iSeqL: Interactive Sequence Learning

Akshat Shrivastava
Paul G. Allen School of Computer Science & Engineering
University of Washington
akshats@cs.uw.edu

Jeffrey Heer
Paul G. Allen School of Computer Science & Engineering
University of Washington
jheer@uw.edu

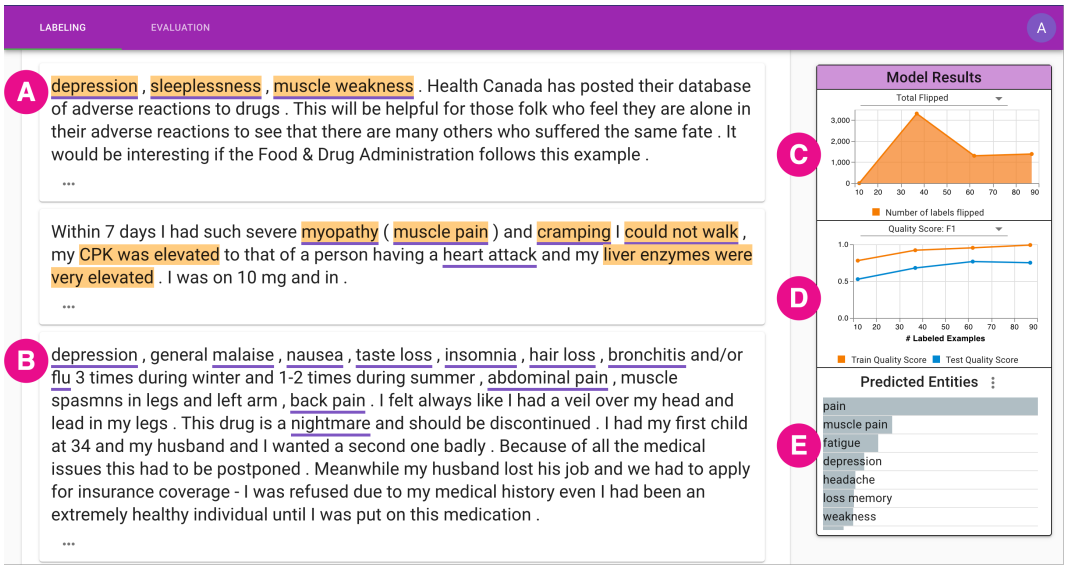

Figure 1: The iSeqL tool for interactive text sequence learning. A) Users label sampled instances; user-annotated entities are highlighted in yellow. B) Predictions from the current model are underlined to expedite annotation and convey model performance. The right-most panel contains evaluation aids: C) the count of labels that "flipped" in the last round, D) a model quality (F1) score against held out data; and E) an entity rank chart that shows the top predicted, labeled, and discovered entities.

ABSTRACT

Exploratory analysis of unstructured text is a difficult task, particularly when defining and extracting domain-specific concepts. We present *iSeqL*, an interactive tool for the rapid construction of customized text mining models through sequence labeling. With iSeqL, analysts engage in an active learning loop, labeling text instances and iteratively assessing trained models by viewing model predictions in the context of both individual text instances and task-specific visualizations of the full dataset. To build suitable models with limited training data, iSeqL leverages transfer learning and pre-trained contextual word embeddings within a recurrent neural architecture. Through case studies and an online experiment, we demonstrate the use of iSeqL to quickly bootstrap models sufficiently accurate to perform in-depth exploratory analysis. With less than an hour of annotation effort, iSeqL users are able to generate stable outputs over custom extracted entities, including context-sensitive discovery of phrases that were never manually labeled.

CCS CONCEPTS

• **Mathematics of computing** → **Exploratory data analysis**; • **Computing methodologies** → **Information extraction**; • **Human-centered computing** → **Interactive systems and tools**.

KEYWORDS

Interactive Machine Learning, Natural Language Processing, Exploratory Data Analysis

ACM Reference Format:
Akshat Shrivastava and Jeffrey Heer. 2020. iSeqL: Interactive Sequence Learning. In *25th International Conference on Intelligent User Interfaces (IUI '20), March 17–20, 2020, Cagliari, Italy.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3377325.3377503

IUI '20, March 17–20, 2020, Cagliari, Italy

ACM ISBN 978-1-4503-7118-6/20/03…$15.00
https://doi.org/10.1145/3377325.3377503

1 INTRODUCTION

Exploratory analysis involving unstructured text is an important and difficult task. Vast troves of unstructured text such as comments, reviews, news articles, and documents contain information lacking in available structured data. For example, pharmaceutical analysts

may want to analyze online drug reviews on a large scale to find out what reactions people are reporting, businesses may want to analyze reviews to see customer feedback, and analysts may want to analyze trends in news articles or scientific papers. Analysts may need to quickly bootstrap custom models for domain-specific concepts, which is difficult due to the domain expertise required and the complexity of Natural Language Processing (NLP) models. Analysts could benefit from interleaving model construction with visual analytics to explore the data and assess model performance.

Existing research mostly leverages model abstractions, ranging from simple word counts to complex statistical models, to understand raw text [8]. Commonly seen approaches struggle to balance three aspects, namely (a) the **semantics** captured, (b) the **effort** required to produce a model sufficiently accurate to initiate exploration, and (c) the **flexibility** to adjust and customize a model to capture idiosyncratic or domain-specific concepts. Models fit to massive training data can usually "read" unstructured text to extract information at different granularities (document, sentence, or phrase level), maximizing the expressiveness of the information we get from the text (a). However, pre-trained off-the-shelf models (e.g., for named entity recognition) only exist for a limited set of domains [3, 16, 24]. For other domains, analysts must build their own models: a task that may require massive human labeling effort, machine learning expertise, and significant computing resources, producing obstacles to both ease (b) and flexibility (c).

On-the-fly construction of dictionary models, commonly seen in the exploratory text analysis literature [10, 12, 26], enables users to iteratively apply their domain knowledge through dictionary construction. However, such methods treat all occurrences of a word as the same. This lack of context-sensitivity limits accuracy (a). Interactive machine learning methods seek to overcome these limitations by allowing users to customize their model; however, the main focus in the general interactive machine learning (IML) literature [18, 45] is to build accurate models first, often incurring too much overhead for rapid exploratory analysis (b).

We contribute *iSeqL* (pronounced "icicle"), an interactive machine learning tool for the rapid construction of sequence labeling models to aid exploratory data analysis. iSeqL operates at the word sequence level so that users can annotate at the desired level of granularity (c). iSeqL guides users to iteratively update their underlying models via active learning, while aiding *in situ* evaluation through integrated visualizations of performance measures, model predictions, and custom task-specific exploratory graphics (b). Importantly, we achieve more powerful models than in prior IML work by leveraging transfer learning within deep recurrent neural networks. iSeqL uses these architectures to learn semantic representations to discover new labels for the user (a). To overcome the latency induced by iteratively training these models — which may frustrate users by making them regularly pause and wait — we apply interleaved interaction and feedback strategies, such that model updates and manual labeling can proceed in parallel.

In an experimental benchmark study, we simulate active learning sessions to evaluate different model architectures and active learning heuristics. We identify those combinations capable of producing sufficiently accurate models given limited training data. Through both case study demonstrations and a controlled user study with 36 workers on Amazon Mechanical Turk, we show how iSeqL can be used to train a model and leverage it to visually explore a large text corpus with an hour's work. We find that with iSeqL novices are able to quickly build effective models for identifying reported adverse drug reactions. We also observe that *in situ* visualization of model predictions within iSeqL's labeling interface accelerated labeling without loss of accuracy. Across the subjects in our study, we find that F1 performance on a ground truth set is comparable to our simulated experiments. Through surveys and free text responses, we find that subjects were, on the whole, confident with their models and resulting outputs.

In summary, iSeqL contributes: (1) An IML system for sequence labeling that leverages pretrained embeddings and transfer learning to expedite interactive labeling; (2) An interface that interleaves model training with annotation in order to make efficient use of user's time, and provides visual aids to evaluate progress and enable insights into the data; and (3) An evaluation confirming that iSeqL users can rapidly train, assess, and apply models for visual analysis of text data. We find that showing the predictions of a current model expedites annotation without adding undue bias. iSeqL is available as open source software at www.github.com/AkshatSh/iSeqL.

2 RELATED WORK

2.1 Exploratory Analysis of Unstructured Text

To the best of our knowledge, we are the first to evaluate a combined interactive machine learning and exploratory data analysis system for *sequence* modeling. However, prior work has examined many other aspects of exploratory data analysis of unstructured text.

Jigsaw [44] is a tool to explore extracted entities using a variety of visualizations. While Jigsaw and iSeqL both analyze entities, a key difference is that Jigsaw uses existing classifiers for a predefined set of entity classes. iSeqL aims to let users define arbitrary classes and construct new deep learning models for their specific use case. Accordingly, iSeqL could serve as a sub-component that integrates with and augments systems such as Jigsaw.

TextTile [13] processes text to produce a list of general linguistic features, including tokenized unigrams, sentiment, part-of-speech tags, etc. Treated as additional "structured" metadata, these features enable rich filters on text snippets. iSeqL shares the objective of augmenting unstructured text with additional information. However, TextTile's predefined feature list is constrained; iSeqL supports flexible customization for sequence level analysis.

Dictionary-building approaches, such as ConceptVector [26], Empath [12], and TextFlow [10], allow users to analyze document concepts by leveraging a word embedding space to find related terms in order to build up dictionaries. While the resulting dictionaries are interpretable, they are also insensitive to context, leading to potential false positives (when word meaning shifts due to context) and an inability to "discover" new entities not in the dictionary at prediction time. In contrast, iSeqL can identify entities that were neither labeled nor observed during training.

Topic model visualizations such as LDAViz [41] and TopicCheck [9], allow users to examine multiple topics inside of texts and analyze their relation to different terms. However, underlying topic modeling algorithms such as LDA [4] generate coarse document-level semantic labels that require human interpretation and are often insufficient for fine-grained analysis [7, 8].

Other tools, such as OpinionSeer [50], support text analysis in a specific domain. For example, OpinionSeer targets customer opinions on products or services. While useful, it lacks the flexibility that we desire to switch to other domains and build custom models.

2.2 Interactive Machine Learning

We seek to build new sequence labeling models for arbitrary classes through *interactive machine learning* (IML). Related prior work includes the Explanatory Debugging [18] and NLPReViz [45] projects. These tools provide similar interactions as iSeqL — namely, selecting labels by highlighting text — but our goals are different. NLPReViz and Explanatory Debugging prioritize the creation of an accurate, personalized model for each individual user. We view model building as an iterative process *integrated* into visual analysis: a user quickly builds a model as an intermediate and iterative step within an exploratory analysis. In turn, the visualized model output (and how it changes as the model updates) aids validation and informs user choices of whether to engage in additional labeling effort.

Prior work has explored challenges with the current state of human-centered machine learning [36], namely *designing interaction for ML adaption* and *measuring quality and consistency*. By leveraging contextual embeddings iSeqL allows ML adaption simply through labeling interactions and utilizes visualized model outputs to allow quality and consistency evaluations.

In addition, our work addresses more complex models. Much prior IML work focuses on shallow models applicable to document-level classification, such as Naive Bayes and Support Vector Machines. iSeqL uses recurrent neural networks that consistently outperform shallow models for sequence labeling [19, 20, 31, 40, 49], leading to a similar interaction space for a different problem.

spaCy [16], an open source NLP package, includes a tool called Prodigy, which also uses an IML approach for sequence modeling. Prodigy aims to help ML engineers annotate data for model prototyping, and provides information about labeling progress and accuracy. iSeqL instead supports analysts, where the goal is exploratory data analysis. iSeqL demonstrates how visual summaries can be used to assess model performance on unlabeled data, and simultaneously explore data while evaluating the model.

Other methods, such as Snorkel [34], attempt more distant supervision in lieu of manual instance labeling, using *data programming* (writing labeling functions) to generate a noisy dataset to learn from. However, writing labeling functions requires programming expertise and often involves access to pre-existing taxonomies. With iSeqL we focus on accelerating model creation via manual labeling; future work might examine how to extend our integrated modeling and exploratory analysis loop to data programming approaches.

3 MODELING & ACTIVE LEARNING

The goal of iSeqL is to enable a single end user with limited machine learning or natural language processing experience to build sequence labeling models to explore text quickly. This goal naturally leads to our objective to minimize time while achieving sufficient accuracy. In this section we explore various modeling architectures and active learning settings to determine what works best in low-resource settings. Informed by prior text analysis work [18, 20, 22, 23, 43, 45] we concluded that promising approaches for accurate, context-sensitive, and data-efficient models include (1) *pre-trained contextual embeddings* to enable transfer learning from large, general language contexts to a specific labeling domain, and (2) *active learning methods* to select relevant instances for annotation.

We evaluate the use of transfer learning with pre-trained embeddings, along with active learning approaches. As we surveyed the sequence labeling literature (e.g., [21, 30, 49]), we noticed a strong trend in favor of the following architecture: *Embedding Layer* → *Recurrent Neural Network* (RNN) → *Conditional Random Field (CRF)*. Due to its popularity and superior performance, we evaluate this approach for iSeqL. We compare state-of-the-art named entity recognition models alongside dictionary classifiers. Through simulated experiments we show how models bootstrapped with ELMo [31] embeddings meet our design objectives, and find that no single active learning technique dominates overall.

3.1 Pre-trained Contextual Embeddings

Word embeddings map tokens in text to a numerical feature vector in a high-dimensional space. Unlike standard word embeddings produced by methods such as word2vec [25] and GloVe [28], *contextual embeddings* produce vector word representations based on surrounding context, such that a single word may be represented differently based on the linguistic context in which it is used [42]. Sequence labeling tasks, such as NER, have been shown to benefit from contextual embeddings [11, 29, 31, 35].

There are many contextual representation models, including ELMo [31], GPT [32], BERT [11], and GPT-2 [33]. In this work we use ELMo, though iSeqL can use other embedding models. In ELMo, words are represented as a function of the characters involved, which helps in identifying unseen words. Prior work [31] demonstrates that ELMo embeddings can be used to build models with less data and better performance for tasks such as semantic role labeling and natural language inference. We extend the experiments done in the ELMo paper, showing how it can be used to increase performance in sequence labeling tasks given limited training data.

However, the use of contextual embeddings can dramatically increase the resources needed to train models. This poses a challenge for interactive tools with strong latency requirements. In response, we precompute ELMo vectors. We first run the ELMo network over every instance in the dataset and store the resulting vectors. During training and evaluation, we retrieve the cached ELMo vector and apply the BiLSTM and CRF layers rather than run each sentence through the full ELMo BiLSTM CRF network. Appendix C presents experimental results on the effect of precomputation.

3.2 Active Learning

To reduce user labeling effort, we examine *active learning* methods to select unlabeled instances to annotate. Informed by the active learning literature [5, 39, 40, 43, 51], we selected three approaches:

Random Sampling: A baseline in which instances are sampled uniformly at random, with no active learning heuristics.

Uncertainty Based Methods: Sample instances about which the model is most "uncertain" in order to refine the decision boundary. We use the minimum Viterbi probability [46], which defines uncertainty as $P(y^*|x)$, where y^* is the most likely sequence and $P(y|x)$ is the Viterbi score of label sequence y on input x.

Embedding Space kNN: A k-Nearest Neighbors (kNN) approach, similar to the heuristic used in *CueFlick* [14]. We sample unlabeled instances that contain the most words that are closest to the positively labeled words. We measure word distance as the cosine similarity between two contextual embedding vectors.

3.3 Experiments

We present a series of experiments and a simulation of an active learning environment across three different datasets from different domains. We begin by benchmarking each model using a classical supervised learning setup. We then evaluate each model in an environment where random training samples are iteratively fed to the model, and report performance on a held-out test set. Finally, we use the best-performing model to assess active learning heuristics.

3.3.1 Experimental Models. We use benchmark experiments to assess two different aspects of our modeling approach. First, we compare dictionary classifiers against state of the art BiLSTM CRF models. We also compare neural BiLSTM CRF models with and without pre-trained embeddings, for a total of four model architectures (hyperparameters are listed in the Appendix, section B).

Word-Level Dictionary Classifier. The model is a dictionary for the entire dataset, storing counts for each word along with the class it belongs to. The model predicts classes at the word level: the tag for each word is predicted to be the most frequent tag for that word in the training data. The model is trained by counting the positive and negative labels for each word in the training set.

Phrase-Level Dictionary Classifier. The model is an entity-level dictionary classifier. Whereas the previous dictionary classifier operates at the word level, this approach stores a dictionary of positively defined entities, including multi-word phrases. At prediction time, if the incoming sentence contains any entities in the dictionary, the model marks those entities as positive labels. All positively labeled entities in the training set are stored in the dictionary.

Word-Level BiLSTM CRF. A state-of-the-art neural network for named entity recognition, without pre-trained embeddings. The model includes a bi-directional RNN, which then feeds into a CRF. The model is trained using stochastic gradient descent, starting from scratch without any pre-trained components.

ELMo BiLSTM CRF. A state-of-the-art neural network bootstrapped with ELMo embeddings. For improved computational efficiency, we freeze the ELMo embeddings as a fixed feature representation rather than fine-tune weights. Peters et al. [29] report a minimal accuracy difference between fine-tuning and fixed feature extraction. The model is trained with stochastic gradient descent.

3.3.2 Datasets. For each model type, we ran benchmark experiments on three datasets drawn from three different domains:

CONLL (News). The CONLL2003 task for named entity recognition [37] involves identifying entity types in news articles, such as Person (PER), Organization (ORG), and Location (LOC). We evaluate our models against the PER tag, to show how our system could be used to identify a series of unique names — a case where dictionary-based methods are unlikely to work well.

CADEC (User Reviews). The CSIRO Adverse Drug Event Corpus [17] is a dataset of user reviews about prescription drugs. The reviews are annotated with Adverse Drug Reaction (ADR), Symptom, Drug, Disease, and Finding. Here we evaluate against the ADR class. The ADR class is one of the most difficult to identify for two reasons. First, adverse reactions are heavily dependent on context and the text is from user source reviews. Second, patient-authored medical text is more prone to linguistic variation (including slang terms and misspellings) than paper abstracts and news articles [23].

Model	CONLL	CADEC	SCIERC
Word Level Dictionary	0.86	0.63	0.23
Phrase Level Dictionary	0.76	0.59	0.16
Word Level BiLSTM CRF	0.90	0.75	0.52
ELMo BiLSTM CRF	**0.99**	**0.80**	**0.64**

Table 1: Overall validation set F1 model performance across datasets, using all available training data.

SCIERC (Paper Abstracts). The SCIERC dataset [20] is a multi-task dataset, including named entity recognition of scientific terms in Artificial Intelligence paper abstracts. Example tasks include identifying Method, Task, and Term classes in a scientific paper. We evaluate our models against the Task tag. We use this dataset to evaluate our model in the computer science domain.

3.3.3 Evaluating Modeling Approaches. We first examine the maximum performance achieved by the models. We train the models on a dataset consisting of 80% training, 10% validation, and 10% test sets. Each model is trained for 15 epochs, and we retain the model with the highest validation set F1 performance. The results in Table 1 show that the ELMo model has the highest performance on all the datasets. Due in part to differences in context sensitivity, neural models strongly outperform the dictionary classifiers. Due to the large performance gaps, we did not spend a considerable amount of time tuning the models, and so our reported numbers may not be optimal. However our results echo prior results in which ELMo-based models outperformed others [20, 31, 52].

Next we assess model performance in a simulated interactive context in which the training dataset incrementally grows. We remove the phrase-level dictionary from consideration, as it was consistently worse than the word-level dictionary. We iteratively query an oracle (ground truth annotations) to label {1, 5, 10, 25, 50, 100, 200, 400} additional instances leading to dataset sizes of {1, 6, 16, 41, 91, 191, 391, 791}. We evaluate performance using pure random sampling of instances. The top row of Figure 2 summarizes the results, plotting F1 scores averaged over 10 separate training runs for each dataset size. In each domain, the ELMo BiLSTM CRF model dominates the other models at every dataset size.

To assess how different heuristics perform, we use an ELMo model within the same active learning simulation as above, but using our three instance selection heuristics (Random, Uncertainty, kNN). We run tests for each method using the same datasets as before. The results are summarized in the bottom row of Figure 2.

Our results indicate that an uncertainty based heuristic is capable of performing better than other active learning heuristics, but it is highly variable depending on the dataset and the number of training instances. For example, out of our 10 runs on the CONLL dataset, at dataset size 41 two of the trials produce a *zero* F1 score, yet the other eight runs outperform the other active learning heuristics (averaging 0.92 F1, while kNN averages 0.64 and random averages 0.73). Consistent with the literature [6], due to seeding effects,

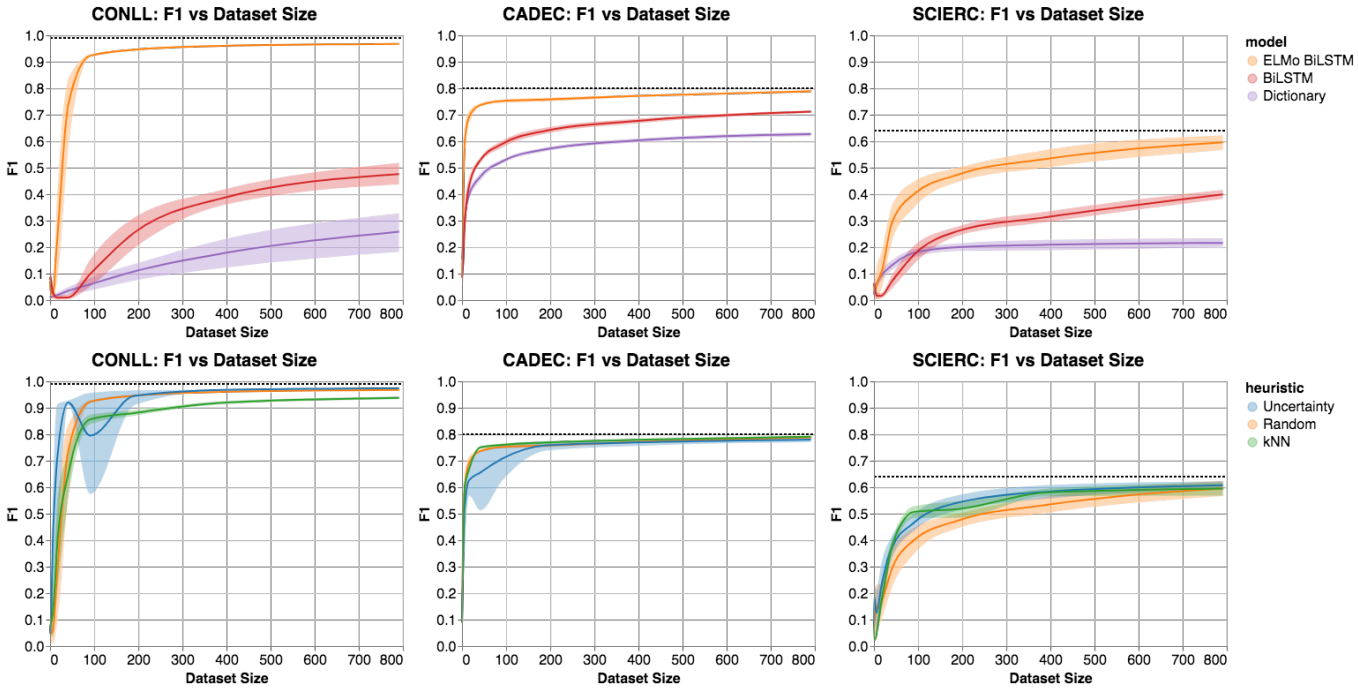

Figure 2: Benchmark model performance by training set size, showing the average F1 score and bootstrapped 95% CIs from 10 modeling runs. Dotted lines indicate the top performing model over the full training set. *Top row:* Performance by model type, with instances selected via random sampling. ELMo models approach maximum performance with a small number of instances. *Bottom row:* ELMo model performance using different active learning heuristics; no single technique dominates.

uncertainty sampling may not help when the dataset size is too low (size < 100). This variation is more likely to occur when there is a sparsity of positive labels, as in CONLL with the PER tag.

We conclude that the ELMo BiLSTM CRF model is the best for our use case, and active learning heuristics are of secondary importance. As a result, we leave the choice of active learning heuristic as a user-configurable parameter. Future work may explore hybrid techniques, such as first using random or kNN selection then switching to uncertainty sampling once a balanced set of labels has been collected [2]. In addition, our results confirm the limitations of dictionary-based approaches, despite their popularity [10, 12, 26].

4 SYSTEM DESIGN

We now present the iSeqL system and interface design. We first describe our methods for interleaving annotation and evaluation. We go on to discuss the user interface, including the annotation page and visualizations for assessing model performance.

The system architecture is shown in Figure 3, showing the relationships between components. Users begin by interacting with iSeqL to label a batch of instances. Once labeled, iSeqL stores the labels and sends them to a Model component for training. While the model trains, the Active Learning Manager evaluates which instances to present next. Once the model is finished training, the user can explore the model predictions on their dataset through both built-in and custom visualizations. All labels that a user provides are saved, and model predictions at each step are cached. This allows users to "step back" to earlier iterations of their model and compare how their visualizations have changed, shown in Appendix A.

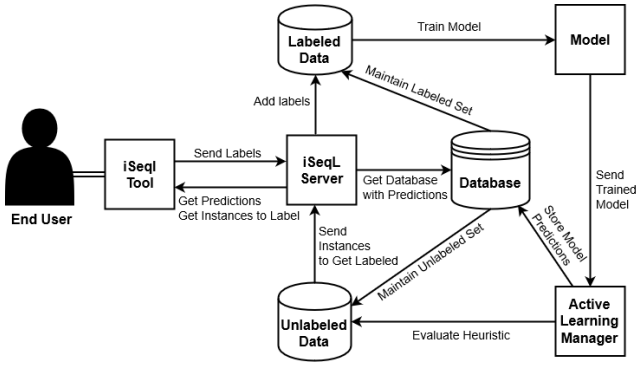

Figure 3: The iSeqL system architecture. The server tracks the active learning loop state for each user.

Our overarching goal is to help end users efficiently build an underlying model, so they can explore new unstructured text datasets within an hour. We had two corresponding system implementation goals: (**S1**) we want the time the user spends waiting for the system (T_s) to be much less than the user annotation time (T_u), denoted as $T_s << T_u$; and (**S2**) we want to support the active learning loop by interleaving annotation, training, and intermediate evaluation.

4.1 Effectively Utilizing Active Learning

In order to support the active learning loop, we present two key design considerations for active learning interactions in iSeqL: (1) batched active learning and (2) evaluation set selection.

4.1.1 Batched Active Learning. Effective active learning requires a balance between annotation and model fitting. We want to limit users' labeling effort and (re-)training time to enable rapid iteration (S1). Due to training and prediction costs, updating models per label introduces too much latency; instead, we apply batched active learning. To further utilize training time we institute "parallel processing" for user annotation and model fitting: as a user labels one batch, the system trains on the previous one (S2).

More formally, starting with an empty model m_0 with no training data, users label batches of inputs $b_i, i \in 0, ..., n$, where b_0 and b_1 are always randomly selected. When a batch has been labeled, the model is retrained and updated from m_{i-1} to m_i on the backend. Meanwhile, users are free to label a new batch b_{i+1} selected using model m_{i-1}. Given a labeling duration $L(b_i)$ and training time $T(b_i)$, we want to pick a batch size such that $L(b_i) \approx T(b_i)$, such that the user will always have something to do as the model trains. Through piloting we found that if we ask a user to label batch sizes of $|b| = 25$, labeling each batch takes roughly 10 minutes, and the server is able to train and evaluate a new model within 10 minutes. However, as our labeled dataset grows, the training pipeline will take longer, and the performance curve will start to plateau. To counteract this, we setup our system to double $|b|$ after every 5 iterations.

4.1.2 Evaluation Set. In classical supervised learning, a model is trained on a training set, tuned on a validation set, and evaluated on a test set. Each of these sets are drawn from similar distributions, e.g., with 80% of the data used for training, 10% for validation, and 10% for testing. To simplify our problem, we do not form a validation set. We justify this as our model hyperparameters are predetermined[1] and the user is not involved in tuning the model. In general, having a test set induces trade-offs, as the user annotation time could be used to provide more training instances to the model. We could ignore the test set and let the user trust the model we provide; however, the user will not know if the model overfits. We could ask the user to build a strong test set, but as user annotation time is limited, labeling 100 instances that do not directly improve the model may not be worthwhile.

Instead, iSeqL iteratively builds the test set alongside the train set. We select a parameter ϵ, such that when we ask the user to label b instances, $\epsilon \cdot |b|$ instances are used for the train set and $(1-\epsilon) \cdot |b|$ are used for the test set. This amortizes the cost of annotating a test set, and provides some statistics on how the model is performing on a hold out set. However, the user has to go through a few iterations before the test set has meaningful information. As a user progresses through iterations of the active learning loop, their test set becomes larger, and the model performance will stabilize. A user is not blocked by labeling a test set, but rather builds an initially noisy test set that gets more accurate over time.

4.2 User Interface Design

Traditional active learning systems rely on surfacing performance metrics on a large annotated test set [40, 43]. In contrast, iSeqL does not have a large test set and should limit the amount of time needed to build a sufficient model (i.e., within an hour), posing the following design challenges:

[1]We find a stable configuration for our hyperparameters through our experiments detailed in Appendix B.

C1. Assess model quality during labeling. In pilot tests of prototypes, users found it difficult to know when to switch between exploring model predictions and providing labels. Our current design integrates these two views so that users can simultaneously provide labels and assess model output.

C2. Know when to stop. Users need to know when they should stop labeling and focus on exploring model output. Error tolerances can vary over different users and different domains. iSeqL provides visualizations to aid judgments of when to stop.

4.2.1 Labeling Interactions. A core function of the iSeqL interface is to label entities within text instances. We constrain an entity to be a consecutive sequence of words. Users can annotate a span of text via drag-based mouse selection, similar to standard text highlighting. Since labels are at the word level, but selection is at the character level, we adjust the highlighting to automatically snap to the beginning word of the sequence and the end word of the sequence. The labeled sequences are highlighted with a yellow background (Figure 1 A). As users may make mistakes while selecting, iSeqL supports deselecting a span by clicking on it. We also include a button to clear all labels for an instance. We allow users to exclude an instance from training if they find it is too difficult to label or introduces irrelevant noise. In addition to manual labels, we show model predictions (C1). Predicted entities are underlined in purple to provide a strong contrast with manual labels. Our hypothesis is that showing predictions will help users label instances more quickly as well as perform *in situ* assessment of model performance.

4.2.2 Evaluation Side Panel. The iSeqL interface includes a side panel with visualizations of model performance and output, intended to help users gauge how well their model is performing. There are three components of the side panel (Figure 1 C, D, E):

Flipping Labels (Figure 1 C): This view depicts how model performance changes across iterations (C2) by depicting how many labels "flipped": spans that were previously identified as an entity but no longer, and vice versa. An area graph shows how many labels flipped from positive to negative (or vice versa) between batches. As a user progresses over training rounds, they should gain an understanding of how their model is changing over time, and estimate if another iteration might significantly improve their model. If the number of flipped labels approaches zero, it signifies the model has stopped learning.

Quality Score (Figure 1 D): This view presents standard model performance metrics. Traditionally in NLP, users will look at metrics (F1, precision, and recall) over a test and train set. In this view we display the chosen metric, F1, as a time series, so users can analyze how model performance is changing through their iterations on both the train set and the test set. This view helps users understand standard metrics for model performance, however only on labeled data, which is limited in most use cases. This helps with C1, however is likely to be noisy until a sufficiently large test set is labeled.

Entity Rank Charts (Figure 1 E): This view summarizes model predictions over the entire dataset, providing insight into model quality. Users can judge the correctness of the model (C1) based on their intuition of what the top entities should be, and assess convergence (C2) based on ranking stability. The list shows entities ranked according to the number of occurrences of that entity in the overall dataset. A bar chart shows the occurrence frequency of

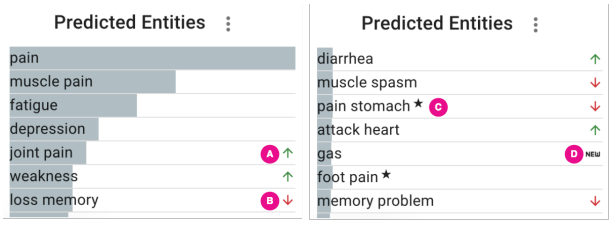

Figure 4: Entity Rank Chart showing predicted entities. A) entity increased in rank; B) entity decreased in rank; C) a discovered entity; D) a newly predicted entity.

predicted entities in the list. Green up-arrows and red down-arrows (Figure 4 A and B) show whether the item in the list went up or down in rank since the previous iteration. Seeing the rank change can help users understand whether the model is stabilizing and assess whether the model is updating properly. A star icon (Figure 4 C) indicates items that are marked as discovered (predicted but not labeled), and a "new" icon indicates entries that did not appear in previous iterations. This helps users identify explicitly what entities their latest iteration helped discover (Figure 4 D).

Using a drop-down menu, users can select among three different ranked lists: all predicted entities (the default), labeled entities only, and discovered (predicted but not labeled) entities only. Using these three lists, users can assess if the model is predicting the correct labels (predicted entities), generalizing properly (discovered entities), and see how it compares to their provided labels.

4.2.3 Exploratory Views. To explore their datasets, we give users two options. Users can examine their dataset at the entity level using pre-configured visualizations, including the entity rank list described above. Inspired by Jigsaw [44], iSeqL also provides a view to visualize the *relationships* between entities. We consider two predicted entities to co-occur if they are both predicted within the same instance. We visualize co-occurrence as an adjacency matrix (Figure 5 D) that shows how the top-k entities interact, where k is a user-configurable parameter.

To explore text data alongside the other metadata, iSeqL also allows users to provide custom Vega-Lite specifications [38]. iSeqL integrates input data tables (including various metadata fields) with entity predictions. This setup allows users to define their own views and use model outputs for exploratory data analysis. The case studies in §5.1 and Figure 5 (I, J, K) shows examples of this functionality. Going forward, iSeqL output could be integrated as a data source for visual analysis tools such as Voyager [47, 48].

5 EVALUATION

We evaluated iSeqL in three different ways. Earlier (§3.3.3) we assessed our modeling choices through simulated active learning runs. Here, we evaluate the utility of iSeqL through case studies and present an analysis of model building capabilities through a controlled online experiment.

5.1 Case Studies

We demonstrate iSeqL through two case studies conducted by the first author of this paper. We assess how iSeqL can be used to reveal new insights about unstructured text in two different domains: patient-authored posts about drug side-effects and Yelp reviews.

5.1.1 Case Study 1: Adverse Drug Reactions. In this case study we examine the CADEC dataset [17]. We chose CADEC because examining drug reviews is an interesting topic and relevant to sequence modeling. However, as CADEC was annotated by medical professionals, we simplify the labeling criteria, by using the shortest sequence that could represent a reaction; e.g., in "severe muscle pain in arms" only "muscle pain" gets annotated. We use iSeqL to analyze patient reports and label adverse drug reactions. We seek to answer two questions: *What are the most frequently occurring adverse reactions? How do adverse reactions co-occur?*

We initially labeled 75 instances, at which point we start to see that our model is performing as desired. Using the visual summaries presented in the side panel (Figure 5 A, B, C), we observe that the number of labels that flip between batches starts to decrease, our predicted entity list has stopped changing, and our model performance metrics stabilize. In addition, while labeling the subsequent batch we see that our model is predicting reactions well (Figure 5, E). Through these visualizations, we get a sense that our model is training properly and is ready to be used for exploratory analysis.

To answer our original question we look at two charts: the predicted entity and entity co-occurrence views. The predicted entity view (Figure 5 C) indicates that the top 3 adverse reactions occurring in our drug reviews are "pain", "muscle pain", and "fatigue". Our second question is to investigate how these entities co-occur. We examine an entity co-occurrence matrix (Figure 5 D). The first column of the matrix shows how "pain" occurs with other entities. We see that "weakness", "fatigue", and "muscle pain" are often mentioned alongside general reports of "pain". Meanwhile, "tiredness" and "dizziness" occur relatively less frequently with "pain".

5.1.2 Case Study 2: Yelp Reviews. Next, we show how the entities extracted by iSeqL can be aggregated to visualize document-level scores. We also show how iSeqL can be used alongside other information (namely sentiment analysis) to explore more complicated questions. We examine a Yelp! reviews dataset [1] containing 7 million reviews for businesses around the world. In addition to text reviews, the dataset contains extensive metadata (e.g., location, user ratings, category, *etc.*). We demonstrate how iSeqL can be used in conjunction with this data to gather information about restaurant service. We investigate a subset of this data by randomly sampling 1,000 reviews for U.S. restaurants and investigate the question: *How does restaurant service quality affect restaurant ratings?*

We start by defining our sequence labeling problem. We want to identify "service entities," a word or phrase concerning wait staff service. To answer our question, we use iSeqL along with an off-the-shelf sentiment analysis tool (from NLTK [3]) to define a service quality score. We calculate average per-sentence sentiment score of sentences containing iSeqL-tagged phrases related to restaurant service. This score is formally defined in Equation 2:

$$N_{service} = \sum_{i=0}^{N} is\_service(\mathrm{r}_i) \tag{1}$$

$$score = \frac{1}{N_{service}} \sum_{i=0}^{N} sentiment(\mathrm{r}_i) \cdot is\_service(\mathrm{r}_i) \tag{2}$$

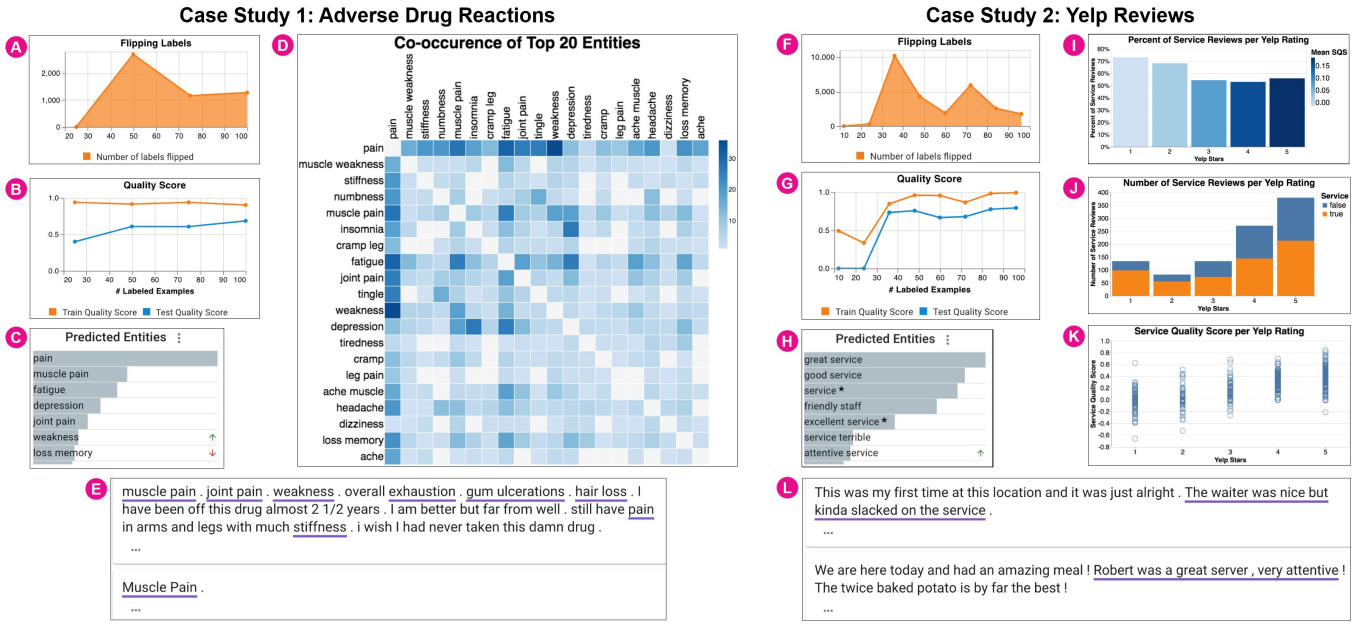

Figure 5: Summary visualizations of case study results. *ADR Case Study (left)*: In the side panel: A) labels are flipping less, B) the quality (F1) scores are stabilizing, and C) the top entities have stopped changing rank. D) An adjacency matrix shows the co-occurrence of adverse drug reactions. E) An example drug review (not yet labeled) with model predictions for adverse reaction entities underlined in purple. Reading these predictions we can more qualitatively see the model's behavior. *Yelp Case Study (right)*: F) label flips have reduced, G) quality scores are stabilizing, and H) the top entities have not changed rank and seem relevant. I) The percentage of reviews that talk about service (predicted using iSeqL), grouped by Yelp ratings (stars). The color scale encodes the average service quality score (SQS). J) Reviews per Yelp star, colors encode reviews that iSeqL predicts discuss service (orange) versus those that do not (blue). K) Service scores computed using iSeqL predictions, grouped by Yelp review ratings (stars). L) Unlabeled example reviews, with model predictions underlined.

where r_i is the ith sentence in the review. The *sentiment* function returns the sentiment score of the sentence, ranging from −1 (negative sentiment) to +1 (positive), with 0 as neutral. The function *is_service* returns 1 if the sentence is about service and 0 if not, based on whether iSeqL predicts the presence of a service entity.

We began labeling in iSeqL. After labeling 100 instances we moved on to further analyze model results: Figure 5 (F, G, H) shows that the statistics stabilized and Figure 5 (L) shows that the displayed model predictions appear to be correct. Figure 5 (K) compares the service quality score to the review ratings for the restaurant. We see that, as one might expect, the service score correlates with star ratings. Looking at Figure 5 (I), we can get a richer sense of how service scores might contribute to a Yelp! user's star rating judgment. The bar heights show that around 73% of the reviews that give 1 star to restaurants mention service, whereas only 53% of reviews that give 4 stars mention service: service is a more prominent concern within negative reviews.

However, we also notice that the top predicted entities are positive terms (Figure 5 F, G, H). Figure 5 (J) shows that our sample contains more 4-and-5-star reviews. While there are more service-related reviews that are positive, the relative percentage is lower because many positive reviews do not mention service.

5.2 Online Experiment

To evaluate our design decisions we conducted a user study on Amazon Mechanical Turk (MTurk). We sought to characterize novice users' timing and resulting model performance, and assess the effect of showing model predictions within the labeling interface. We hypothesized that, (**H1**) by guiding user attention to potentially relevant words, showing model predictions would result in faster annotation times. However, showing predictions might bias users, perhaps causing them to overlook relevant spans that lack predictions. Nevertheless, as users still must read through the text to ensure appropriate recall, we hypothesized that (**H2**) models in both conditions (with and without predictions shown) would exhibit similar performance in terms of F1, precision, and recall scores. We tested these conditions in a between-subjects design.

5.2.1 Protocol. Following our case study in §5.1.1, we asked participants to build a model to identify adverse drug reactions in the CADEC dataset (§3.3.2). To make our task more accessible to a general public, we altered the definition of "Adverse Reaction" to be any reaction that is an unintended consequence of taking the drug. Our study started with instructions explaining the task, related technical terms, and the iSeqL interface. We used three quiz questions to verify that participants understood the task. They could not progress until they had correctly answered the questions.

Participants completed five sequential batches of annotation in iSeqL, labeling 25 instances per iteration. A model training run

commenced on the server whenever a labeled batch was submitted. The interface notified the user when the training was complete and provided a button to update the model evaluation visualizations. To enable proper comparison, for each batch all subjects labeled the same instances, presented in the same order.

We tested two conditions using a between-subjects design. In the *show predictions* condition, participants were shown model predictions within the annotation interface (Figure 1). In the *no predictions* condition, participants did not see any machine-produced labels in the annotation interface. Both groups did not see any model predictions during the first labeling batch, but the *show predictions* group would see predictions once their first model had been trained. Both groups had access to the summary visualizations showing label "flips" between batches, per-batch validation set F1 scores, and a ranked list of top entities (Figure 1 C, D, E).

At the end of the task, the subjects completed an exit survey about the model they had built and their confidence in the results. We asked subjects a series of 4 questions:

Top 3 Entities: As a compliance check, we asked subjects to identify the top 3 predicted entities that were shown in the side panel chart in Figure 1E. A subject was not allowed to submit the survey until they had identified these entities correctly.

Entity Confidence: Subjects responded to the statement "The top 3 predicted entities are correctly identified." Responses are on a 5-point Likert scale from "Strongly Disagree" to "Strongly Agree".

Prediction Confidence: Subjects responded to the statement "The model is correctly able to identify the adverse drug reactions in the drug reviews." The responses are on a 5-point Likert scale from "Almost Never True" to "Almost Always True".

Stopping Criteria: Subjects responded to the statement "In the bottom-right panel there is a predicted entities list. How would another round of labeling would improve the contents or the ranking of the top 10 entities in the predicted entities list?" Responses are on a 5-point Likert scale from "All of them would change" to "None of them or one of them would change". The intent was to gauge when users might decide to stop training their model.

Lastly, participants were encouraged to leave free-text comments and feedback on their experiences using iSeqL.

5.2.2 Participants. We recruited 40 subjects on Mechanical Turk. We restricted our participants to be in the United States, and have an approval rating above 97%. Each participant was compensated $12 USD. We did not ask the participants for their age or gender.

5.2.3 Results. We analyzed the collected experimental data in terms of both per-batch labeling times and output model performance. To analyze labeling times for batches 2 through 5 (recall that no subjects saw predictions for batch 1), we use linear mixed effects models of the log-transformed response time, with *show condition* as a fixed effect and random intercept terms for both user and batch. The random effects are included to account for per-subject variance and the different contents of each labeling batch (e.g., with potentially varied text lengths, ADR occurrence frequency, *etc.*). To assess significance of final model performance metrics, we use the non-parametric Wilcoxon rank-sum test.

Prior to analysis, we visualized raw responses to assess data quality. We found four subjects who produced results unlike the others, involving either very low final F1 scores (likely indicating confusion, e.g. less than 0.35) or wildly varying batch completion times (with batches completed too quickly for a full annotation effort, e.g. less than 10 min). We dropped data for these four participants, and analyzed data from 36 subjects, 18 in each condition.

Showing predictions accelerates annotation, supporting H1. Showing model predictions within the annotation interface significantly reduced average labeling time ($\chi^2(1) = 4.344, p = 0.037$). Our model predicts that the average time per batch is 1.01 to 1.44 minutes faster (95% CI, transformed from log domain) when predictions are shown. The total experiment took a median 66.9 minutes in the *show predictions* condition, versus a median 81.0 minutes in the *no predictions* condition — a difference of 14 minutes. The results support our hypothesis (H1) that showing predictions accelerates annotation, thus reducing labeling effort.

Showing predictions does not effect performance, supporting H2. We found no significant effect of showing model predictions on F1 ($W = 185, p = 0.481$), precision ($W = 178.5, p = 0.613$), or recall ($W = 189, p = 0.584$) measured against a test set of 100 instances labeled by the first author. The results show no evidence that showing predictions biases the labeling results (H2). That said, we do observe slightly better median performance for the *no predictions* case (F1 = 0.71 vs. 0.69, precision = 0.62 vs 0.62, recall = 0.83 vs 0.80). However, these differences are not significant.

Subjects create "good enough" models. Across subjects, the median performance was F1 = 0.70 (IQR [0.67, 0.71]), precision = 0.62 (IQR [0.59, 0.67]), and recall = 0.82 (IQR [0.71, 0.86]). The higher recall rate indicates a preference for false positives over false negatives, arguably a desirable priority when analyzing potential health risks. Though not directly comparable due to our modified definition of ADR, the overall F1 of 0.70 echoes model performance trained on 100 ground-truth labeled instances in Figure 2.

We can also examine the top-k ADR rankings induced by each participants' model. Figure 6 plots the top ADRs (in terms of average frequency across subject models) versus the rank indices of that entity within each model. We can see that some models may "overlook" an entity, but that overall there is a strong correlation among the produced rankings. Notable discrepancies appear to be due to differences in subjects' strategies for labeling general words versus more specific phrases; for example, "pain" versus "muscle pain", "cramp" versus "leg cramp", and "ache" versus "muscle ache". These results also illustrate the contextual nature of iSeqL predictions: had we used a dictionary classifier, there would be no variance in the counts for entities across multiple dictionaries.

The sequence model's ability to discover new terms also enriches the collected entities. Looking at the top 100 entities for each model across our user study, 189 unique entities (out of 334 total unique entities) were "discovered" by at least one iSeqL model. These are terms that, for at least one user's model, were never annotated within a model's training data and yet were still positively predicted. Of those, 94 entities – roughly half – were never annotated by *any* of the subjects. For example, the span "brain fog" was never labeled by anyone, yet was discovered by 30/36 user models (83%). Figure 7 shows the predicted counts of entities that were never annotated, yet discovered by at least half of the participants' models.

Exit survey results. Exit survey results are presented in Figure 8. For each Likert scale question, a rating of 1 implies low

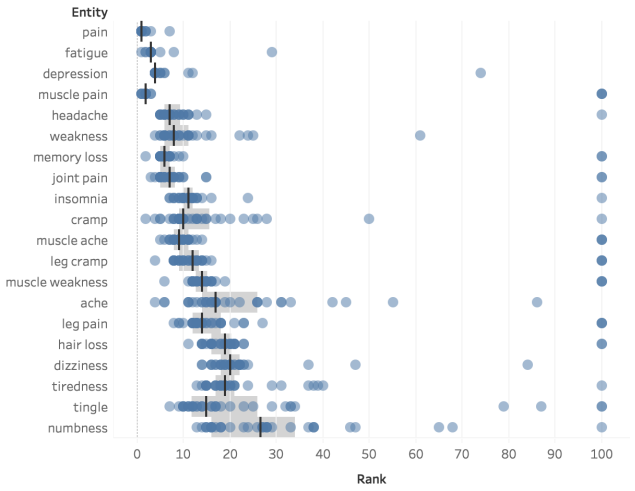

Figure 6: Entity ranks for subjects' final models, annotated with the median rank and interquartile range. The top 20 ADR entities (by average occurrence frequency per model) are shown. Entities are given a maximum rank of 100 if they do not appear in the top 100 produced by a model.

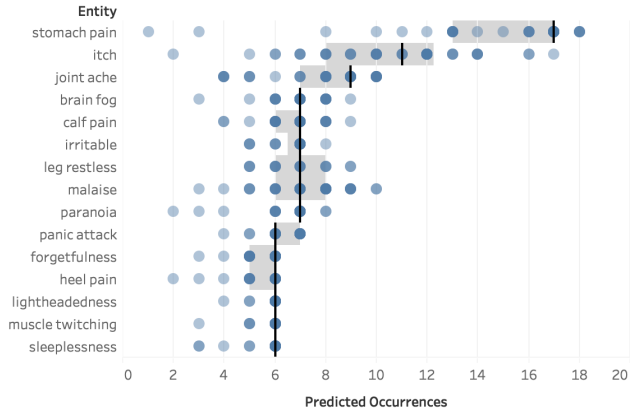

Figure 7: Entities that were not annotated, but predicted by 50%+ of subjects' models. The plot shows prediction counts per model, with median and interquartile range.

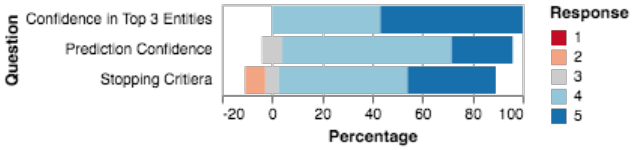

Figure 8: Exit survey results, visualized on a diverging scale. Generally participants tend to trust their model predictions.

confidence in the user's model, and a rating of 5 implies high confidence. We found no significant difference between prediction conditions in terms of entity confidence ($\mu = 4.56$, $\sigma = 0.50$; $W = 180$, $p = 0.520$), prediction confidence ($\mu = 4.17$, $\sigma = 0.56$; $W = 196.5$, $p = 0.195$), or stopping criteria ($\mu = 4.14$, $\sigma = 0.87$; $W = 170.5$, $p = 0.781$). The averages show that the users have a high confidence in the top few entities being correct and agree that the model is performing reasonably. While the average score for the stopping criteria is near the other Likert scale averages, the standard deviation is more than 50% higher. These results suggest that, though iSeqL provides insight into whether another labeling batch is needed, subjects exhibit uncertainty for when "enough is enough."

6 DISCUSSION AND FUTURE WORK

We presented iSeqL, an interactive machine learning system for sequence labeling to aid exploratory analysis of unstructured text. Users are able to simultaneously evaluate and explore their data to gain initial insights about unstructured text. We presented two case studies that demonstrate how iSeqL can be used to view trends at the entity level and enable exploratory visualization in combination with other metadata fields of a dataset. In addition, we conducted a user study on Mechanical Turk to assess overall performance and evaluate the automatic inclusion of machine predictions within iSeqL's labeling interface. We found that online participants are able to build sequence labeling models for identifying drug reactions in about an hour. We demonstrate productivity gains, with no evidence found for loss of performance, when including predictions and confirm iSeqL's ability to help users "discover" new entities. In short, iSeqL can help users build models for complex scenarios quickly without expert knowledge of how the models work.

There are a few limitations of iSeqL that lend themselves to future work. One limitation is that iSeqL processes large training and evaluation jobs on the entire dataset and sends predictions over the network to clients to visualize, potentially causing latency issues. Future system optimizations are needed for iSeqL to support very large datasets with over a million sentences.

In our case studies, our models identified multiple entities that refer to similar symptoms: *e.g.*, "leg pain", "leg hurts", and "discomfort in leg". We attempted some simple transforms to help group entities together, using the lemma of each word and removing stop words to form extracted entities. While helpful, this transform does not adequately solve the entity resolution problem, which is a compelling interactive machine learning challenge in its own right. Future iterations of iSeqL could explore methods for interactive grouping that leverage the contextual embedding space, hierarchical structure, and/or a knowledge base to provide additional insight.

Finally, further empirical work is needed to better understand the labeling and exploration processes. We conducted our user study online through Amazon Mechanical Turk. While this choice gives us access to a diverse participant pool (at least, relative to college campuses), we were unable to observe participants as they worked with the tool. A think-aloud study could provide additional insights into subjects' experiences and the current limitations of the system.

To support these and other future research questions, iSeqL is available as open source software at www.github.com/AkshatSh/iSeqL. We hope that iSeqL will provide a valuable building block for continuing work on interactive machine learning for visual analytics.

ACKNOWLEDGMENTS

The project was supported by the Moore Foundation Data-Driven Discovery Investigator program. We gratefully thank Tongshuang Wu, Matthew Conlen, Dominik Moritz, Jane Hoffswell, Younghoon Kim, Halden Lin, and Yang Liu for their helpful comments.

REFERENCES

[1] [n. d.]. Yelp Dataset. https://www.yelp.com/dataset/challenge. Accessed: 2018.

[2] Jürgen Bernard, Matthias Zeppelzauer, Markus Lehmann, Martin Müller, and Michael Sedlmair. 2018. Towards User-Centered Active Learning Algorithms. *Comput. Graph. Forum* 37 (2018), 121–132.

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

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (March 2003), 993–1022. http://dl.acm.org/citation.cfm?id=944919.944937

[5] Yukun Chen, Thomas A. Lask, Qiaozhu Mei, Qingxia Chen, Sungrim Moon, Jingqi Wang, Ky Nguyen, Tolulola Dawodu, Trevor Cohen, Joshua C. Denny, and Hua Xu. 2017. An active learning-enabled annotation system for clinical named entity recognition. *BMC Medical Informatics and Decision Making* 17, 2 (05 Jul 2017), 82. https://doi.org/10.1186/s12911-017-0466-9

[6] Yunxia Chen and S. Mani. 2011. Active Learning for Unbalanced Data in the Challenge with Multiple Models and Biasing. In *Active Learning and Experimental Design @ AISTATS*.

[7] Jason Chuang, Sonal Gupta, Christopher Manning, and Jeffrey Heer. 2013. Topic model diagnostics: Assessing domain relevance via topical alignment. In *International Conference on Machine Learning*. 612–620.

[8] Jason Chuang, Daniel Ramage, Christopher Manning, and Jeffrey Heer. 2012. Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 443–452.

[9] Jason Chuang, Margaret E Roberts, Brandon M Stewart, Rebecca Weiss, Dustin Tingley, Justin Grimmer, and Jeffrey Heer. 2015. TopicCheck: Interactive alignment for assessing topic model stability. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 175–184.

[10] Weiwei Cui, Mengchen Liu, Lizhe Tan, Conglei Shi, Yangqiu Song, Zekai Gao, Huamin Qu, and Xin Tong. 2011. TextFlow: Towards Better Understanding of Evolving Topics in Text. *IEEE Transactions on Visualization and Computer Graphics* 17 (2011), 2412–2421.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018).

[12] Ethan Fast, Binbin Chen, and Michael S. Bernstein. 2016. Empath: Understanding Topic Signals in Large-Scale Text. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 4647–4657. https://doi.org/10.1145/2858036.2858535

[13] Cristian Felix, Anshul Vikram Pandey, and Enrico Bertini. 2017. TextTile: An Interactive Visualization Tool for Seamless Exploratory Analysis of Structured Data and Unstructured Text. *IEEE Transactions on Visualization and Computer Graphics* 23 (2017), 161–170.

[14] James Fogarty, Desney S. Tan, Ashish Kapoor, and Simon A. J. Winder. 2008. CueFlik: interactive concept learning in image search. In *CHI*.

[15] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A Deep Semantic Natural Language Processing Platform. arXiv:arXiv:1803.07640

[16] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear* (2017).

[17] Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. Cadec: A corpus of adverse drug event annotations. *Journal of biomedical informatics* 55 (2015), 73–81.

[18] Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of Explanatory Debugging to Personalize Interactive Machine Learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI '15)*. ACM, New York, NY, USA, 126–137. https://doi.org/10.1145/2678025.2701399

[19] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *HLT-NAACL*.

[20] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction. In *EMNLP*.

[21] Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017. Scientific Information Extraction with Semi-supervised Neural Tagging. *CoRR* abs/1708.06075 (2017). arXiv:1708.06075 http://arxiv.org/abs/1708.06075

[22] Diana MacLean, Sonal Gupta, Anna Lembke, Christopher D. Manning, and Jeffrey Heer. 2015. Forum77: An Analysis of an Online Health Forum Dedicated to Addiction Recovery. In *ACM Computer-Supported Cooperative Work (CSCW)*. http://idl.cs.washington.edu/papers/forum77

[23] Diana L. MacLean and Jeffrey Heer. 2013. Identifying medical terms in patient-authored text: a crowdsourcing-based approach. In *JAMIA*.

[24] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*.

[26] Deok Gun Park, Seungyeon Kim, Jurim Lee, Jaegul Choo, Nicholas Diakopoulos, and Niklas Elmqvist. 2018. ConceptVector: Text Visual Analytics via Interactive Lexicon Building Using Word Embedding. *IEEE Transactions on Visualization and Computer Graphics* 24 (2018), 361–370.

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

[28] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[29] Matthew Peters, Sebastian Ruder, and Noah A. Smith. 2019. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks.

[30] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *CoRR* abs/1705.00108 (2017). arXiv:1705.00108 http://arxiv.org/abs/1705.00108

[31] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

[32] Alec Radford. 2018. Improving Language Understanding by Generative Pre-Training.

[33] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[34] Alexander J. Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan 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 (2017), 269–282.

[35] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer Learning in Natural Language Processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*. 15–18.

[36] Dominik Sacha, Michael Sedlmair, Leishi Zhang, John Aldo Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C. North, and Daniel A. Keim. 2017. What you see is what you can change: Human-centered machine learning by interactive visualization. *Neurocomputing* 268 (2017), 164–175.

[37] Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *CoNLL*.

[38] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2017). http://idl.cs.washington.edu/papers/vega-lite

[39] Burr Settles and Mark Craven. 2008. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1070–1079. http://dl.acm.org/citation.cfm?id=1613715.1613855

[40] Yanyao Shen, Hyokun Yun, Zachary Chase Lipton, Yakov Kronrod, and Anima Anandkumar. 2017. Deep Active Learning for Named Entity Recognition. In *Rep4NLP@ACL*.

[41] Carson Sievert. 2014. LDAvis: A method for visualizing and interpreting topics.

[42] Noah A. Smith. 2019. Contextual Word Representations: A Contextual Introduction. *CoRR* abs/1902.06006 (2019).

[43] Gabriel Stanovsky, Daniel Gruhl, and Pablo Mendes. 2017. Recognizing Mentions of Adverse Drug Reaction in Social Media Using Knowledge-Infused Recurrent Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, 142–151. http://aclweb.org/anthology/E17-1014

[44] John T. Stasko, Carsten Görg, Zhicheng Liu, and Kanupriya Singhal. 2007. Jigsaw: Supporting Investigative Analysis through Interactive Visualization. *2007 IEEE Symposium on Visual Analytics Science and Technology* (2007), 131–138.

[45] Gaurav Trivedi, Phuong Pham, Wendy W. Chapman, Rebecca Hwa, Janyce Wiebe, and Harry Hochheiser. 2018. NLPReViz: an interactive tool for natural language processing on clinical text. *Journal of the American Medical Informatics Association : JAMIA* 25 1 (2018), 81–87.

[46] Dittaya Wanvarie, Hiroya Takamura, and Manabu Okumura. 2011. Active learning with subsequence sampling strategy for sequence labeling tasks. *Information and Media Technologies* 6, 3 (2011), 680–700.

[47] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics* 22, 1 (2016), 649–658.

[48] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2648–2659.

[49] Yonghui Wu, Min Jiang, Jun Xu, Degui Zhi, and Hua Xu. 2017. Clinical Named Entity Recognition Using Deep Learning Models. *AMIA ... Annual Symposium proceedings. AMIA Symposium* 2017 (2017), 1812–1819.

[50] Yingcai Wu, Furu Wei, Mengchen Liu, Norman Au, Weiwei Cui, Hong Zhou, and Huamin Qu. 2010. OpinionSeer: Interactive Visualization of Hotel Customer Feedback. *IEEE Transactions on Visualization and Computer Graphics* 16 (2010), 1109–1118.

[51] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. OpenTag: Open Attribute Value Extraction from Product Profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. ACM, New York, NY, USA, 1049–1058. https://doi.org/10.1145/3219819.3219839

[52] Henghui Zhu, Ioannis Ch. Paschalidis, and Amir M Tahmasebi. 2018. Clinical Concept Extraction with Contextual Word Embedding. *CoRR* abs/1810.10566 (2018).

APPENDIX FOR ISEQL IUI 2020 SUBMISSION

A TIME STEPPING VIEWS

As users iterate through exploration and model building, they're visualizations will change less as the model stabilizes and reaches it optimal performance. For users to understand this change, we allow users to step to previous iteration of models and their respective predictions, to see how their visualizations and stats have changed. Consider figure 9. Here we can see how our visualization changes over iterations. In early iterations we can see drastic changes as our model learns our task, however in later iterations we see our visualizations changes stabilizing, signifying that the model may be done learning.

B MODEL HYPERPARAMETERS

In §3 we presented the various models we experimented with. Since tuning model parameters can get tricky and hard, especially for people without machine learning expertise, we did not want our user to be involved with tuning the model. Instead, through experimentation, we find a stable hyperparameter configuration, that worked across all the domains we experimented with in §3.

Our dictionary models, simply memorize the training data so there are no hyper parameters involved. However, for our neural models we list the hyper parameters for the models and their training algorithms.

Word-Level BiLSTM CRF: The embedding layer has an embedding dimension of 300, the BiLSTM has 1000 hidden dimensions (500 in each direction). The learning rate for the SGD optimizer is 0.01, and the weight decay was 1e-4. The batch size used for training is 1. The supervised model was trained for 15 epochs on each corpus, and the best performing model was selected. During the active learning experiments, at each dataset size the model was trained for 5 epochs, and the best performing one was reported.

ELMo BiLSTM CRF: We use 1024 dimension ELMo embeddings, the BiLSTM has 1000 hidden dimensions (500 in each direction). The learning rate for the SGD optimizer is 0.01, and the weight decay was 1e-4. The batch size used for training is 1. The supervised model was trained for 15 epochs on each corpus, and the best performing model was selected. During the active learning experiments, at each dataset size the model was trained for 5 epochs, and the best performing one was reported.

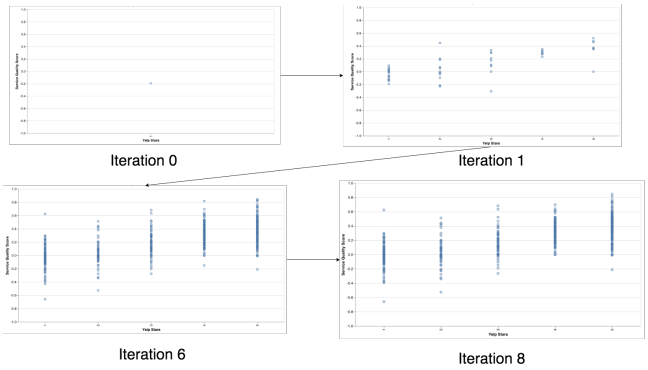

Figure 9: In this figure, we can see a view of service score vs. yelp stars for our Yelp Case Study (§5.1.2). This shows hour our changes through iterations. We can see large changes in the first couple iterations, and minimal changes in the last few signifying that the visualizations have stabilized and model is nearing optimal performance.

C MODEL PERFORMANCE

In §3.1, we presented the concern of the scalability of our neural models and our caching models, which cache the intermediate ELMo vectors to save them from being computed during run time. In this section we present an experiment showing the performance increase of using our caching infrastructure on both CPU and GPU, in both prediction (forward pass of network) and training (forward and backward pass of network). Our results are presented in table 2, we show that with both CPU and GPU our models are much faster when relying on the caching infrastructure. All experiments are done on the CADEC dataset. Our models are implemented in pytorch [27] and rely on ELMo from AI2's AllenNLP [15].

	Without Cache		With Cache	
	CPU it/s	GPU it/s	CPU it/s	GPU it/s
Prediction	0.83	5.11	15.42	22.49
Training	0.57	3.70	1.47	12.54

Table 2: We present results for prediction and training with and without our cacheing models implemented, to show our speedup. We can see a drastic increase in iterations per second for prediction and training when the cache is used.