Noise Pollution in Hospital Readmission Prediction: Long Document Classification with Reinforcement Learning

Liyan Xu[†] **Julien Hogan**[‡] **Rachel Patzer**[‡] **Jinho D. Choi**[†]

[†]Department of Computer Science, [‡]Department of Surgery

Emory University, Atlanta, US

`{liyan.xu,julien.hogan,rpatzer,jinho.choi}@emory.edu`

Abstract

This paper presents a reinforcement learning approach to extract noise in long clinical documents for the task of readmission prediction after kidney transplant. We face the challenges of developing robust models on a small dataset where each document may consist of over 10K tokens with full of noise including tabular text and task-irrelevant sentences. We first experiment four types of encoders to empirically decide the best document representation, and then apply reinforcement learning to remove noisy text from the long documents, which models the noise extraction process as a sequential decision problem. Our results show that the old bag-of-words encoder outperforms deep learning-based encoders on this task, and reinforcement learning is able to improve upon baseline while pruning out 25% text segments. Our analysis depicts that reinforcement learning is able to identify both typical noisy tokens and task-specific noisy text.

1 Introduction

Prediction of hospital readmission has always been recognized as an important topic in surgery. Previous studies have shown that the post-discharge readmission takes up tremendous social resources, while at least a half of the cases are preventable (Basu Roy et al., 2015; Jones et al., 2016). Clinical notes, as part of the patients' Electronic Health Records (EHRs), contain valuable information but are often too time-consuming for medical experts to manually evaluate. Thus, it is of significance to develop prediction models utilizing various sources of unstructured clinical documents.

The task addressed in this paper is to predict 30-day hospital readmission after kidney transplant, which we treat it as a long document classification problem without using specific domain knowledge. The data we use is the unstructured clinical documents of each patient up to the date of discharge.

In particular, we face three types of challenges in this task. First, the document size can be very long; documents associated with these patients can have tens of thousands of tokens. Second, the dataset is relatively small with fewer than 2,000 patients available, as kidney transplant is a non-trivial medical surgery. Third, the documents are noisy, and there are many target-irrelevant sentences and tabular data in various text forms (Section 2).

The lengthy documents together with the small dataset impose a great challenge on representation learning. In this work, we experiment four types of encoders: bag-of-words (BoW), averaged word embedding, and two deep learning-based encoders that are ClinicalBERT (Huang et al., 2019) and LSTM with weight-dropped regularization (Merity et al., 2018). To overcome the long sequence issue, documents are split into multiple segments for both ClinicalBERT and LSTM (Section 4).

After we observe the best performed encoders, we further propose to combine reinforcement learning (RL) to automatically extract out task-specific noisy text from the long documents, as we observe that many text segments do not contain predictive information such that removing these noise can potentially improve the performance. We model the noise extraction process as a sequential decision problem, which also aligns with the fact that clinical documents are received in time-sequential order. At each step, a policy network with strong entropy regularization (Mnih et al., 2016) decides whether to prune the current segment given the context, and the reward comes from a downstream classifier after all decisions have been made (Section 5).

Empirical results show that the best performed encoder is BoW, and deep learning approaches suffer from severe overfitting under huge feature space in contrast of the limited training data. RL is experimented on this BoW encoder, and able to improve upon baseline while pruning out around 25%

Type	P	T	Description
CO	1,354	4,395.3	Report for every outpatient consultation before transplantation
DS	514	1,296.7	Summary at the time of discharge from every hospital admission happened before transplant
EC	1,110	1,073.6	Results of echocardiography
HP	1,422	3,025.1	Summary of the patient's medical history and clinical examination
OP	1,472	4,224.8	Report of surgical procedures
PG	1,415	13,723.4	Medical note during hospitalization summarizing the patient's medical status each day
SC	2,033	1,189.2	Report from the evaluation of each transplant candidate by the selection committee
SW	1,118	1,407.6	Report from encounters with social workers

Table 1: Statistics of our dataset with respect to different types of clinical notes. P: # of patients, T: avg. # of tokens, CO: Consultations, DS: Discharge Summary, EC: Echocardiography, HP: History and Physical, OP: Operative, PG: Progress, SC: Selection Conference, SW: Social Worker. The report for SC is written by the committee that consists of surgeons, nephrologists, transplant coordinators, social workers, etc. at the end of the transplant evaluation. All 8 types follow the approximately 3:7 positive-negative class distribution.

text segments (Section 6). Further analysis shows that RL is able to identify traditional noisy tokens with few document frequencies (DF), as well as task-irrelevant tokens with high DF but of little information (Section 7).

2 Data

This work is based on the Emory Kidney Transplant Dataset (EKTD) that contains structured chart data as well as unstructured clinical notes associated with 2,060 patients. The structured data comprises 80 features that are lab results before the discharge as well as the binary labels of whether each patient is readmitted within 30 days after kidney transplant or not where 30.7% patients are labeled as positive.

The unstructured data includes 8 types of notes such that all patients have zero to many documents for each note type. It is possible to develop a more accurate prediction model by co-training the structured and unstructured data; however, this work focuses on investigating the potentials of unstructured data only, which is more challenging.

2.1 Preprocessing

As the clinical notes are collected through various sources of EMRs, many noisy documents exist in EKTD such that 515 documents are HTML pages and 303 of them are duplicates. These documents are removed during preprocessing. Moreover, most documents contain not only written text but also tabular data, because some EMR systems can only export entire documents in the table format. While there are many tabular texts in the documents (e.g., lab results and prescription as in Table 2), it is impractical to write rules to filter them out, as the exported formats are not consistent across EMRs. Thus, any tokens containing digits or symbols, except for one-character tokens, are removed during

Lab Fishbone (BMP, CBC, CMP, Diff) and critical labs - Last 24 hours 03/08/2013 12:45 142(Na) 104(Cl) 70H(BUN) - 10.7L(Hgb) < 92(Glu) 6.5(WBC) 137L(Plt) 3.6(K) 26(CO2)

Table 2: An example of tabular text in EKTD.

preprocessing. Although numbers may provide useful features, most quantitative measurements are already included in the structured data so that those features can be better extracted from the structured data if necessary. The remaining tabular text contains headers and values that do not provide much helpful information and become another source of noise, which we handle by training a reinforcement learning model to identify them (Section 5).

Table 1 gives the statistics of each clinical note type after preprocessing. The average number of tokens is measured by counting tokens in all documents from the same note type of each patient. Given this preprocessed dataset, our task is to take all documents in each note type as a single input and predict whether or not the patient associated with those documents will be readmitted.

3 Related Work

Shin et al. (2019) presented ensemble models utilizing both the structured and the unstructured data in EKTD, where separate logistic regression (LR) models are trained on the structured data and each type of notes respectively, and the final prediction of each patient is obtained by averaging predictions from each models. Since some patients may lack documents from certain note types, prediction on these note types are simply ignored in the averaging process. For the unstructured notes, concatenation of Term Frequency-Inverse Document Frequency

(TF-IDF) and Latent Dirichlet Allocation (LDA) representation is fed into LR. However, we have found that the representation from LDA only contributes marginally, while LDA takes significantly more inferring time. Thus, we drop LDA and only use TF-IDF as our BoW encoder (Section 4.1).

Various deep learning models regarding text classification have been proposed in recent years. Pre-trained language models like BERT have shown state-of-the-art performance on many NLP tasks (Devlin et al., 2019). ClinicalBERT is also introduced on the medical domain (Huang et al., 2019). However, deep learning approaches have two drawbacks on this particular dataset. First, deep learning requires large dataset to train, whereas most of our unstructured note types only have fewer than 2,000 samples. Second, these approaches are not designed for long documents, and difficult to keep long-term dependencies over thousands of tokens.

Reinforcement learning has been explored to combat data noise by previous work (Zhang et al., 2018; Qin et al., 2018) on the short text setting. A policy network makes decision left-to-right over tokens, and is jointly trained with another classifier. However, there is little investigation of using RL on the long text setting, as it still requires an effective encoder to give meaningful representation of long documents. Therefore, in our experiments, the first step is to select the best encoder, and then apply RL on the long document classification.

4 Document Representation

4.1 Bag-of-Words

For the baseline model, the bag-of-words representation with TF-IDF scores, excluding stopwords (Nothman et al., 2018), is fed into logistic regression (LR). The objective is to minimize the negative log likelihood of the gold label y_i:

$$-\frac{1}{m}\sum_{i=1}^{m}[y_i \log p(g_i)+(1-y_i)\log 1-p(g_i)] \quad (1)$$

where g_i is the TF-IDF representation of D_i. In addition, we experiment two common techniques in the encoder to reduce feature space: token stemming, and document frequency cutoff.

4.2 Averaged Word Embedding

Word embeddings generated by fastText are used to establish another baseline, that utilizes subwords to better represent unseen terms (Bojanowski et al.,

2017). It is suitable for this task as unseen terms or misspellings frequently appear in these clinical notes. The averaged word embedding is used to represent the input document consisting of multiple notes, which gets fed into LR with the same training objective.

4.3 ClinicalBERT

Following Huang et al. (2019), the pretrained language BERT model (Devlin et al., 2019) is first tuned on the MIMIC-III clinical note corpus (Johnson et al., 2016), which has shown to provide better related word similarities in medical domains. Then, a dense layer is added on the CLS token of the last BERT layer. The entire parameters are fine-tuned to optimize the binary cross entropy loss, that is the same objective as Equation 1.

Since BERT has a limit on the input length, the input document of each patient is split into multiple subsequences. Each subsequence is within the BERT length limit, and serves as an independent sample with the same label of the patient. The training data is therefore noisily inflated. The final probability of readmission is computed as follows:

$$p(y_i = 1|g_i) = \frac{p_{\max}^{n_i} + p_{\mean}^{n_i} n_i/c}{1 + n_i/c} \quad (2)$$

where g_i is the BERT representation of patient i, n_i is the corresponding number of subsequences, and c is a hyperparameter to control the influence of n_i. $p_{\max}^{n_i}$ and $p_{\mean}^{n_i}$ are the max and mean probability across the subsequences, respectively.

The motivation behind balancing between the max and mean probability is that subsequences do not contain equal information. $p_{\max}^{n_i}$ represents the best potential, while longer text should give more importance to $p_{\mean}^{n_i}$, because $p_{\max}^{n_i}$ is more easily affected by noise as the text length grows. Although Equation 2 seems intuitive, the use of pseudo labels on subsequences becomes another source of noise, especially when there are thousands of tokens; thus, the performance is uncertain. Section 6.2 provides detailed empirical analysis for this model.

4.4 Weight-dropped LSTM

We split documents of each patient into multiple short segments, and feed the segment representation to long short-term memory network (LSTM) at each time step:

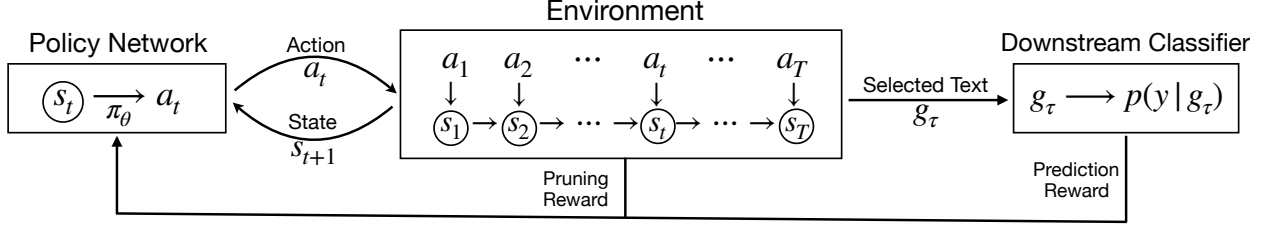$$h_j \leftarrow \text{LSTM}(s_j, h_{j-1}; \theta) \quad (3)$$

Figure 1: Overview of our reinforcement learning approach. Rewards are calculated and sent back to the policy network after all actions $a_{1:T}$ have been sampled for the given episode.

where h_j is the hidden state at time step j, s_j is the jth segment, and θ is the set of parameters. Although segmentation of documents is still necessary, no pseudo labels are needed. We get the segment representation by averaging its token embedding from the last layer of BERT. The final hidden state at each step j is the concatenated hidden states of a single-layer Bi-directional LSTM. After we get the hidden state for each segment, a max-pooling operation is performed on $h_{1:n}$ over the time dimension to obtain a fixed-length vector, similar to Kim (2014); Adhikari et al. (2019). A dense layer is immediately followed.

It is particularly important to strengthen regularization on this dataset with small sample size. Dropout (Srivastava et al., 2014) as a way of regularization has been shown effective in deep learning models, and Merity et al. (2018) has successfully applied dropout-like technique in LSTM: the use of DropConnect (Wan et al., 2013) is applied on the four hidden-to-hidden matrices, preventing overfitting from occurring on the recurrent weights.

5 Reinforcement Learning

Reinforcement learning is applied to the best performing encoder in Section 4 to prune noisy text, which can lead to comparable or even better performance, as many text segments in these clinical notes are found to be irrelevant to this task. Figure 1 describes the overview of our reinforcement learning approach. The pruning process is modeled as a sequential decision problem, for the fact that these notes are received in time-order. It consists of two separate components: a policy network, and a downstream classifier. To avoid having too many time steps, the policy is performed on the segment level instead of token level. For each patient, documents are split into short segments $g_{1:T} = \{g_1, g_2, \cdots, g_T\}$, and the policy network conducts a sequence of decisions $a_{1:T} = \{a_1, a_2, \cdots, a_T\}$ over segments. The downstream classifier is re-

sponsible for the reward, and the REINFORCE algorithm is used to train the policy (Williams, 1992).

State At each time step, the state s_t is the concatenation of two parts: the representation of previously selected text, and the current segment representation g_i. The previously selected text serves as the context and provides a prior importance. Both parts are represented by an effective encoder, e.g. the best performing encoder from Section 4.

Action The action space at each step is binary: $\{\texttt{Keep}, \texttt{Prune}\}$. If the action is $\texttt{Keep}$, the current segment is added to the selected text; otherwise, it is discarded. The final selected text for a patient is the concatenated segments selected by the policy.

Reward The reward comes at the end when all actions are sampled for the entire sequence. The final selected text is fed to the downstream classifier, and negative log-likelihood of the gold label is used as the reward R. In addition, we also include a reward term R_p to encourage pruning, as follows:

$$R_p = c \cdot \alpha \cdot [2\sigma(\frac{l}{\beta}) - 1] \qquad (4)$$

where c and β are hyperparameters to control the scale of R_p, l is the number of segments, α is the ratio of pruned segments $|\{a_k = \texttt{Prune}\}|/l$, σ is the sigmoid function. The value of the term $2\sigma(\frac{l}{\beta}) - 1$ falls into range $(0, 1)$. When l is small, it downgrades the encouragement of pruning; when l is large, it also gives an upper bound of R_p. Additionally, we apply exponential decay on the reward. The final reward is $d^l R + R_p$. d is the discount rate.

Policy Network The policy network maintains a stochastic policy $\pi(a_t|s_t; \theta)$:

$$\pi(a_t|s_t; \theta) = \sigma(W s_t + b) \qquad (5)$$

where θ is the set of policy parameters W and b, a_t and s_t are the action and state at the time step t respectively. During training, an action is sampled at

Encoder	CO	DS	EC	HP	OP	PG	SC	SW
Bag-of-Words (§4.1)	58.6	62.1	52.0	58.9	51.8	61.2	**59.3**	51.6
+ Cutoff	58.6	**62.3**	52.8	59.0	51.9	61.3	**59.3**	**51.9**
+ Stemming	**58.9**	61.8	**53.4**	**59.4**	51.9	**61.5**	**59.3**	51.6
Averaged Embedding (§4.2)	56.3	53.7	52.4	54.0	**53.4**	54.7	54.2	46.6
ClinicalBERT (§4.3)	51.9	53.3	-	52.7	-	-	52.3	-
Weight-dropped LSTM (§4.4)	53.7	55.8	-	54.2	-	-	54.5	-

Table 3: The Area Under the Curve (AUC) scores achieved by different encoders on the 5-fold cross-validation. See the caption in Table 1 for the descriptions of CO, DS, EC, HP, OP, PG, SC, and SW. For deep learning encoders, only four types are selected in experiments (Section 6.2).

each step with the probability from the policy. After the sampling is performed over the entire sequence, the delayed reward is computed. During evaluation, the action is picked by $\arg\max_a \pi(a|s_t; \theta)$.

The training is guided by the REINFORCE algorithm (Williams, 1992), which optimizes the policy to maximize the expected reward:

$$J(\theta) = \mathbb{E}_{a_{1:T} \sim \pi} R_{a_{1:T}} \quad (6)$$

and the gradient has the following form:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \sum_{t=1}^{T} \nabla_\theta \log \pi(a_t|s_t; \theta) R_\tau \quad (7)$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi(a_{it}|s_{it}; \theta) R_{\tau_i} \quad (8)$$

where τ represents the sampled trajectory $\{a_1, a_2, \cdots, a_T\}$, N is the number of sampled trajectories. R_{τ_i} here equals the delayed reward from the downstream classifier at the last step.

To encourage exploration and avoid local optima, we add the entropy regularization (Mnih et al., 2016) on the policy loss:

$$J_{reg}(\theta) = \frac{\lambda}{N} \sum_{i=1}^{N} \frac{1}{T_i} \nabla_\theta H(\pi(s_{it}; \theta)) \quad (9)$$

where H is the entropy, and λ is the regularization strength, T_i is the trajectory length.

Finally, the downstream classifier and policy network are warm-started by separate training, and then jointly trained together.

6 Experiments

Before experiments, we perform the preprocessing described in Section 2.1, and then randomly split patients in every note type by 5 folds to perform cross-validation as suggested by Shin et al. (2019). To evaluate each fold F_i, 12.5% of the training set, that is the combined data of the other 4 folds, are held out as the development set and the best configuration from this development set is used to decode F_i. The same split is used across all experiments for fair comparison. Following Shin et al. (2019), the averaged Area Under the Curve (AUC) across these 5 folds is used as the evaluation metric.

6.1 Baseline

Bag-of-Words We first conduct experiments using the bag-of-words encoder (BoW; Section 4.1) to establish the baseline. Many experiments are performed on all note types using the vanilla TF-IDF, document frequency (DF) cutoff at 2 (removing all tokens whose DF ≤ 2), and token stemming. For every experiment, the class weight is assigned inversely proportional to class frequencies, and the inverse of regularization strength C is searched from $\{0.01, 0.1, 1, 10\}$, where the best results are achieved with $C = 1$ on the development set.

Table 3 describes the cross-validation results on every note type. The top AUC is 62.3%, which is within expectation given the difficulty of this task. Some note types are not as predictive as the others, such as Operative (OP) and Social Worker (SW), with the AUC under 52%. Most note types have the standard deviations in range 0.02 to 0.03.

In comparison to the previous work (Shin et al., 2019), we achieve 0.671 AUC combining both structured and unstructured data, despite without the use of LDA in our encoder.

Noise Observation The DF cutoff coupled with token stemming significantly reduce feature space for the BoW model. As shown in Table 4, the DF cutoff itself can achieve about 50% reduction of the feature space. Furthermore, applying the DF cutoff leads to slightly higher AUCs on most of the note

types, despite almost a half of the tokens are removed from the vocabulary. This implies that there exists a large amount of noisy text that appears only in few documents, causing the models to be overfitted more easily. These results further verify our previous observation and strengthen the necessity to extract noise from these long documents using reinforcement learning (Section 6.3).

Averaged Word Embedding   For the averaged word embedding encoder (AWE; Section 4.2), embeddings generated by FastText trained on the Common Crawl and the English Wikipedia with the 300 dimension is used.[1] AWE is outperformed by BoW on every note type except Operative (OP; Table 3). This empirical result implies that AWE over thousands of tokens is not so effective in generating the document representation so that the averaged embeddings are less discriminative than the sparse vectors generated by BoW for such long documents.

Type	Vanilla	+ Cutoff	+ Stemming
CO	28,213	15,022 (46.8)	12,243 (56.6)
DS	11,029	6,117 (44.5)	5,228 (52.6)
HP	20,245	11,276 (44.3)	9,329 (53.9)
SC	19,050	9,873 (48.2)	8,200 (57.0)

Table 4: The dimensions of the feature spaces used by each BoW model with respect to the four note types. The numbers in the parentheses indicate the percentage reduction from the vanilla model, respectively.

6.2   Deep Learning-based Encoders

For deep learning encoders, the four note types with good baseline performance ($\approx 60\%$ AUC) and reasonable sequence length (< 5000) are selected to use in the following experiments, which are Consultations (CO), Discharge Summary (DS), History and Physical (HP), and Selection Conference (SC) (see Tables 1 and 3).

Segmentation   For both ClinicalBERT and the LSTM models, the input document is split into segments as described in Section 4.3. For LSTM, we set the maximum segment length to be 128 for CO and HP, 64 for DS and SC, to balance between segment length and sequence length. The segment length for ClinicalBERT is set to 318 (approaching 500 after BERT tokenization) to avoid noise brought by too many pseudo labels. More statistics about segmentation are summarized in Table 5.

[1]https://fasttext.cc/docs/en/crawl-vectors.html

For the ClinicalBERT, we use the PyTorch BERT implementation with the base configuration:[2] 768 embedding dimensions and 12 transformer layers, and we load the weights provided by Huang et al. (2019) whose language model has been finetuned on large-scale clinical notes.[3] We finetune the entire ClinicalBERT with batch size 4, learning rate 2×10^{-5}, and weight decay rate 0.01.

For the weight-dropped LSTM, we set the batch size to 64, the learning rate to 10^{-3}, the weight-drop rate to 0.5, and search the hidden state dimension from $\{128, 256, 512\}$ on the development set. Early stop is used for both approaches.

Type + Model	SEN	SEQ	INST
CO + BERT	318	14.8	11,376
CO + LSTM	128	36.8	948
DS + BERT	318	4.6	1,588
DS + LSTM	64	22.5	371
HP + BERT	318	10.1	8,364
HP + LSTM	128	27.3	987
SC + BERT	318	3.7	5,206
SC + LSTM	64	25.4	1,422

Table 5: SEN: maximum segment length (number of tokens) allowed by the corresponding model, SEQ: average sequence length (number of segments), INST: average number of samples in the training set.

Result Analysis   Table 3 shows the final results achieved by the ClinicalBERT and LSTM models. The AUCs of both models experience a non-trivial drop from the baseline. After further investigation, the issue is that both models suffer from severe overfitting under the huge feature spaces, and struggle to learn generalized decision boundaries from this data. Figure 2 shows an example of the weak correlation between the training loss and the AUC scores on the development set.

As more steps are processed, the training loss gradually decreases to 0. However, the model has high variance and it does not necessarily give better performance on the development set as the training loss drops. This issue is more apparent with ClinicalBERT on CO because there are too many pseudo labels acting as noise, which makes it harder for the model to distinguish useful patterns from noise.

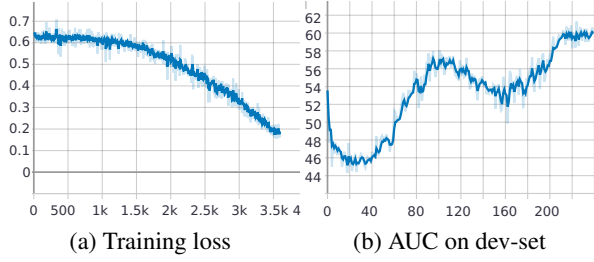[2]https://github.com/huggingface/transformers
[3]https://github.com/kexinhuang12345/clinicalBERT

(a) Training loss   (b) AUC on dev-set

Figure 2: Training loss and AUC scores on the development set during the LSTM training on the `CO` type. The AUC scores depict high variance while showing weak correlation to the training loss.

6.3 Reinforcement Learning

According to Table 3, the BoW model achieves the best performance. Therefore, we decide to use TF-IDF to represent the long text of each patient, along with logistic regression as the classifier for reinforcement learning. Document segmentation is the same as LSTM (Table 5). During training, segments within each note are shuffled to reduce overfitting risks, and sequences with more than 36 segments are truncated.

The downstream classifier is warm-started by loading weights from the logistic regression model in the previous experiment. The policy network is then trained for 400 episodes while freezing the downstream classifier. After the warm start, both models are jointly trained. We set the number of sampling N as 10 episodes, learning rate 2×10^{-4}, and fix the scaling factor β in Equations 4 as 8, and discount rate as 0.95. Moreover, we search the reward coefficient c in $\{0.02, 0.1, 0.4\}$, and entropy coefficient λ in $\{2, 4, 6, 8\}$.

	CO	DS	HP	SC
Best	58.9	62.3	59.4	59.3
RL	59.8	62.4	60.6	60.2
Pruning	26%	5%	19%	23%

Table 6: The AUC scores and the pruning ratios of reinforcement learning (RL). Best: AUC scores from the best performing models in Table 3.

The AUC scores and the pruning ratios (the number of pruned segments divided by the sequence length) are shown in Table 6. Our reinforcement learning approach outperforms the best performing models in Table 3, achieving around 1% higher AUC scores on three note types, `CO`, `HP`, and `SC`, while pruning out up to 26% of the input documents.

Tuning Analysis   We find that two hyperparameters are essential to the final success of reinforcement learning (RL). The first is the reward discount rate d. The scale of the policy gradient $\nabla_\theta J(\theta)$ depends on the sequence length T, while the delayed reward R_τ is always on the same scale regardless of T. Therefore, different sequence length across episodes causes turbulence on the policy gradient, leading to unstable training. It is important to apply reward decay to stabilize the scale of $\nabla_\theta J(\theta)$.

The second is the entropy regularization coefficient λ, which forces the model to add bias towards uncertainty. Without strong entropy regularization, the training is easy to fall into local optima in early stage, which is to keep all segments, as shown by Figure 3(a). $\lambda = 6$ gives the model descent incentive to explore aggressively, as shown by Figure 3(b), and finally leads to higher AUC.
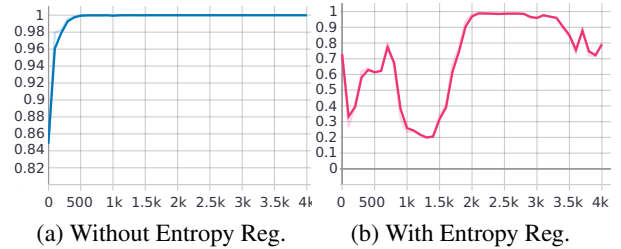

(a) Without Entropy Reg.   (b) With Entropy Reg.

Figure 3: Retaining ratios on the development set of `SC` while training the reinforcement learning model. Entropy regularization encourages more exploration.

7 Noise Analysis

To investigate the noise extracted by RL, we analyze the pruned segments on the validation sets of the Consultations type (`CO`), and compare the results with other basic noise removal techniques.

Qualitative Analysis   Table 7 demonstrates the potential of the learned policy to automatically identify noisy text from the long documents. The original notes of shown examples are tabular text with headers and values, mostly lab results and medical prescription. After the data cleaning step, the text becomes broken and does not make much sense for humans to evaluate. The learned policy can identify noisy segments by looking at the presence of headers such as "lab fishbone", "lab report", and certain medical terms that frequently appear in tabular reports such as "chloride", "creatinine", "hemoglobin", "methylprednisolone", etc. We find that many pruned segments have strong indicators

lab fishbone (bmp , cbc , **cmp** , **diff**) and critical labs - last hours (not an official **lab report** . please see flowsheet (or printed official **lab** reports) for official **lab results** .) (na) (cl) h (**bun**) - (hgb) (glu) (wbc) (plt) ( ) h (**cr**) (hct) na = not applicable a = abnormal (ftn) = **footnote** .
laboratory studies : sodium , **potassium** , **chloride** , . , **bun** , **creatinine** , **glucose** . **total** bilirubin 1 , phos of , calcium , **ast** 9 , **alt** , alk phos . parathyroid hormone level . white blood cell count , **hemoglobin** , hematocrit , **platelets** . inr , ptt , and pt .
methylprednisolone ivpb : mg , **ivpb** , give in surgery , routine , / , infuse over : minute . **mycophenolate mofetil** : mg = 4 cap , po , capsule , once , now , / , stop date / , ml . documented medications documented accupril : mg , po , **qday** , 0 refill , substitution allowed .

Table 7: Examples of pruned segments by the learned policy. Tokens that have feature importance lower than -0.001 (towards *Prune* action) are marked bold.

the **social worker** met with this pleasant year old **caucasian male** on this date for kidney transplant **evaluation** . the patient was **alert** , oriented and easily **engaged** in conversation with the **social worker** today . he resides in atlanta with his spouse of years , who he describes as very **supportive** .
he **reports** occasional alcohol **drinks** per month but **denies** any illicit drug **use** . he has a **grade** education . he has been **married** for years . he is working full - time while on **peritoneal dialysis** as a business asset manager . he has **medicare** and an aarp **prescriptions** supplement . family history : mother **deceased** at age with complications of **obesity** , **high blood pressure** and **heart** disease .

Table 8: Examples of kept segments by the learned policy. Tokens that have feature importance greater than 0.0005 (towards *Keep* action) are marked bold.

of headers and specific medical terms, which appear mostly in tabular text rather than written notes.

Table 8 shows examples that are kept by the policy. Tokens that contribute towards *Keep* action are words related with human and social life, such as "social worker", "engaged", "drinks", "married", "medicare", and terms related with health conditions, such as "obesity", "heart", "high blood pressure". These terms indeed appear mostly in written text rather than tabular data.

In addition, we also notice that the policy is able to remove certain duplicate segments. Medical professionals sometimes repeat certain description from previous notes to a new document, causing duplicate content. The policy learns to make use of the already selected context, and assigns negative coefficients to certain tokens. Duplicate segments are only selected once if the segment contains many tokens that have opposite feature importance in the context and segment vectors.

Quantitative Analysis We examine tokens that are pruned by RL and compare with document frequency (DF) cutoff. We select 3000 unique tokens in the vocabulary that have the top negative feature importance (towards *Prune* action) in the segment vector of CO. Figure 4 shows the DF distribution of these tokens.
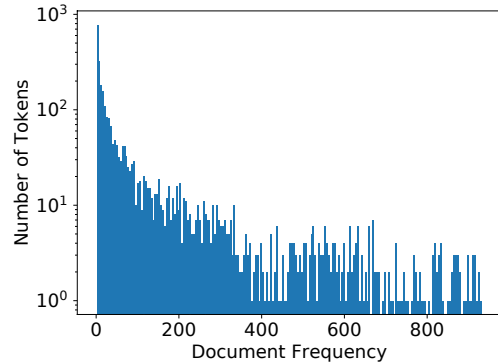

Figure 4: Log scale distribution on document frequency of tokens with top negative feature importance.

We observe that the majority of those tokens have small DF values. It shows that the learned policy is able to identify certain tokens with small DF values as noise, which aligns with DF cutoff. Moreover, the distribution also shows a non-trivial amount of tokens with large DF values, demonstrating that RL can also identify task-specific noisy tokens that commonly appear in documents, which in this case are certain tokens in noisy tabular text.

Either RL or DF cutoff achieves higher AUC while reducing input features, proving that given the small sample size, the extracted text is more likely to cause overfit than being generalizable pattern, which also verifies our initial hypothesis.

8 Conclusion

In this paper, we address the task of 30-day readmission prediction after kidney transplant, and propose to improve the performance by applying reinforcement learning with noise extraction capability. To overcome the challenge of long document representation with a small dataset, four different encoders are experimented. Empirical results show that bag-of-words is the most suitable encoder, surpassing overfitted deep learning models, and reinforcement learning is able to improve the performance, while being able to identify both traditional noisy tokens that appear in few documents, and task-specific noisy text that commonly appear.

References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Rethinking complex neural network architectures for document classification. 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 4046–4051, Minneapolis, Minnesota. Association for Computational Linguistics.

Senjuti Basu Roy, Ankur Teredesai, Kiyana Zolfaghar, Rui Liu, David Hazel, Stacey Newman, and Albert Marinez. 2015. Dynamic hierarchical classification for patient risk-of-readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1691–1700, New York, NY, USA. Association for Computing Machinery.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *CoRR*, abs/1904.05342.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035.

Caroline Jones, Robert Hollis, Tyler Wahl, Brad Oriel, Kamal Itani, Melanie Morris, and Mary Hawn. 2016. Transitional care interventions and hospital readmissions in surgical populations: A systematic review. *The American Journal of Surgery*, 212.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.

Joel Nothman, Hanmin Qin, and Roman Yurchak. 2018. Stop word lists in free open-source software packages. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 7–12, Melbourne, Australia. Association for Computational Linguistics.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2137–2147, Melbourne, Australia. Association for Computational Linguistics.

Bonggun Shin, Julien Hogan, Andrew B. Adams, Raymond J. Lynch, and Jinho D. Choi. 2019. Multimodal ensemble approach to incorporate various types of clinical notes for predicting readmission. In *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1058–III–1066. JMLR.org.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning. In *AAAI Conference on Artificial Intelligence*.