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.

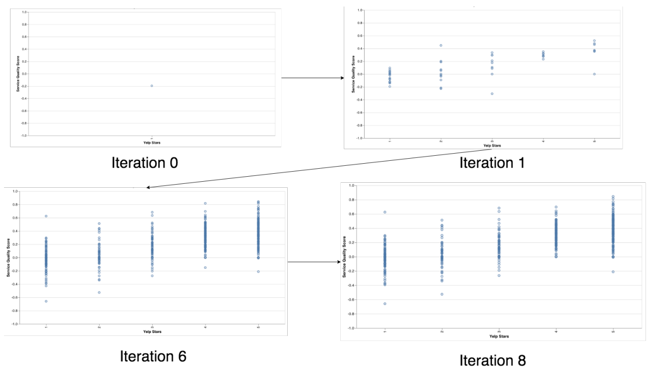

Figure 9: In this figure, we can see a view of service score vs. yelp stars for our Yelp Case Study (§5.1.2). This shows hour our changes through iterations. We can see large changes in the first couple iterations, and minimal changes in the last few signifying that the visualizations have stabilized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