms other commonly-used approaches. Focusing SME effort on developing rules for the Reducer is thus more time-efficient than requiring SMEs to provide additional target labels, whether manually or via data programming. With quality text segments, a neural Producer model can be effectively fine-tuned on just one thousand labels.

2 Related Work

A review of data programming literature suggests that semi-supervised techniques might be a good fit for our problem space. Several existing pipelines combine a limited amount of training data, rule-based systems and neural models to achieve strong results on benchmark datasets (Maheshwari et al., 2020) and in various medical fields (Ling et al., 2019; Smit et al., 2020; Dai et al., 2021). By comparison, weak supervision-based data programming methods tend to focus on bootstrapping in the absence of data (Ratner et al., 2017, 2018), which is a nontrivial performance constraint.

Regardless of supervision strength, an architecture based on rule-based systems may be useful for generating "candidates" as input to downstream neural models; Zhang et al. (2019) explores the time efficiency of manual labeling compared with rule-writing (via regular expressions) for named entity recognition (NER), where results are compared over a bidirectional LSTM-based classifier, finding that in most circumstances, a combination of rule-based and machine-learning classifiers optimizes human time investment.

We therefore adopt the approach of using a rule-based system for candidate generation. One new challenges with our corpus is that parole hearings generally center around one individual, so the candidates for downstream models are not named entities, but more loosely defined segments of the hearing. Compared to NER, there is less prior work exploring rule-based methods for more general retrieval and segmentation.

Our goal of achieving 0.80 F1 in an abstractive format is currently beyond the capabilities of state-of-the-art (SOTA) neural models on comparable tasks, only one of which is in the legal domain.

On Natural Questions (NQ; Kwiatkowski et al., 2019), SOTA models achieve F1 scores of 0.79 and 0.64 on its long and short answer tasks, respectively. However, NQ is purely extractive and averages only 7,300 words per input. On the Doc2EDAG financial statements dataset (Zheng et al., 2019), the Graph-based Interaction model with a Tracker (Xu et al., 2021) surpasses 0.80 F1 when extracting events from documents averaging 912 tokens in length, but this SOTA result drops to 0.76 F1 in the longest quartile. On Open-Domain Question Answering, the SOTA Dense Passage Retrieval (Karpukhin et al., 2020) has an extractive top-5 accuracy of just 0.66. For downstream applications, a model must have a robust top-1 accuracy.

The closest comparable legal dataset is the Contract Understanding Atticus Dataset (CUAD) (Hendrycks et al., 2021). Over CUAD, a SOTA model like RoBERTa (Liu et al., 2019) achieves a lower, and extractive, question answering performance of 0.80 recall at 0.31 precision, representing an F1 score of only 0.45, with documents still averaging one-quarter the length of parole transcripts.

3 Data

We have obtained a corpus of 35,105 parole hearing transcripts, averaging 18,499 words each from 2007–2019.[1] Each hearing is a dialogue, primarily between one or more commissioners and the parole candidate. Most case factors are embellished with history and context, which is important for the procedure of a parole hearing, but challenging for information extraction. Hong et al. (2021) identified eleven fields for information extraction. We study the four fields that the previous study failed to extract with near 0.80 F1: `job_offer` (whether the parole candidate has a job offer upon release), `edu_level` (the candidate's educational level), `risk_assess` (a psychological assessment score), and `last_writeup` (the date of the candidate's last disciplinary writeup in prison).

[1]Transcripts may be requested from the California Department of Corrections and Rehabilitation under the California Public Records Act.

COMM: Dr. [REDACT], R-E-D-A-C-T, found you to be a moderate risk and also diagnosed you with anti-social personality disorder.

(a) Example of passage discussing `risk_assess`, the Comprehensive Risk Assessment score assigned to a parole candidate by a psychologist during an evaluation conducted leading up to the hearing.

COMM: When you were going to school, everything was -- how far did you get in school?
CAND: Junior high.
COMM: Okay. Junior high, okay. And have you gotten education in prison?

(b) Example of passage discussing `edu_level`, the candidate's level of education. The passage continues for several more conversational turns, in which the commissioner and the candidate discuss various educational programs.

COMM: And -- um -- if you are paroled or -- pardon me -- if you are deported to [REDACT], what is your plan?
CAND: Well -- um -- I had a couple of offers from there -- um -- I would have to -- uh -- check out the -- um -- [REDACT] center and maybe they could help me -- you know -- train me to get a job there and get my life together.

(c) Example of passage discussing `job_offer`, whether the candidate has a job offer upon release.

COMM: So when's your last 115?
CAND: Uh, when they had a -- we had a work -- had a work strike around here. That was the last 115 I remember. I forgot what -- what year it was.
COMM: I'm showing one from maybe January of 2010 with a mattress.
CAND: Oh, I didn't realize it was a 115.

(d) Example of passage discussing `last_writeup`, the date of the candidate's last disciplinary infraction, or Form 115, in prison.

Figure 1: Example passages of the four features we study. The speaker `COMM` refers to the presiding commissioner, and the speaker `CAND` refers to the parole candidate.

Figure 1 shows examples of how these four features arise in dialogue. On average, each annotator takes forty minutes to label a transcript. Only 3% of the dataset is labeled: `job_offer`, `edu_level`, and `risk_assess` each have 1,173 training examples and 106 validation examples, whereas `last_writeup` has 563 and 48, respectively. The corpus also includes 218 transcripts with labeled spans, i.e. the sentences from which the correct label was determined.

4 Methods

We use a Reducer-Producer paradigm (Figure 2) in the spirit of the Document Retriever-Reader model used in open-domain question answering (ODQA; Chen et al., 2017; Das et al., 2019), with two differences: (1) The Reducer selects one or more relevant passages from within a *single* document (Clark and Gardner, 2017; Krishna et al., 2021), and (2) the Producer model is not necessarily a QA model. We use separate Reducers and Producers for each field. Prior applications of data programming to this corpus used SMEs to write noisy labels for training a neural model; it does not significantly reduce the input text into shorter segments and instead relies on an end-to-end neural approach (Hong et al., 2021). By contrast, our approach uses SMEs to focus on the smaller task of reducing input text and relies on only gold labels, however few, for training the neural model. One subproblem is designed to be tractable for an SME (the Reducer), and the other for a pretrained language model (the Producer).

4.1 Reducer

The SME (1) encodes keywords and patterns into programmatic rules (Zhang et al., 2019), and (2) evaluates the rules against silver-standard metrics. The SME examines any errors and repeats the process until the development subset is covered to >95% recall on silver metrics.

Rules. The SME uses keywords to generate candidate segments and candidate substrings (e.g., for risk assessments, "low" is interesting, but only if it occurs in the proximity of "risk"), sequenced in order of increasing breadth and decreasing precision (Zhang et al., 2019). The framework provides high-level functions that enable SMEs to easily operate on pipelines of candidate segments, filtering in or out, splitting, deoverlapping, and limiting results to create a high-quality reduced output passage.

Evaluation. We reserve the 218 transcripts with labeled spans to serve as a held-out evaluation set. For intermediate SME evaluation and iterations, we use three silver-standard evaluations as a proxy for true Reducer performance: (a) the percentage of results with empty outputs, (b) whether true labels (and common synonyms) appear within reduced passages, and (c) performing interim Producer fine-tuning runs, and evaluating end-to-end performance across a set of hyperparameter sweeps.

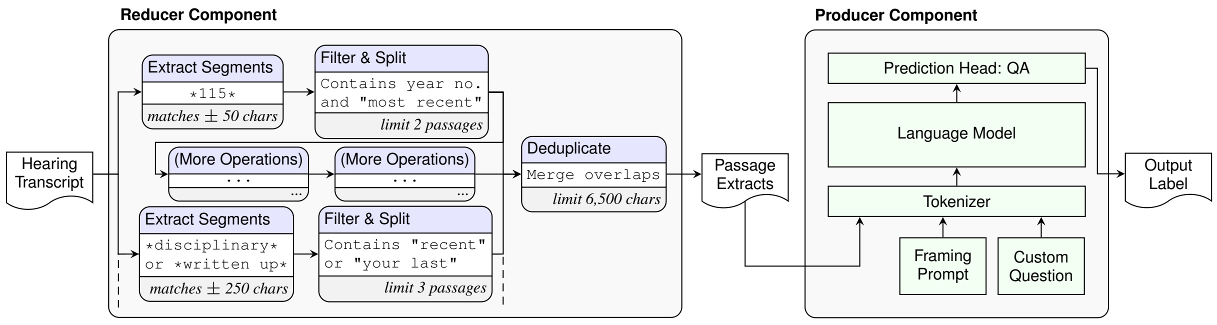

Figure 2: Reducer-Producer architecture sketch for the `last_writeup` field. The Reducer is entirely rule-based, with a few high-level operations over various regular expressions. The Producer is entirely neural and builds on a pretrained language model.

4.2 Producer

We write several simple rule-based Producers to build an understanding of the problem space, and then fine-tune pretrained language models on the passages returned by the Reducer.

Choice of language model. To ensure high training efficacy, we identify the smallest language model that meets the required benchmark in the general case. We evaluate a range of models' capabilities on a small task: For each of the four fields, we identify ten transcripts with particularly challenging dialogue (see Appendix C for examples). We manually extract passages from each transcript and benchmark each language model on its average zero-shot classification accuracy on all 40 passages, across 25 random seeds.

Choice of prediction heads. Fields with a small, fixed set of values are a good fit for a classification head (CLS), such as `edu_level` which is grouped into four categories, and `risk_assess`, for which a psychologist ascribes one of five possible risk levels. Fields with an open-ended set of values may be more suited to the masked language model (MLM) (Hermann et al., 2015; Hill et al., 2016; Chen, 2018; Devlin et al., 2019) or question answering (QA) approach, e.g., `last_writeup` can be any year from 1960–2020.

The MLM and QA heads require a user-defined prompt, which are not always natural for all fields. For example, for `job_offer`, we prompt MLM with token choices, e.g. "*Commissioner: As to whether you have a job offer lined up: You have [one / none].*"). For `last_writeup`, where the correct year exists within each passage, we try various prompts, such as "Your last writeup was in `[MASK]`". We chose prompts with good fine-tuning performance on training data, e.g. for `last_writeup`, we use the prompt "Ignoring chronos and 128s, your most recent 115, RVR (rule violation report) occurred in: `[MASK]`"). QA requires a question formulation, for some fields, we augment QA heads with a prefix sentence containing tokens representing all of the current field's possible classes, a technique used in QA benchmarks such as CoQA (Reddy et al., 2019) and BoolQ (Clark et al., 2019), which enables extractive models to always return values from desired classes.

We tried using a multiple choice reading comprehension (MRC) head (Richardson et al., 2013; Lai et al., 2017; Chen, 2018), which proved to be an an elegant way of grounding the model, with similarities to contrastive learning, and able to generate dynamic classification options, e.g. unlike year classification, MRC choices are *only* the year that appear in the passage. However, MRC requires a full backpropagation across the entire model for each option of every question, which is memory-intensive for passages where over a dozen options might exist per question, and unnecessarily slow even with gradient accumulation. We do not include MRC in our results.

Training details. We use base models from the HuggingFace Transformers library (Wolf et al., 2020), applying standard hyperparameter ranges (Sun et al., 2019) and techniques for training BERT-based models, such as the use of a slanted triangular learning rate. However, we set batch size to 1 and use gradient accumulation to simulate a larger batch size, in order to allow Reducer outputs to be as large as possible (approximately 1,500 tokens for RoBERTA + BigBird Base on a 16GB GPU) without affecting training performance. We ran hyperparameter sweeps for approximately six hours per field on a NVIDIA Tesla V100 GPU.

	Prev F1	Train F1	Val F1	Producer Model	Head
risk_assess	0.53	0.86	**0.83**	Rules	N/A
last_writeup	0.42	0.86	**0.84**	RoB+BB	MLM
edu_level	0.41	0.98	**0.84**	RoB+BB	CLS
job_offer	0.63	0.96	**0.89**	RoB+BB	QA

Table 1: Overall results. Previous best results are from Hong et al. (2021). RoB + BB = RoBERTa + BigBird.

	RL-R	R2-R	BoW-R
`risk_assess`	0.85	0.76	0.88
`last_writeup`	0.87	0.76	0.91
`edu_level`	0.92	0.82	0.95
`job_offer`	0.87	0.72	0.92

Table 2: Evaluating Reducers on labeled spans: Rouge-L and Rouge-2 Recall, Bag-of-Words Recall.

Model Family	Model Variant	Size	Max Len	Benchmark Score
BERT	Base Cased	108M	512	26.7 ± 9.4
	Large Cased	334M	512	33.0 ± 8.0
RoBERTa	Vanilla Base	125M	512	29.8 ± 8.1
	Vanilla Large	355M	512	28.3 ± 8.7
	BigBird Base	**128M**	**4,096**	$\mathbf{30.5 \pm 5.6}$
	BigBird Large	360M	4,096	33.0 ± 6.0
Transformer-XL	Vanilla Base	284M	N/A	32.1 ± 7.6
	XLNet Base	117M	N/A	29.1 ± 5.9
	XLNet Large	361M	N/A	30.2 ± 5.9
GPT	GPT2 Base	124M	2,048	32.2 ± 6.3
	GPT2 Medium	355M	2,048	33.5 ± 5.6
	GPT2 Large	774M	2,048	34.0 ± 7.1
	GPT-Neo 1.3B	1.32B	2,048	34.2 ± 6.3

Table 3: Zero-shot language model performance (average classification accuracy) on a benchmark of complex, challenging passages, over 25 random seeds.

5 Results

Our methods achieve the 0.80 F1 benchmark[2][3] for all four fields, as shown in Table 1. One rule-based Producer achieved an F1 of 0.83 for `risk_assess`, which narrowly outperformed RoBERTa + BigBird model performance of 0.81 F1. However, all other rule-based Producer attempts fell near or below the "Previous F1" mark on their tasks. The `risk_assess` task lends itself to rule-writing, because its values are restricted to combinations of "low," "moderate," and "high", and there are a few phrasings that are commonly used (e.g., "Overall, your risk was low to moderate"). By comparison, neural models may have been confused by the multiple other types of psychological assessments that occur in the text (e.g., PCL-R, HCR-20, LS/CMI), which are all assessed on the same "low," "moderate," and "high" scale.

5.1 Standalone Reducer Performance

Table 2 shows the Reducer's performance on three different measures of recall on the 218 labeled spans. We focus on recall because a Producer can still perform well on a short input even if there are occasional spurious phrases. Also, correct answers are not necessarily unique; labeled spans often point to a single sentence, whereas a fact may be repeated multiple times during the course of a hearing. The Reducer may select *a* correct span, but not the exact sentence selected by the annotator. Recall sidesteps the former issue and slightly mitigates the incorrect penalty imposed by the latter, as similar words may be used in both spans.

The Rouge-L recall ranges from 0.85–0.92: the Reducer frequently finds the exact set of sentences annotated by a human labeler. The Rouge-2 recall is lower, from 0.72–0.82: when the Reducer fails to find the exact sentences, the phrasing of its result is different. However, the bag-of-words recall is still high: 0.88–0.95, which means that the Reducer tends to finds sentences that use almost the same words, if not in the exact same order.

Given the span labeling issue described above, Table 2 is almost certainly an underestimate of Reducer performance. This is supported by other assessments of Reducer performance: end-to-end F1 scores of 0.83–0.89 are effectively a guarantee on the lower bound of Reducer performance, and based on the error analysis in Section 5.4, only a small fraction of errors were due to the Reducer. This implies *significantly* higher true recall scores. This is also in line with our silver-standard Reducer evaluations, which are consistently above 0.95.

5.2 Language Model Benchmarks

Benchmark performance for each language model is provided in Table 3. Figure 3 plots model performance against size and shows power-law scaling characteristics, a known feature of neural language models (Kaplan et al., 2020).

Given the relatively small range in performance between the models in our evaluation set (7.5%

[2]F1 scores are calculated on *exact match* for all prediction heads, instead of the relatively easier bag-of-words metric used in the extractive setting, or precision at 0.80 recall (Hendrycks et al., 2021). This is a more accurate measurement of abstractive performance, which is essential to downstream results.

[3]Related existing work reports F1, but F1 is an imperfect proxy for the impact of errors for downstream analyses. Any application that seeks to use extracted data should perform its own analysis to understand the relative costs of, for example, false positives versus false negatives for a given field.

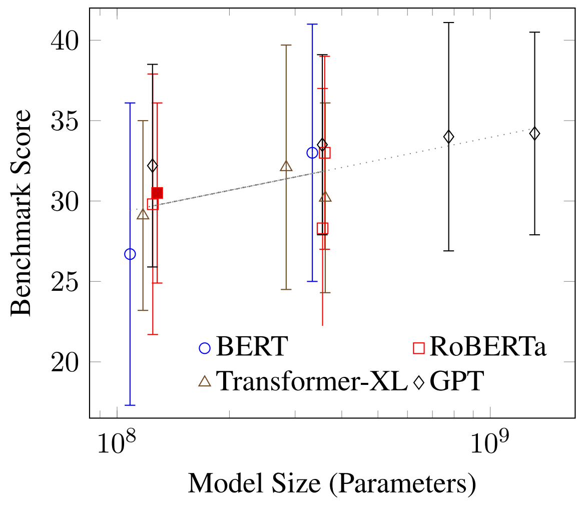

Figure 3: Performance on benchmark from Table 5.2 versus model size.

Hyperparameter	Value(s)
Learning Rate	5e-4 to 5e-7
Batch Size (Accumulative)	1, 2, 4, 8, 16
Number of Epochs	6 to 10
LR Warmup Epochs	0.4, 0.6, 0.8, 1.0, 1.2
Dropout	0.1
Adam Optimizer	$\beta_1 = 0.9, \beta_2 = 0.999$

Table 4: Hyperparameter sweep configurations for prediction head selection exercise.

	CLS	MLM	QA
`last_writeup`	0.76	*0.79*	*0.82*
`edu_level`	**0.82**	0.43	0.70
`job_offer`	0.83	0.69	**0.89**

Table 5: The effects of different prediction heads on Validation F1 scores (results in italics are not definitive, as MLM outperforms QA on end-to-end evaluation).

across all model families and variants), we also run some supplementary tests, finding that (a) models pretrained on question answering datasets performed 10–15% better in this setting, but a comprehensive evaluation was not feasible as QA outputs are extractive and require manual assessment, and (b) large GPT models performed dramatically better in the few-shot setting, with GPT3 performing at 90–100% accuracy on some problems.

We ultimately use RoBERTa + BigBird Base (RoB + BB; Zaheer et al., 2020) as our default model due to its balance of long input length, low computation requirements, and performance. This model supports inputs of up to 4,096 tokens, allowing the Reducer to provide multiple candidate passages without having to split input into multiple model calls and integrate a la Clark and Gardner (2017). It is in the smallest size class of the models tested, facilitating the fine-tuning of large input passages within GPU memory limits. Within its size class, RoB + BB is the second-best performer, performing within 2–3% of models 2–3x its size. Compared to the top performer (GPT2), BERT is known to have better versatility on downstream tasks (Klein and Nabi, 2019) and well-explored fine-tuning characteristics (Sun et al., 2019).

5.3 Prediction Heads

We evaluate the gains from prediction head choice by performing 25 fine-tuning runs for each combination of field and head and reporting the highest validation F1 score achieved for each. To ensure test fairness within a reasonable amount of computation, each run uses a random configuration from Table 4. F1 scores are recorded at the point where validation loss is at a minimum.

Table 5 shows the performance of each prediction head on each field. `edu_level` and `job_offer` performed comparably to the main runs in Table 1. `last_writeup` performed best under a question answering head during this exercise, but underperformed the masked language model F1 score of 0.84 in Table 1, leaving this result ambiguous. Selecting a suitable prediction head dramatically affects model performance after fine-tuning: suboptimal head choices result in F1 scores of 52-93% of the scores achieved with the best prediction head.

The CLS prediction head performs well across all fields except `last_writeup`, where only 20% of all runs score above 0.25, and most score below 0.10. Classification is not a natural format for this field: in order to classify a passage, the model must learn 50 separate classes, one for each possible year from 1969–2019. CLS performs well when the number of classes is relatively low, especially when the answer is abstractive. However, it tends to fail to understand factual relationships. For example, when used for risk assessment its ratings correlate with the number of times the word “gang” or “murder” occurs in the passage (see Appendix D for more information).

The MLM head has nearly the opposite performance characteristics: it performs best on `last_writeup`, at an average level on `job_offer`, and very poorly on `edu_level`. It is telling that `last_writeup` can be expressed as a sentence with a single masked token (which may hold many values), whereas the classes of the latter are all concepts which do not fit into a single token. The MLM head's F1 scores tend to be several points lower than its accuracy, a symptom of the model occasionally filling the mask with arbitrary freeform values.

The QA heads perform well on `job_offer`, fairly well on `last_writeup`, and at an average level on `edu_level`. The first field is easily expressed as in the form of a yes/no question, and the second field's value is extractable from within the passage as with a regular QA task. However, the third requires the model to parse the passage to locate the answer, classify this into into one of four fixed phrasings, and return this phrasing from the prefix sentence, a task which is somewhat foreign to a question answering-based model.

5.4 Error Analysis

Errors fall into a few clear classes. Approximately 70% of all errors result from what appears to be the model learning spurious associations with co-occurring words. For example, in one conversational turn, a parole candidate describes both his own and the victim's level of education. The Producer incorrectly returns the victim's level of education, which uses the phrase "college courses."

Around 10% of errors result from complex passages (comparable to the examples in Appendix C), which continue to challenge language models. Spoken narrative language can be arbitrarily complex, and grounding in real world knowledge and presuppositions remain hard to encode. In one transcript, the commissioner asks, "Are you working towards a college degree?" which presupposes that the parole candidate completed high school. However, the model classifies this candidate as not having completed high school or a GED, as the transcript does not explicitly mention either. Some passages require numerical abilities which smaller language models tend to find difficult (Dua et al., 2019). Table 3 suggests that a larger language model may improve performance in many cases.

In the remaining 20% of errors, the Reducer failed to find a match for a given transcript or returned an incorrect passage.

Surprisingly, we found that in 15–50% of the total errors returned (varying by field), the model was actually correct, and had identified incorrectly-labeled or ambiguous data. To be conservative, we did not adjust F1 scores upwards and instead excluded the examples from this error analysis.

A detailed breakdown of errors for `edu_level` is provided in Appendix D for illustrative purposes.

6 Discussion

6.1 Combining Rules and Neural Models

Previous approaches to our problem use rule-generated labels to supervise a model. We instead split the problem into two, where the Reducer is entirely rule-based, and the Producer trains only on the few, but high quality, human labels.

Both rule-generated labels and a rule-based Reducer scale with the number of features to extract, but not the complexity of model or dataset. However, given a fixed development time, we find it more valuable for an SME to focus on only the Reducer. In contrast, end-to-end data programming requires rules for the Producer as well, which can be much more challenging to write. On our data, it takes about ten hours for an SME to write Reducer rules for a model that performs at the exceptional recall rates from Table 2. Hong et al. (2021) report the same number of hours per feature for an end-to-end data programming model, which performs much worse overall.

As future work, we hope to investigate whether a well-designed Reducer can improve human performance in creating gold-standard labels, saving time by reducing the need to read through entire transcripts.

6.2 Assessment of Human-in-the-Loop

We find that an hand-written rules can effectively isolate key segments of text in the overwhelming majority of situations.

The tradeoff of incurring the cost of writing rules per each additional feature proved to be very reasonable for our domain. We have few features, and our requirements demand accuracy over speed. In comparison, prior work suggests that for a neural model to achieve accuracy in the same ballpark, the model would require an order of magnitude more spans, which would be a prohibitive cost. In the general case, when applying our architecture, the per-feature cost of SME time should be considered against (a) the potential per-example savings from reducing labeling requirements, and (b) the performance requirements of the problem space.

The Human-in-the-Loop (HITL) approach enables SMEs to exert a positive influence on the quality of both the final model and the dataset. Given a probable baseline label error rate of a few percentage points (Alt et al., 2020; Reiss et al., 2020; Northcutt et al., 2021), as the Reducer's recall increases towards the 0.9 level, many of the mismatches against silver-standard Reducer evaluations and fine-tuning errors will actually be labeling errors. For example, in a case study where we checked `last_writeup` Reducer outputs against a silver-standard evaluation, we found that over 80% of "errors" were actually errors in human labeling. This also provides opportunities for SMEs to apply domain knowledge to more subtle classes of data issues, such as where Reducer rules surface mislabelings caused by labeler confusion.

As such, a unique advantage to HITL over a neural-only model is improving data quality during the training process. Purely neural models are forced to learn from mislabeled data points, which destabilizes benchmarks and damages model performance. (Northcutt et al., 2021) By comparison, we frequently detect label errors prior to fine-tuning, and as errors tend to occur in patches (such as under a particular labeler or a particular time period) we can quickly make corrections or exclude large bad patches from the training dataset. This can significantly increase training performance: excluding a patch of bad labels resulted in a 0.2 F1 improvement in one case. Appendix B elaborates on the data quality improvement process.

6.3 Modular Architecture

The Reducer-Producer architecture is useful for enabling iterative, componentwise development. Components may be improved in isolation as requirements arise, such as improving Reducer coverage or upgrading Producer language models, heads or prompts, and sometimes may be entirely replaced without any impact to their counterparts.

In particular, we hope to leave the door open for a general neural Reducer and Producer, allowing downstream users to perform open-ended querying and exploration of the dataset. This architecture enables future work to continue to use our Producer models, which are already trained. The information bottleneck between its components allows for rigorous measurement of the quality of Reducer output, which enables each component to be trained separately. Additionally, using present models to generate silver-standard data labels may alleviate issues of label scarcity.

7 Conclusion

Our corpus of parole hearings poses the challenge of information extraction with few gold labels: one thousand labels is not enough to locate and identify the answer in a long document. Parole, like many other applications, requires domain-specific knowledge, which raises the question of how best to incorporate the labor of subject matter experts to assist neural models in making optimal use of available labels, in order to achieve high performance on extraction tasks.

We identified two problems with existing work on the parole dataset, which fell short of the 0.80 F1 on many tasks: (1) Text segments remained too long for many SOTA neural models to digest, and contained many spurious signals. (2) Question answering was a useful first approach to handle a wide range of different feature types. However, out-of-the-box, it was rarely the best way to handle each individual feature type.

We present an approach that uses an SME-designed rule-based Reducer to identify relevant text segments, and a neural Producer to generate labels using those text segments.

We argue that it is time-efficient and performant for human SMEs to write mostly keyword-based rules for finding relevant parts of a parole transcript. In a parole transcript, a field of interest might be discussed in practically infinite different ways, but is usually somewhat well-defined by a limited set of words and patterns that are almost always used (for example, "GED", "college courses", "did not graduate" for a parole candidate's level of education). These keywords are relatively easy for a human to identify and write combinations of regular expressions to identify. However, training a neural model to recognize the phrases over the course of 20,000-word documents requires at least an order of magnitude more labels than are available (Hendrycks et al., 2021). Therefore, we focus SME energy on the Reducer, and *only* the Reducer.

For the Producer model, the role of human and machine are reversed. When the text is shortened to a sufficiently succinct context, neural models can be successfully fine-tuned to extract labels at an F1 of 0.80. It is practically impossible for a human to write rules to interpret every possible phrasing of, for example, someone's educational journey. However, pretrained language models excel at producing labels from small, targeted pieces of text. The 1,000 available labels are sufficient for good performance on this task (Zhang et al., 2020). We use a base model that can handle relatively long tokens. We also explore a range of different fine-tuning heads.

Our architecture shows the effectiveness of a modular, two-step approach, where not every module needs to be a neural or machine learning model. Such efforts to involve subject matter experts are especially important in applications that require substantial domain expertise. We hope that this work encourages additional research to better understand other legal processes whose workings are yet opaque to the public.

References

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020. Tacred revisited: A thorough evaluation of the tacred relation extraction task. *arXiv preprint arXiv:2004.14855*.

Kristen Bell, Jenny Hong, Nick McKeown, and Catalin Voss. 2021. The Recon Approach: A new direction for machine learning in criminal law. *Berkeley Technology Law Journal*, 37.

Danqi Chen. 2018. *Neural Reading Comprehension and Beyond*. Ph.D. thesis, Stanford University.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv:1710.10723*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhihao Dai, Zhong Li, and Lianghao Han. 2021. Bonebert: A bert-based automated information extraction system of radiology reports for bone fracture detection and diagnosis. In *IDA*, pages 263–274.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. *arXiv preprint arXiv:1905.05733*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Catherine D'ignazio and Lauren F Klein. 2020. *Data feminism*. MIT press.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. CUAD: An expert-annotated NLP dataset for legal contract review. *arXiv preprint arXiv:2103.06268*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Jenny Hong, Catalin Voss, and Christopher Manning. 2021. Challenges for information extraction from dialogue in criminal law. In *Proceedings of the 1st Workshop on NLP for Positive Impact*, pages 71–81, Online. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Tassilo Klein and Moin Nabi. 2019. Learning to answer by learning to ask: Getting the best of gpt-2 and bert worlds. *arXiv:1911.02365*.

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to progress in long-form question answering. *arXiv:2103.06332*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Albee Y Ling, Allison W Kurian, Jennifer L Caswell-Jin, George W Sledge Jr, Nigam H Shah, and Suzanne R Tamang. 2019. A semi-supervised machine learning approach to detecting recurrent metastatic breast cancer cases using linked cancer registry and electronic medical record data. *arXiv preprint arXiv:1901.05958*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ayush Maheshwari, Oishik Chatterjee, KrishnaTeja Killamsetty, Rishabh Iyer, and Ganesh Ramakrishnan. 2020. Data programming using semi-supervision and subset selection. *arXiv preprint arXiv:2008.09887*.

Curtis G Northcutt, Anish Athalye, and Jonas Mueller. 2021. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*.

Alex Ratner, Braden Hancock, Jared Dunnmon, Roger Goldman, and Christopher Ré. 2018. Snorkel metal: Weak supervision for multi-task learning. In *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*, pages 1–4.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, 11(3):269–282.

Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in Neural Information Processing Systems*, 29:3567.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Frederick Reiss, Hong Xu, Bryan Cutler, Karthik Muthuraman, and Zachary Eichenberger. 2020. Identifying incorrect labels in the conll-2003 corpus. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 215–226.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

Akshay Smit, Saahil Jain, Pranav Rajpurkar, Anuj Pareek, Andrew Y Ng, and Matthew P Lungren. 2020. Chexbert: combining automatic labelers and expert annotations for accurate radiology report labeling using bert. *arXiv preprint arXiv:2004.09167*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Peng Wang, Zhenkai Deng, and Ruilong Cui. 2021. Tdjee: A document-level joint model for financial event extraction. *Electronics*, 10(7):824.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Runxin Xu, Tianyu Liu, Lei Li, and Baobao Chang. 2021. Document-level event extraction via heterogeneous graph-based interaction model with a tracker. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3533–3546, Online. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Shanshan Zhang, Lihong He, Eduard Dragut, and Slobodan Vucetic. 2019. How to invest my time: Lessons from human-in-the-loop entity extraction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2305–2313.

Shun Zheng, Wei Cao, Wei Xu, and Jiang Bian. 2019. Doc2EDAG: An end-to-end document-level framework for Chinese financial event extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 337–346, Hong Kong, China. Association for Computational Linguistics.

A Reducer Operations and Rules

SMEs write Reducers for each field by composing pipelines of high-level operations, as described in Table 6. Operations run on an input transcript or a list of text segments, and emit matches which are compiled into a final output passage.

Extract Segments		
Extracts a list of segments from a raw transcript which match one or more regular expressions (regexes).		
	Input	Transcript text with any preprocessing.
	Regex	Accepts a list of regexes and searches the transcript separately for each item, returning matches in the same left-to-right order they are found.
	Limit	Length of segment returned around each match.
Filter & Split		
Filters a list of segments against two lists of regexes, to return two lists of matching and non-matching segments.		
	Regex	Accepts a "filter in" regex list which segments must match, and a "filter out" regex list which segments must not match.
Emit Matches		
Saves segments from a given list to a specified list for future compilation.		
	Limit	Length of segment to store around each match, and maximum segments to store.
Deduplicate		
Ensures a list of segments is free of duplicate or overlapping text ranges. Merges segments with partial overlaps.		
Compile Passage		
Merges a list of segments into a single text passage.		
	Separator	String inserted between each segment.
	Limit	Trims passage to a maximum length.

Table 6: Overview of Reducer operations.

To illustrate these operations in use, the pipeline for `job_offer` is provided in Table 7.

#	Param.	Value(s)
01	**Extract Segments**	
	Input	Transcript (lowercase)
	Regex	`job offer`
	Limit	1,000 chars centered on each match
02	**Filter & Split**	
	Input	Operation 01
	Regex	`letter`
03	**Emit Matches**	
	Input	Operation 02: Matches only
	Limit	2 segments
	Effect	*Emits 2x1,000-char segments which mention "job offer" in proximity to "letter".*
04	**Emit Matches**	
	Input	Operation 02: Non-matches only
	Limit	2 segments, 500 chars centered on each match
	Effect	*Emits 2x500-char segments which mention "job offer" but not "letter".*
05	**Extract Segments**	
	Input	Transcript (lowercase)
	Regexes	`jobs? ([\w,]+ ){2,10}offer OR offer\w+ ([\w,]+ ){2,10}job`
	Limit	500 chars centered on each match
06	**Emit Matches**	
	Input	Operation 05
	Limit	2 segments
	Effect	*Emits 2x500-char segments in which "job" and "offer" are within ten words of each other.*
07	**Extract Segments**	
	Input	Transcript (lowercase)
	Regex	`(?:find\w+|locat\w+|get\w+) (\w+ ){0,5}(?:work |employment|job(?! offer))`
	Limit	1,000 chars centered on each match
08	**Emit Matches**	
	Input	Operation 07
	Limit	2 segments
	Effect	*Emits 2x1,000-char segments in which a verb and a noun about job hunting are within five words of each other.*
09	**Extract Segments**	
	Input	Transcript (lowercase)
	Regex	`(?:job(?! offer)|employ|hire|work)`
	Limit	1,000 chars centered on each match
10	**Filter & Split**	
	Input	Operation 09
	Regexes	`letter AND offer`
11	**Emit Matches**	
	Input	Operation 10: Matches only
	Limit	2 segments
	Effect	*Emits 2x1,000-char segments which contain a word about employment in proximity to both "offer" or "letter".*
12	**Filter & Split**	
	Input	Operation 10: Non-matches only
	Regex	`letter`
13	**Emit Matches**	
	Input	Operation 12: Matches only
	Limit	2 segments
	Effect	*Emits 2x1,000-char segments which contain a word about employment in proximity to only "letter".*
14	**Emit Matches**	
	Input	Operation 12: Non-matches only
	Limit	5 segments
	Effect	*Catch-all: Emits 5x1,000-char segments which contain a word about employment.*
15	**Deduplicate**	
	Input	All emitted segments
16	**Compile Passage**	
	Input	Operation 15
	Separator	`[SEP]`
	Limit	First 6,500 characters

(Continued overleaf)

Table 7: Reducer pipeline for `job_offer`.

B Improving Data Quality using Silver-Standard Evaluations

Mismatches on silver-standard Reducer evaluations were often a product of real label errors: the datasets examined in Northcutt et al. (2021) had a 3.4% error rate on average, which is a similar order of magnitude to label errors encountered in our dataset when performing detailed manual verification.

The parole dataset includes records that span over more than a decade, and labeling has occurred in several waves over the years. As such, the semantic meanings of labels includes subtle shifts and inconsistencies. For example, a blank label might mean any one of the following:

- the annotator was uncertain,
- the transcript is unclear,
- the transcript is clear but the situation itself is ambiguous,
- "none" is a reasonable answer in this situation (such as `last_writeup` for a candidate with zero writeups),
- the feature was not applicable in this situation (such as `job_offer` for a candidate who is not working age); or
- the feature was simply not fully annotated.

To address these issues, we: (a) write code to correct issues where this is possible, (b) drop entire sections of low-quality train labels where patterns of errors exist, (c) hand-correct validation labels and keep track of all manual corrections, and (d) write small data transforms to simplify the job of the Producer (e.g., fixing common spelling and transcription errors).

C Sample Challenging Passages

Table 8 provides examples of the complex, challenging passages selected to benchmark language models in section 5.2, trimmed for brevity and redacted as per the conventions described within Figure 1.

D Supplemental Error Analysis: `edu_level`

This section provides a detailed breakdown of the error analysis for a single field and data split (`edu_level`, Validation), in order to illustrate typical patterns of errors encountered in our fine-tuned models.

This field was fine-tuned with a classification (CLS) prediction head, and correctly classified 89/106 of its labeled examples. Its 17 incorrectly-classified examples are examined in Table 9. The four possible values this field may hold are:

- NA: Did not finish high school
- HS: Completed high school or GED
- SC: Some college classes
- GC: Graduated from college

Field	Passages
risk_assess	**COMM:** With respect to violence risk assessment conclusions [...] the doctor uses a number of measurements. One is the PCL, which is the psychopathy checklist, and states that, "Overall score placed Mr. [REDACT] in the moderate range of psychopathy. [...]" Historically, on the HCR checklist, HCR20, the doctor writes, "[...] he has risk factors that place him in the low moderate risk range for future violence [...] The inmate's overall LS/CMI score indicates that he is in the medium category." And then the doctor goes on to discuss the historical domain and concludes, "[...] the inmate presents a moderate risk for future violence. [...] In the clinical or more current and dynamic domain of risk assessment [...] the inmate presents a moderate risk of future violence. As for the management of future risk domain [...] the inmate presents as a low risk of future violence. Overall then, risk assessment estimates suggests that the inmate poses a low moderate likelihood to become involved in a violent offense if released to the free community."
edu_level	**COMM:** Okay. So, and at the last hearing, it was discussed and I don't want to get - Well, that's parole plans. We're not going to talk about that right now. But, so you've taken a number of courses. It looks like in 2013, 2014, General Studies. Are you working towards a college degree? **CAND:** No. We're not able to take a college degree where I'm at. **COMM:** You say you've taken World War II, Europe Civilization, Ecology. Are these television courses or - **CAND:** They're videotapes, CDs.
job_offer	**COMM:** Do you have any job offers if you were to get a parole date? **CAND:** Uh, I used to be a mechanic before in, uh, [REDACT], my not in a company, but uh, in uh, a little shop with my friends. **COMM:** Do you have any job offers as a plumber? **CAND:** Yes. No, no, no, no, no, no. Not as a plumber. But, uh, I got, uh, as a mechanic I got offer with my cousin. **COMM:** Okay. Yeah. But he's in the United States, right? **CAND:** No, he's in [REDACT].
last_writeup	**COMM:** You've had 19 115s, starting in 1996, and most of these have been covered in prior hearings but, sort of running through them, couple in 1996, two in 1997, four in 1998, two in 1999, three in 2001, 2002, 2004, 2005, there was a pair. And then 2008, disobeying a direct order was your final 115. What was the 2005, knowingly providing a false claim?

Table 8: Examples of complex, challenging passages.

#	Source	Type	Label	Pred.	Error Details
1	Dataset	Ambiguous situation	HS	NA	Self-reported overseas high school completion (no records)
2	Dataset	Ambiguous situation	SC	HS	Vocational courses but taken at a college
3	Dataset	Mislabeling	HS	SC	Transcript explicitly discusses college courses taken
4	Reducer	Reducer pattern miss	NA	HS	Did not capture key sentence: *"I loved school. You know, I played the cello, you know, was ahead in school. I was graduating. I needed one credit to graduate from high school."*
5	Producer	Spurious associations	NA	HS	Two discussions about GED
6	Producer	Spurious associations	SC	HS	Confirms receipt of GED twice plus vocational training, just one brief mention of a college course
7	Producer	Spurious associations	NA	HS	Cluster of words: "school", "high school", "GED"
8	Producer	Spurious associations	SC	HS	Three mentions of graduating high school, one brief mention of college courses
9	Producer	Spurious associations	SC	GC	Candidate discusses the future receipt of an Associate's degree, later uses word "degree"
10	Producer	Spurious associations	NA	HS	Mentions "school" twice, "grade" three times
11	Producer	Spurious associations	SC	HS	Confirms receipt of GED twice, vocational training, two mentions of college courses
12	Producer	Spurious associations	SC	HS	Confirms receipt of GED twice, vocational training, mentions two colleges but not classes, units or degrees
13	Producer	Spurious associations	NA	HS	Mentions "school" three times, "college" twice
14	Producer	Spurious associations	GC	SC	Candidate confirms he has been doing college courses and is close to qualifying for an AA degree, but later notes he already has one degree
15	Producer	Spurious associations	SC	HS	Cluster of discussion around hgh school diploma, GED (four mentions) and reading scores
16	Producer	Spurious associations	SC	HS	Four separate mentions of having receiving GED, one small mention of college courses
17	Producer	Complex phrasing	NA	HS	Description is challenging to interpret: *"I started ditching school and hanging out when I was in high school. I think part of the reason for that was because we never had anything at home, everything was always, seemed like we're always struggling for everything, you know. Our electric bill, I didn't want to keep living like that, so I left, I left when I was 13 years old."*

Table 9: Example-level error assessments: `edu_level`.