Reinforced Active Learning for Low-Resource, Domain-Specific, Multi-Label Text Classification

Lukas Wertz
University of Stuttgart
lukas.wertz@ims.uni-stuttgart.de

Jasmina Bogojeska
Zurich University of Applied Sciences
bogo@zhaw.ch

Katsiaryna Mirylenka
IBM Research – Zurich
kmi@zurich.ibm.com

Jonas Kuhn
University of Stuttgart
jonas.kuhn@ims.uni-stuttgart.de

Abstract

Text classification datasets from specialised or technical domains are in high demand, especially in industrial applications. However, due to the high cost of annotation such datasets are usually expensive to create. While Active Learning (AL) can reduce the labeling cost, required AL strategies are often only tested on general knowledge domains and tend to use information sources that are not consistent across tasks. We propose Reinforced Active Learning (RAL) to train a Reinforcement Learning policy that utilizes many different aspects of the data and the task in order to select the most informative unlabeled subset dynamically over the course of the AL procedure. We demonstrate the superior performance of the proposed RAL framework compared to strong AL baselines across four intricate multi-class, multi-label text classification datasets taken from specialised domains. In addition, we experiment with a unique data augmentation approach to further reduce the number of samples RAL needs to annotate.

1 Introduction

Modern text classification systems powered by Pre-Trained Language Models achieve excellent accuracy across a variety of tasks and corpora. Consequently, there is a rapidly growing demand to build such systems for increasingly challenging problem settings, such as dealing with domain-specific, low-resource data of high categorical complexity. Typically, this kind of data pertains to specialised domains such as medical or legal, and is often critical in order to realize real world, industrial scale applications. In addition, many of these datasets are multi-label, meaning each sample can have several different categories assigned to it. Annotation in these specialised domains is extremely costly since experts have to be consulted in order to guarantee a reliable assignment of labels.

Active Learning (AL) provides a technique for an effective sample selection that reduces the amount of labeling required to reach a certain model performance or increases model performance given an annotation budget. While AL strategies are numerous, their effectiveness tends to highly depend on the dataset and the classification model used. For example, one of the best-performing AL strategy to select samples based on model confidence can be unreliable if the model has not yet reached a certain level of performance.

In this work we aim to combine the expressiveness of several Active Learning strategies with the capabilities of Reinforcement Learning (RL) in a sequential AL setup. In Reinforced Active Learning (RAL), the agent evaluates samples based on several AL strategies and other model-based features. The resulting decision policy can dynamically adapt to the AL task, underlying classification model and dataset. Selected samples are annotated and added to the labeled pool. The agent then receives a reward based on the classifiers performance given the newly labeled set and consequently, can learn connections between the presented information from the sample representations and the underlying classification model. As such, RAL learns the best combination of AL strategies for a particular dataset and furthermore, retains the ability to sample randomly should the AL strategies be less effective. In addition, RAL also tunes a variety of other parameters that are often set arbitrarily, such as the most beneficial number of samples to be labelled in each AL step.

Our work contains several direct continuations of the Extreme Multi-Label study in (Wertz et al., 2022). In their experiment, the authors showed that no single AL strategy would outperform all baselines across several datasets. In this work, we aim to expand on these findings and devise an adaptive AL strategy that is effective across domains.

The main contributions of our work are as follows:

1. We propose the RAL framework which is a flexible AL tool for Text Classification.
2. RAL performance is evaluated against strong AL baselines on four domain-specific, multi-label text classification datasets, each with more than one hundred classes.
3. RAL parameters and their effect on classification model learning are thoroughly analyzed.
4. We strengthen RAL with a unique data augmentation module allowing the agent to choose samples that were already labeled if they are still beneficial.

2 Related Work

The effectiveness of AL for Text Classification has been subject to extensive research (Tong and Koller (2001), Goudjil et al. (2018)) with specific solutions for deep models (Schröder and Niekler (2020), An et al. (2018a)) and multi-label settings (Reyes et al. (2018), Yang et al. (2009)). AL specifically for Deep Learning however is a topic that still requires further exploration (Ein-Dor et al., 2020). Successful AL strategies for Deep Models often employ only limited use of the classifiers prediction probabilities, for example by using structures within the DNN (Liu et al., 2021), separate selection networks (An et al. (2018b), Gissin and Shalev-Shwartz (2019)) or properties of pre-trained word embeddings (Yuan et al., 2020).

Powerful representations learned by Deep Models can be leveraged for Reinforcement Learning in order to solve tasks "previously out of reach for a machine." (François-Lavet et al., 2018). While results are often difficult to reproduce (Henderson et al., 2018), Deep RL is effectively used for various applications e.g. in games and robotics (Arulkumaran et al., 2017) as well as modeling human intelligence and behaviour, for example, in multi-task, multi-agent settings (Du and Ding (2021), Johanson et al. (2022)).

Learning AL strategies that generalize across domains, has been employed by transferring policies learned with RL from a source task to a target task (Konyushkova et al. (2018), Fang et al. (2017), Desreumaux and Lemaire (2020)). Alternative approaches look to move beyond RL, using imitation learning (Liu et al., 2018) or cyclic "dreaming" (Vu et al., 2019) but keep the concept of transferring the learned policy in order to fully learn on the source task. Similar to our approach (Hsu and Lin, 2015) and (Baram et al., 2004) switch between AL strategies depending on their predicted effectiveness.

The term Active Reinforcement Learning appears in works dealing with improved RL methods, such as guiding the policy search space or attaching a query cost to obtaining the reward (Epshteyn et al., 2008) (Krueger et al., 2020), which share theoretical foundations with our work but are otherwise not related.

Algorithm 1 Reinforced Active Learning for Text Classification using *DeepQ* Reinforcement Learning.

procedure RAL(labeled set D, unlabeled set $U = [U[0], ..., U[n]]$, model M, AL budget b)
  initialize π randomly
  initialize empty transition memory T
  for episodes 1,2,3.... **do**
    shuffle U
    $i \leftarrow 0$
    $s_i \leftarrow get\_next\_state(U[i])$
    while budget b > 0 **do**
      train M_i on D_i
      $s_{i+1} \leftarrow get\_next\_state(U[i+1])$
      $a \leftarrow get\_action(s_{i+1}, \pi)$
      if $a = 1$ **then**
        annotate $U[i+1]$
        $D_{i+1} \leftarrow D_i \cup U[i+1]$
        $U \leftarrow U \setminus U[i+1]$
        $b \leftarrow b - 1$
      else
        $D_{i+1} \leftarrow D_i$
      end if
      if n steps completed **then**
        train M_{i+n} on D_{i+n}
        $r \leftarrow get\_reward(M_{i+n})$
        $j \leftarrow i - n$
        while j < n-1 **do**
          add $\{(s_j, s_{j+1}, r, a)\}$ to T
          $j \leftarrow j + 1$
        end while
      end if
      update π with T
      $i \leftarrow i + 1$
    end while
  end for
end procedure

3 Reinforced Active Learning (RAL)

In Reinforcement Learning an agent interacts with an environment by perceiving a **state**, deciding on an **action** and then receiving a **reward** from the environment based on the action. Over the course of many such interactions, the agent attempts to maximize the cumulative reward and as a consequence, learns to better interact with the environment. The decision process that maps states to actions is called the *policy* π. An important part of learning π is to perform random actions in the early learning stages; a process also called *exploration*. The idea is that the agent can "try out" actions randomly and form an impression of how the newly encountered environment will react. With an increasing number of steps the agent gradually shifts to using π to decide the action.

Active learning strategically selects samples to be annotated from an unlabeled collection U starting from a small initial labeled set D. First, AL trains a model M on D. Then AL improves the performance of M iteratively: Samples are selected, an annotator provides labels for the selected samples and the labeled samples are used to update M. The algorithms used to decide which sentences to select are referred to as *AL Strategies*.

In RAL, we transform AL into a environment that a RL agent can interact with. In particular, we treat each sample from the unlabeled set as an observable **state** which is defined by a variety of features (see Section 3.2). For the **action**, the agent decides whether or not to annotate a sample after observing the corresponding state. Similar to (Fang et al., 2017), we transform AL into a sequence of decisions. The **reward** is received from the performance of M after training with the newly annotated samples. In particular, we do not train the model each time a single sample is annotated. Instead, the model is only trained after a fixed number of steps have been completed (see Section 3.1). We stop the AL when a certain annotation budget is exceeded. π is updated each step following a RL scheme using the model accuracy as reward. When the AL is finished, the environment resets and a new RL episode begins. RAL using *DeepQ*-RL is illustrated in Algorithm 1.

3.1 Reward

We define the reward r as the improvement of the performance of M when training with a batch of newly annotated samples. At the end of the RL episode, the sum of all r are equal to the final accuracy of M. Updating a large, pre-trained language model with a single sentence will not make a significant difference in classification performance. Therefore, we average the reward for each batch of transitions T_m^n over all n steps of the Agent in the batch, where m is the index of the individual step. The reward is split evenly across all transitions in the batch. Thus the agent associates all actions used to select the training batch with a fraction of the received r.

$$T_m^n = \{(s_{l-1}, s_l, \frac{r_{m*n+n}}{n}, a_l) | m*n \leq l \leq m*n+n\} \quad (1)$$

In practice, n corresponds to the number of texts added in each AL step . The transitions in T_k^n are added to the agent's memory once the AL step has been completed.

3.2 State

The set of states S represents the unlabeled texts as well as the progress of AL and the performance of the classification model. Each $s \epsilon S$ consists of features pertaining to M, the progress of AL and the different AL sampling strategies. All features are normalized to $[0, 1]$.

Model Features - We use the average loss and the slope of the loss of M (calculated with linear regression). In addition we also use the average F1 (averaged over all updates of M) and the current reward.*

Active Learning Features - Different AL strategies use various information sources in order to select the most informative samples. We treat each AL strategy as a ranking algorithm that orders all unlabeled samples $u \epsilon U$ by their potential informativeness and returns a ranking $u_k, ..., u_{|U|}$ where the $u_k > u_{k+1}$ w.r.t. the information criterion used by the AL strategy. Whenever the classification model is updated, we run each AL strategy on the unlabeled set and use the resulting ranking until the next update of the classification model.

When building the state for the unlabeled sample u, given the ranking of an AL strategy we retrieve the corresponding rank k. We then fit k to $[0, 1]$ by dividing k with the length of the unlabeled set. Since lower ranks signify the more informative samples, we add $1 - k$ as the information for the corresponding AL strategy to the state.

*We also experiment using the class-wise model predictions but find no significant changes in the results.

The *alps* strategy does not rank the whole unlabeled set. Therefore we instead use the distance to the closest *surprisal embedding* (see Section 4.3) and rank by shortest distance.
Finally, in order to reflect the progress of Active Learning, we also include the remaining annotation budget. Section 4.3 presents all AL strategies used in our experiments, both in the state representation as well as for the baselines.

4 Experiment

4.1 Datasets

For the experiments, we consider 4 multi-label datasets from different domains with hierarchical label structures. For each dataset, we flatten the label hierarchy and consider only the leaves as the final labels. In addition, we filter all labels that occur less than 50 times in total in the corpus as they are very unlikely to be well represented across training, validation and test sets. The classes in all datasets are highly imbalanced as can be seen in Figures **??** to **??** in the Appendix.
ArXiv - The *arxiv* dataset (https://www.kaggle.com/Cornell-University/arxiv) contains scientific abstracts annotated with respective categories. The dataset is licensed under *CC BY - Creative Commons Attribution*.
EurLex57k - The *eurlex* dataset (Chalkidis et al., 2019) contains excerpts from European Law which are annotated with the category tags from EuroVoc[†]. The dataset is licensed under *CC BY - Creative Commons Attribution*.
Patents - The original *patents* dataset was collected from USPTO (https://www.uspto.gov/ip-policy/economic-research/research-datasets). It is annotated with classes from CPC (Cooperative Patent Classification) which correspond to the type of invention. The dataset does not have public access.
Yelp - The *yelp* dataset (https://www.yelp.com/dataset) contains various reviews which are annotated with classes pertaining to the subject of the review. The dataset is licensed under **Yelp Dataset Terms of Use**.

[†]Can be browsed at https://eur-lex.europa.eu/browse/eurovoc.html?locale=en

4.2 Setup

We employ pre-trained language models for text classification by using a single feed-forward output layer on top of BERT (Devlin et al., 2018)[‡]. We train each model for 15 epochs on an NVIDIA RTX A6000 with early stopping and best model selection based on the validation set. For evaluation, whenever we refer to F1 we employ Macro-F1 for all experiments[§]. Micro-F1 is usually higher than Macro-F1 due to the imbalance of classes. We decide to report Macro-F1 on these datasets since it reflects per-class performance of the classification model. Micro-F1 results for all experiments can be found in the Appendix. Similarly, a weighted Macro F1 which is skewed according to the class distribution will also be an easier target than Macro F1.

For evaluating the multi-label text classification we adapt a multi-label evaluation threshold τ by testing a wide range of τ on the validation set and selecting the one which gives the highest Macro-F1. All results are reported on the test set.

For the AL baselines, we employ the same setup and strategies We simulate AL by running each strategy on the fully annotated corpora and querying the oracle annotations instead of using real human annotators. We use an annotation budget of 1000 samples for all datasets. AL starts with a seeded random choice of 100 samples drawn from the respective dataset. This best reflects the use case in which only a handful of labeled samples are available and many classes are still missing from the initial set.

For learning the sample selection policy π (see Section 3) we employ DeepQ Reinforcement Learning. We use the common technique of representing the policy with both a target network and a value network which are realised as feed forward neural networks with a single hidden layer of size 64. We end the RL episode when the annotation budget is reached. Otherwise the classifier is trained in the same way as for AL. A full list of hyperparameters can be found in the Appendix in Table 3.

[‡]using the "bert-base-uncased" model for English from *huggingface*

[§]Macro-F1 calculates F1 per class and averages all classes, while Micro-F1 calculates F1 on the errors of the whole dataset.

	train	val	test	#classes	Macro-F1	Micro-F1
arXiv	352,651	38,899	42,949	114	0.61	0.74
eurlex	44,688	5,962	5,953	739	0.58	0.64
patents	236,281	25,967	111,394	751	0.58	0.72
yelp	1,977,753	224,945	950,391	581	0.52	0.58

Table 1: Details of all datasets describing split sizes, number of classes as well as Macro and Micro F1 on the test set when training a BERT classifier on the full train set.

4.3 Active Learning Strategies

We employ 3 sampling strategies for running the Active Learning and representing the RAL states:

ALPS (Yuan et al., 2020) - Uses BERT language modeling on unlabeled text, calculating *surprisal embeddings* from languge modeling confidence. *ALPS* returns the closest sample to each of n centroids in the clustered suprisal embedding space.

DAL (Gissin and Shalev-Shwartz, 2019) - Uses a separate classifier to discriminate (*Discriminative Active Learning (DAL)* between samples of the labeled and unlabeled collection. The higher the confidence of the discriminator that the sample comes from the unlabeled space, the higher it is ranked for annotation.

Subword (Wertz et al., 2022) - Uses the pretrained language model (BERT) tokenizer to find sentences with many subword units, i.e. unknown words. *subword* is only calculated once at the beginning of AL since the tokenizer does not change. We also considered Reyes et al. (2018) but confirmed the findings from Wertz et al. (2022) that it could not be efficiently computed for high numbers of classes.

In addition, RAL shares a number of characteristics to Policy-Based Active Learning (PAL) presented in Fang et al. (2017) especially with regards to the sequential AL setup (see Section 3). However, PAL relies on policy transfer from a fully annotated dataset to a related, unlabeled dataset. Preliminary experiments with RAL have demonstrated that transferring the policy between unrelated tasks in our setup generally yields worse results than random AL. Consequently, PAL is not a suitable baseline for our task. RAL is a step towards a more generalized application of RL in the context of AL. The RAL framework can in principle be applied to any AL task by substituting the AL strategies and the Classification Model. PAL will be the method of choice when parallel or closely related text datasets are available.

4.4 Results

Figure 1 displays Macro-F1 results when running RAL on the full datasets compared to the AL baselines. We find that across all datasets, RAL yields big improvements of 0.1 Macro-F1 on the *arxiv* dataset up to 0.3 Macro-F1 on the patents dataset. We also observe a difference in how many samples are selected per model training: All AL baselines update the classification model after 100 samples have been selected.[¶] In comparison, the RAL agent selects to annotate less samples, starting at around 25 to 35 samples in early stages of the episode and reducing this number to less then 20 samples later on on the *patents* and *eurlex* datasets. Finally, we find that AL strategies behave differently depending on the dataset and on the *arxiv* dataset specifically, generally perform worse than random selection. RAL consistently outperforms all baselines across all datasets from different domains.

5 Analysis

5.1 AL Update Size

The improvements shown by RAL on the datasets in Figure 1 are large and consistent across the different domains. Moreover, RAL outperforms all the strong AL baselines. We run additional experiments in order to investigate whether the improvement is a result of the dynamic AL update size chosen by the agent in RAL. More specifically, we use the insights from RAL and reduce the number of annotated samples added in each model training step for all AL baselines. We also refer to this number as AL update size. Figure 3 shows how the AL update sizes behave across the datasets. We find that depending on the dataset, RAL decides to add between 10 and 35 samples before training the model. We find the largest update sizes on the *arxiv* dataset, ranging from 25 to 35 samples. The *patents* dataset uses the smallest AL update size on

[¶]Deep AL literature typically uses between 50 and 100 samples for meaningful model updates.

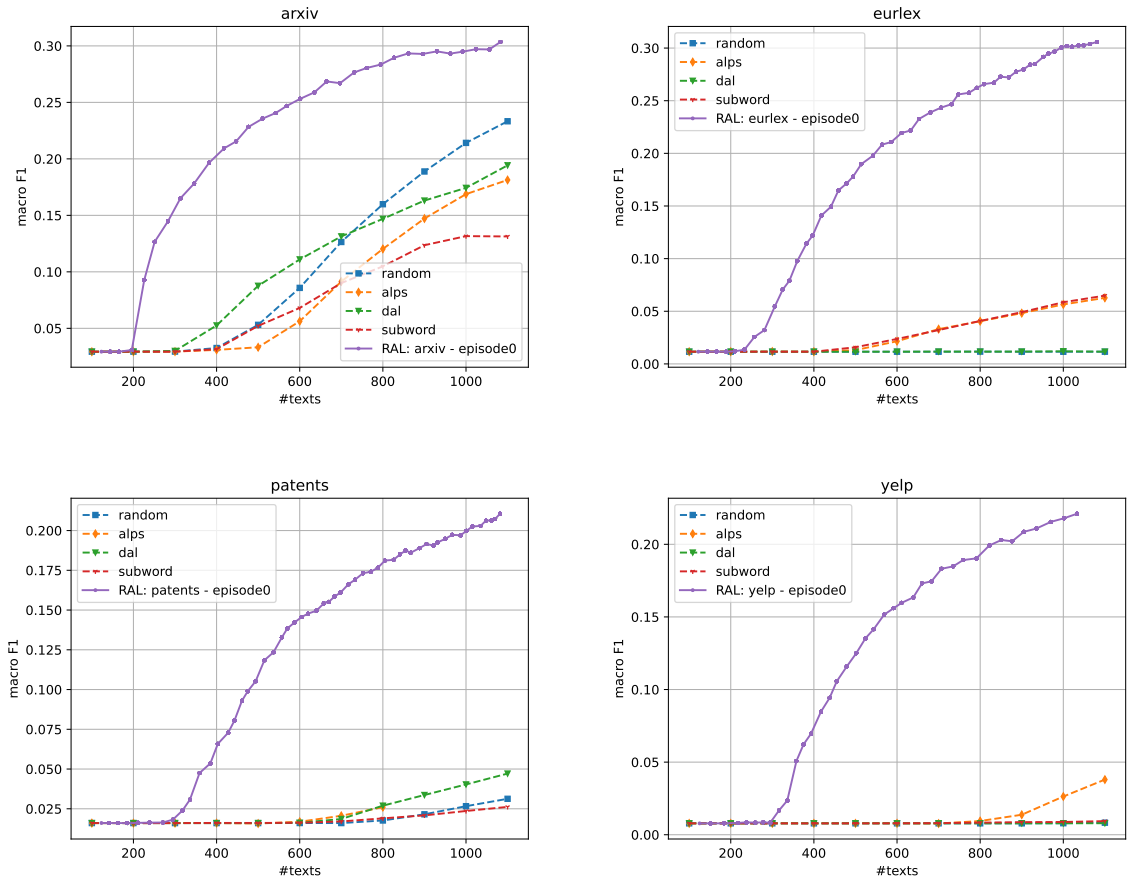

Figure 1: Macro-F1 for a single episode of RAL on full datasets. *random*, *alps*, *dal* and *subword* conditions were run following the AL setup in (Wertz et al., 2022).

average at less than 20 samples. The AL update size is different between datasets and also varies within the RAL episode. This demonstrates that the RAL agent is experimenting with different numbers of annotated samples. It is also likely that the effect of AL update size changes depending on the dataset and current state of the classification model. In order to test this hypothesis, we select the update size of 25 as an average of the RAL results to train the AL baselines.

Figure 2 demonstrates the results. We do find that the performance of all AL baselines greatly improves using an update size of 25 compared to the results shown in Figure 1. Effectively, a smaller update size for the AL leads to additional training of the model, which improves the classification performance. Despite the performance increase especially for the random AL baseline, RAL still outperforms all baselines up to a margin of 0.1 on the patents dataset or converges to the best baseline.

Another important observation from Figure 2 is, that using the smaller AL update size, all AL strategies perform worse than random selection. In contrast, Figure 1 shows that various strategies improve upon random selection. It is clear that the selected samples lose their informative value when the model is trained more often using the smaller AL update size. Consequently, choosing an AL strategy along with an AL update size is not feasible in practice as we do not have information about how the strategy and update size behave on an unknown dataset. RAL has the advantage of dynamically adapting to the dataset during the training process. In addition to adapting the AL update size, the agent can still choose to pay less attention to the AL strategies.

5.2 AL Strategies in RAL

We investigate the effect of the AL strategy rankings used in the state representation of unlabeled data (see Section 3.2). We run an experiment in which we remove all the AL strategies from the

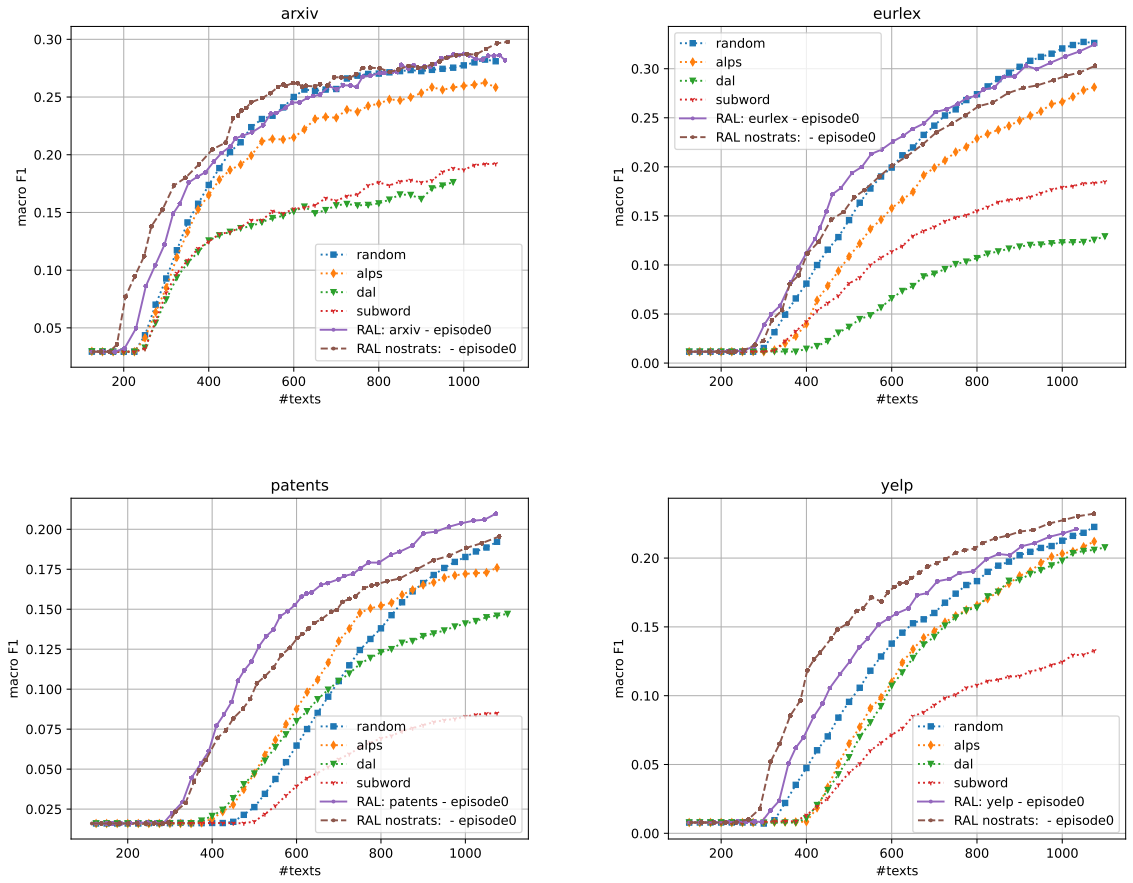

Figure 2: Macro-F1 for a single episode of RAL with and without using AL strategies on all datasets. AL baselines have been trained with an update size of 25.

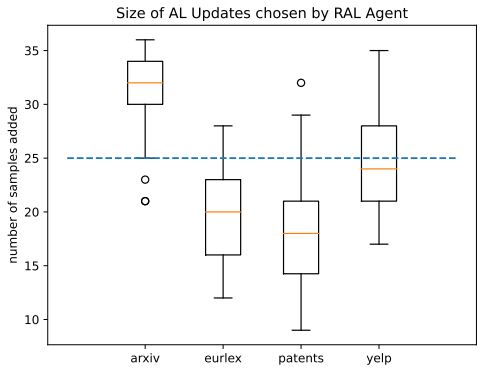

Figure 3: Distribution of the sizes of each AL update as chosen by the RAL agent in the first episode. AL baselines are re-trained with an update size of 25.

state. This leaves the agent only with information about the model and the status of AL (See Section 3.2). The results are shown in Figure 2 under *ARL nostrats*.

We find that RAL performs quite similar without the information from the AL strategies across all 4 full datasets. In particular, on the *yelp* dataset we find that *ARL nostrats* slightly outperforms regular RAL. For the rest of the datasets using AL strategies is slightly beneficial. Given the detrimental effect of the AL strategies shown in Figure 2, this result is not surprising. In fact, if the RAL would have chosen to stuck more closely to any AL strategy, it would only degrade classification performance. The advantage of RAL is, it can learn to utilize or ignore the AL strategies in favor of other sources of information for each specific dataset in the training process.

5.3 Labeled Samples per Step

The RAL agent observes a new unlabeled sample each step. We analyse how many samples are selected for labeling over the course of two episodes. We find that across all datasets, the agent is more conservative in the first episode and selects less observed samples for annotation. Especially on

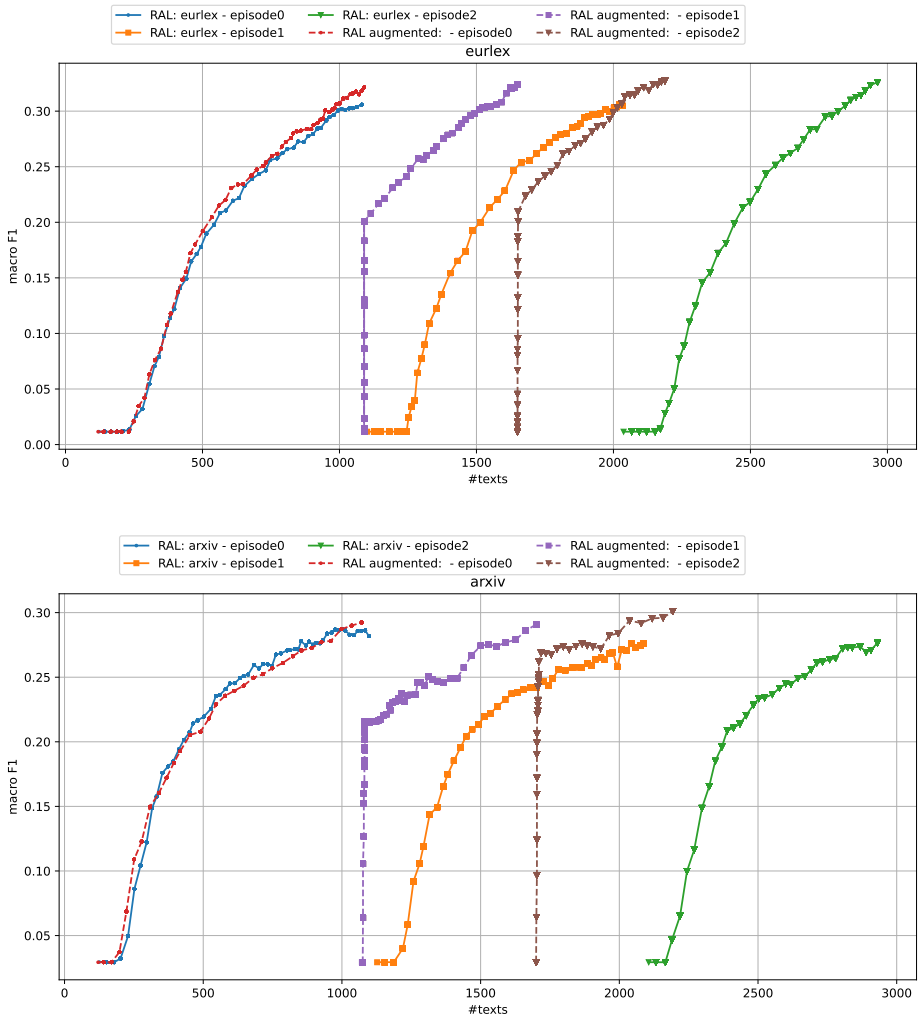

Figure 4: Macro-F1 for multiple episodes of RAL on the *eurlex* and *arxiv* datasets with and without augmentation. x-axis shows total number of annotated samples for a given F1 result.

the *arxiv* and *eurlex* datasets, less than 10% of observed samples are selected. In the second episode, the agent chooses around 50% of observed samples for labeling on all datasets (see Figure 9 in the Appendix for the full results). It is likely that in the first episode, the agent learns that selecting more samples leads to an increase in F1 and then applies this observation immediately at the start of the next episode. However, this behaviour may also prevent the agent from finding potentially more beneficial samples later on. Overall, we assume that within more episodes, the RAL agent will learn to wait for informative samples in later observations.

5.4 Multiple Episodes of RAL with Augmentation

Running several episodes of RL is important for teaching the agent how to find better long-term rewards. In RAL however, additional episodes require the annotation of newly selected samples which increases the annotation cost and thus becomes less effective than simply training with all annotated samples. In order to reduce the annotation cost over several episodes, we employ an augmentation scheme. Instead of randomly sampling from the unlabeled set, we first present the agent with all samples that have been labeled in a previous episode. This way, the agent is encouraged to re-use existing annotations, potentially rebalancing the sparse training set. We demonstrate the effect of multiple episodes of RAL on the *eurlex* and *arxiv* datasets in Figure 4.

We observe that sometimes there is no big performance improvement going from one episode to another. For example, we see that in episode 2 there is a slight performance increase on both datasets over episode 1. Notably, the performance in episode 0

is already quite high given the small training set. As such, we expect such small improvements over episodes. Moreover, on the *arxiv* dataset, episode 1 performs slightly worse than episode 0 by up to 0.025 F1. Subsequent episodes performing worse than previous episodes is not unexpected behaviour in RL. The agent needs to explore parts of the environment that yield less reward in order to get an understanding of less effective actions. Over the course of many episodes, these actions are then remembered and can be avoided. In the RAL setup however, we can not extensively re-run the environment because of the cost of additional annotations. This can be observed in Figure 4 as the end of the third episode approaches 3000 labeled samples.
Figure 4 also demonstrates the performance and number of labeled sentences for the augmented RAL. On both datasets the macro-F1 reaches very similar values to the performance without augmentation, but the necessary number of samples is greatly reduced in the augmented episodes by up to 1000 samples in episode 2, as shown in Table 2. On the *arxiv* dataset, episode 2 with augmentation also reaches higher F1. Overall, the augmentation helps to reach at least the same performance values while using less annotations with the unique ability of RAL to efficiently draw from both the labeled and unlabeled set. Subsequent experiments can include larger amounts of augmentation, such as re-visiting labeled samples from the same episode.

	# annotations	
episode	eurlex	arxiv
0	1000	1000
1	563	672
2	386	494
total	1949	2166

Table 2: Number of new annotated samples per RAL episode on the *eurlex* and *arxiv* datasets using augmentation.

6 Conclusions & Future Work

We present Reinforced Active Learning (RAL), a framework to sequentially learn an effective AL policy for multi-label text classification. RAL improves upon several state-of-the-art AL strategies on 4 multi-class, multi-label datasets from technical or scientific language domains. The RAL approach is easily modifiable and extendable with new AL strategies, more information from the classifier or dataset and can easily be adapted to different kinds of model architectures. Instead of transferring a learned policy from a fully annotated dataset, we run the RL directly on the unlabeled set, requiring only an initial annotated training set and a validation set. We also experiment with augmentation to make running multiple RL episodes feasible. While our experiments use a state-of-the-art BERT-based text classifier, RAL is not limited to Transformer models and can in principle be used for any classifier with slight modifications to the model features. We demonstrate that RAL adapts to different features of the data and outperforms all AL strategies, performing on par with the best AL baseline in the worst case. In addition, RAL still shows improved performance when AL strategies yield worse results than random selection. As our experiments show that AL strategies can fail, future experiments should focus on integrating a wider range of information in the state representation. In addition, we find that augmenting RAL with previously annotated samples is a promising technique to enable longer periods of learning without additional annotations and encourage further research into how to optimize the agent to re-use training samples.

Limitations

All experiments are conducted on data containing exclusively English language. Consequently, the results may differ in particular on morphology-rich languages and/or non-inflectional languages. Similarly, all presented techniques expect languages to use latin characters. Therefore, our method first needs to be adapted in order to be used on languages using different characters, such as Cyrillic languages, Korean or Persian.
Using the BERT-classifier in conjunction with AL is resource intensive. Loading the *"bert-based-uncased"* model from *huggingface* along with one of the datasets with a batch size of 24 requires around 22GB of GPU memory. We train for a maximum of 15 epochs, requiring up to 1 GPU hour, depending on the size of the dataset and the length of individual inputs. As such, a single AL experiment requires approximately 20 GPU hours to complete. In addition, computing the AL strategies requires up to 2 GPU hours for the *alps* strategy and up to 1 GPU hour for the *dal* strategy, depending on the dataset. The subword strategy is calculated only once and used up to 1 GPU hour. In total, our RAL

experiments take around 60 to 100 GPU hours to complete. It is important to note that the RL itself is not expensive and does not require GPU, therefore RAL can easily be adapted to scenarios with low computational resources by employing a different classification model as well as using AL strategies that do not rely on large Pre-Trained Language Models.

References

Bang An, Wenjun Wu, and Huimin Han. 2018a. Deep active learning for text classification. In *Proceedings of the 2nd International Conference on Vision, Image and Signal Processing*, pages 1–6.

Bang An, Wenjun Wu, and Huimin Han. 2018b. Deep active learning for text classification. In *Proceedings of the 2nd International Conference on Vision, Image and Signal Processing*, ICVISP 2018, New York, NY, USA. Association for Computing Machinery.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38.

Yoram Baram, Ran El Yaniv, and Kobi Luz. 2004. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5(Mar):255–291.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019. Large-scale multi-label text classification on eu legislation. *arXiv preprint arXiv:1906.02192*.

Louis Desreumaux and Vincent Lemaire. 2020. Learning active learning at the crossroads? evaluation and discussion. *CoRR*, abs/2012.09631.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Wei Du and Shifei Ding. 2021. A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. *Artificial Intelligence Review*, 54(5):3215–3238.

Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online. Association for Computational Linguistics.

Arkady Epshteyn, Adam Vogel, and Gerald DeJong. 2008. Active reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 296–303.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. *CoRR*, abs/1708.02383.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. 2018. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354.

Daniel Gissin and Shai Shalev-Shwartz. 2019. Discriminative active learning. *arXiv preprint arXiv:1907.06347*.

Mohamed Goudjil, Mouloud Koudil, Mouldi Bedda, and Noureddine Ghoggali. 2018. A novel active learning method using svm for text classification. *International Journal of Automation and Computing*, 15(3):290–298.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Wei-Ning Hsu and Hsuan-Tien Lin. 2015. Active learning by learning. In *Twenty-Ninth AAAI conference on artificial intelligence*.

Michael Bradley Johanson, Edward Hughes, Finbarr Timbers, and Joel Z. Leibo. 2022. Emergent bartering behaviour in multi-agent reinforcement learning.

Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. 2018. Discovering general-purpose active learning strategies. *CoRR*, abs/1810.04114.

David Krueger, Jan Leike, Owain Evans, and John Salvatier. 2020. Active reinforcement learning: Observing rewards at a cost. *CoRR*, abs/2011.06709.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1874–1883, Melbourne, Australia. Association for Computational Linguistics.

Qiang Liu, Yanqiao Zhu, Zhaocheng Liu, Yufeng Zhang, and Shu Wu. 2021. *Deep Active Learning for Text Classification with Diverse Interpretations*, page 3263–3267. Association for Computing Machinery, New York, NY, USA.

Oscar Reyes, Carlos Morell, and Sebastián Ventura. 2018. Effective active learning strategy for multi-label learning. *Neurocomputing*, 273:494–508.

Christopher Schröder and Andreas Niekler. 2020. A survey of active learning for text classification using deep neural networks. *arXiv preprint arXiv:2008.07267*.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66.

Thuy Vu, Ming Liu, Dinh Phung, and Gholamreza Haffari. 2019. Learning how to active learn by dreaming. In *Proceedings of the 57th annual meeting of the Association for Computational Linguistics*, pages 4091–4101.

Lukas Wertz, Katsiaryna Mirylenka, Jonas Kuhn, and Jasmina Bogojeska. 2022. Investigating active learning sampling strategies for extreme multi label text classification. In *Proceedings of the Language Resources and Evaluation Conference*, pages 4597–4605, Marseille, France. European Language Resources Association.

Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. 2009. Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 917–926.

Michelle Yuan, Hsuan-Tien Lin, and Jordan L. Boyd-Graber. 2020. Cold-start active learning through self-supervised language modeling. *CoRR*, abs/2010.09535.

Appendix

Class Distributions on the Training Set

Figure 5 show the distribution of classes on the respective training sets. All datasets have imbalanced class distributions with very few, fery frequent classes and hundreds of rare classes. The *arxiv* dataset contains less classes total than the other datasets and as such, also contains fewer rare classes. We also show the average number of classes per text. The *patents* dataset is the most dense with 5.9 classes per text on average while on the *arxiv* dataset, there are only 1.7 classes annotated for each text on average.

Micro F1 Results

We show the Micro F1 results for all experiments conducted in the main paper. RAL compared to AL baselines are found in Figure 6. The ablation study and re-trained baselines with smaller AL update size are found in Figure 7. The comparison of multiple episodes of RAL with and without augmentation is found in Figure 8. Overall, we find that our observations from the main paper hold for both Micro F1 and Macro F1 results.

RAL Hyperparameters

We list all hyperparameters of RAL in Table 3. Note that the parameters are grouped according to which component of RAL they belong to. *Active Learning* parameters govern how many samples are drawn and when the model is updated. *Text Classifier* parameters belong to the classification model itself. The parameters from *Reinforcement Learning* govern the behaviour of the DeepQ learning.

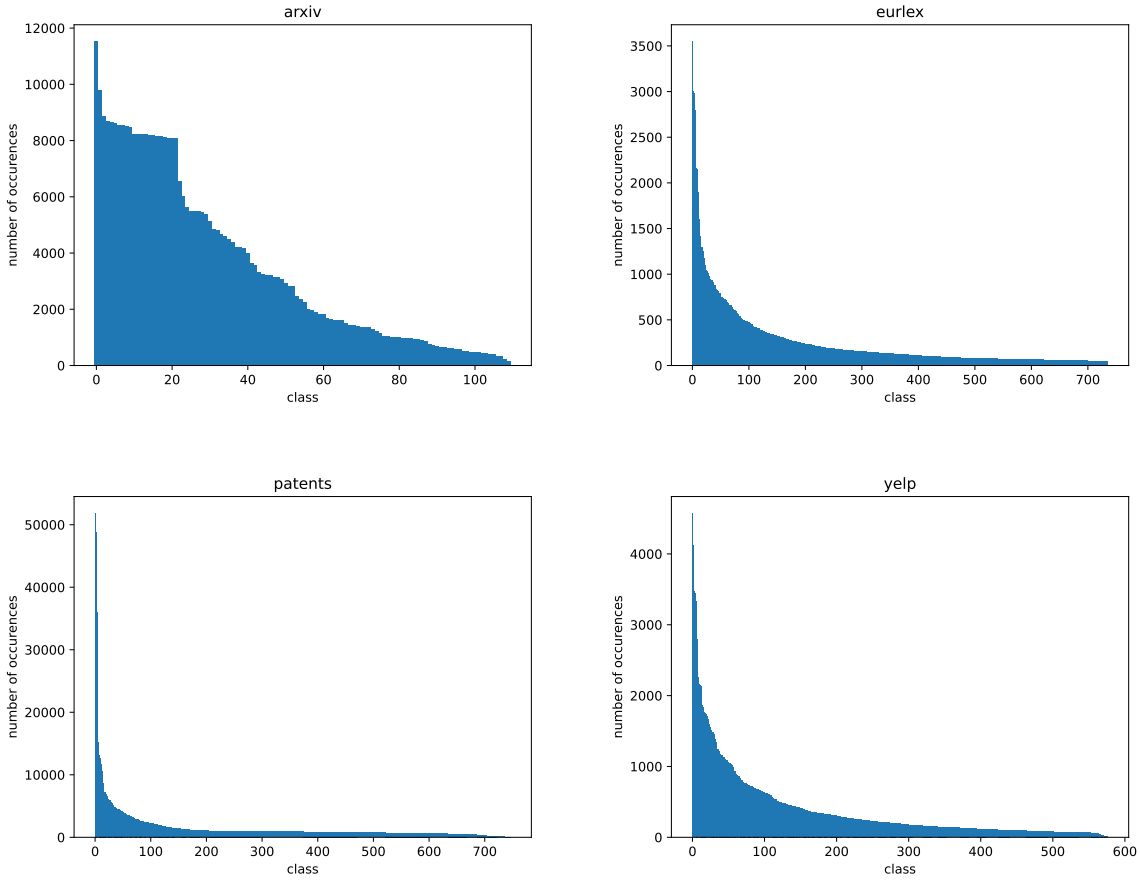

Figure 5: Distribution of classes on all datasets. Average number of classes assigned to a text are 1.7 on the *arxiv* dataset, 4.4 on the *eurlex* dataset, 5.9 on the *patents* dataset and 4.4 on the *yelp* dataset.

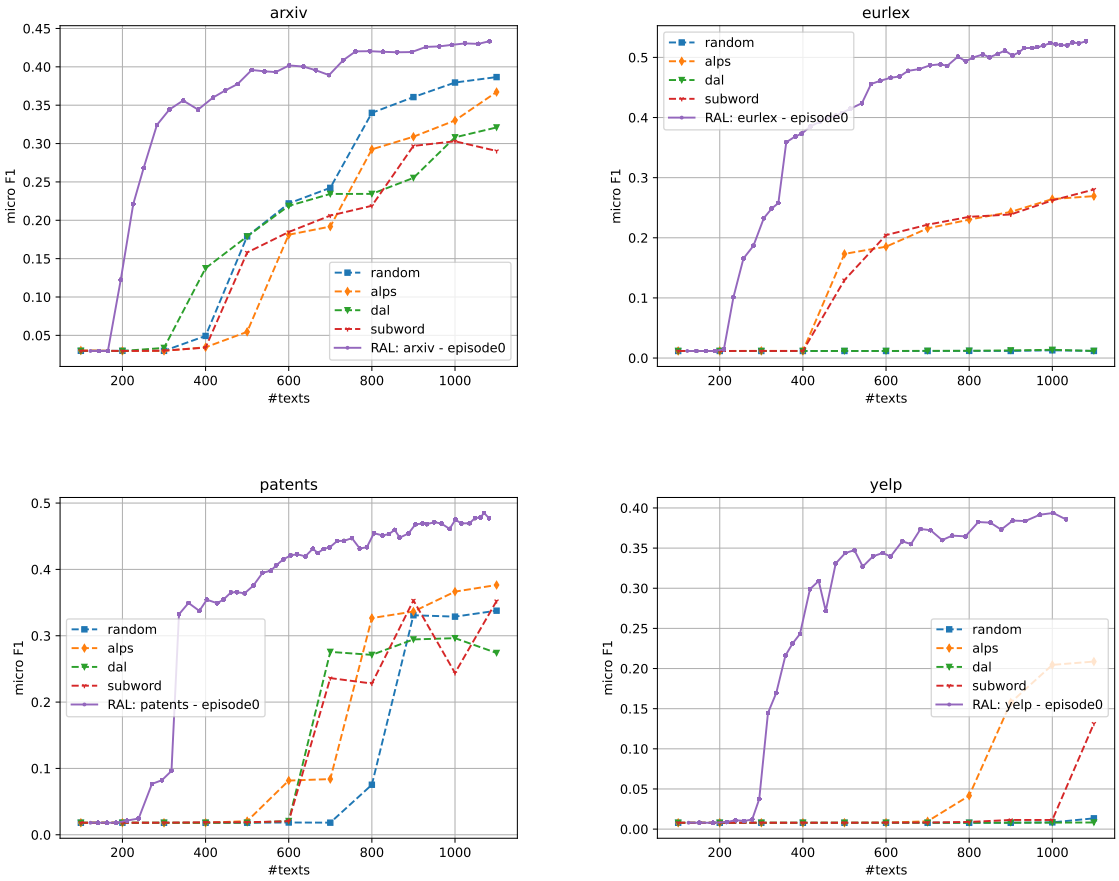

Figure 6: Micro-F1 for a single episode of RAL on full datasets.

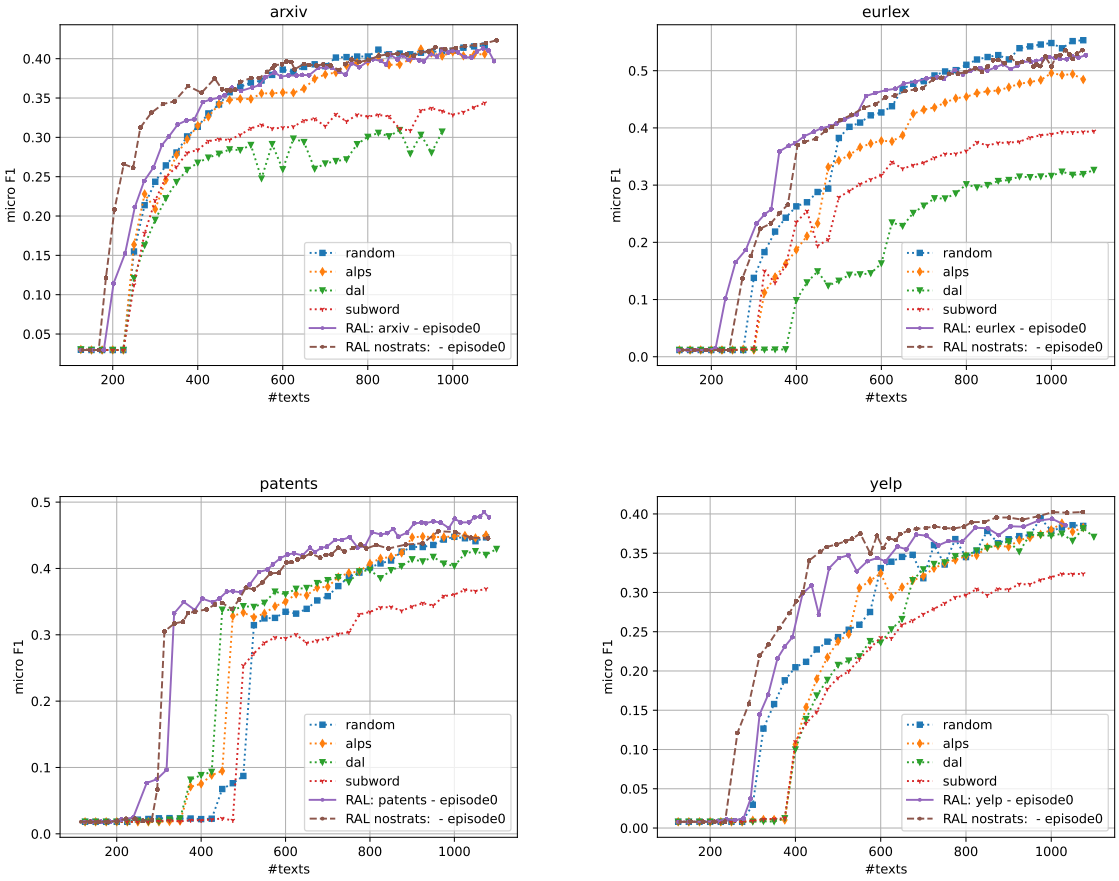

Figure 7: Micro-F1 for a single episode of RAL with and without using AL strategies on all datasets. AL baselines have been trained with an update size of 25.

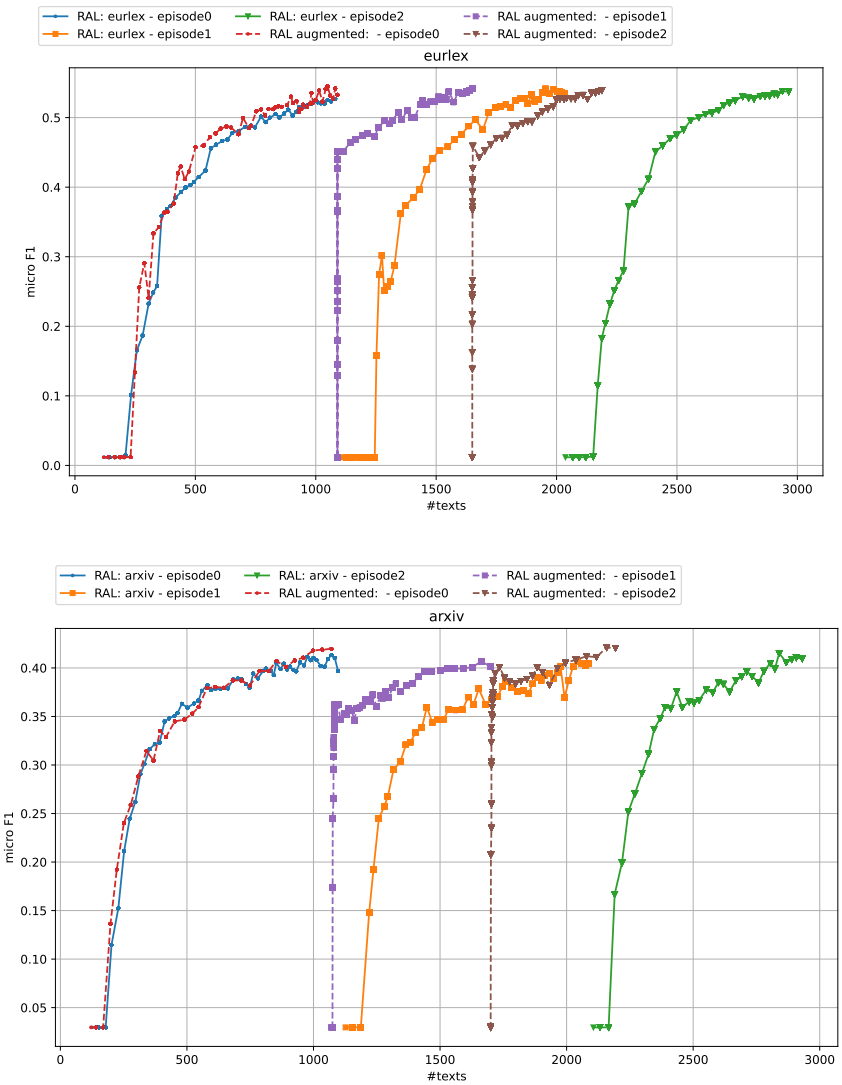

Figure 8: Micro-F1 for multiple episodes of RAL on the *eurlex* and *arxiv* datasets with and without augmentation. x-axis shows total number of annotated samples for a given F1 result.

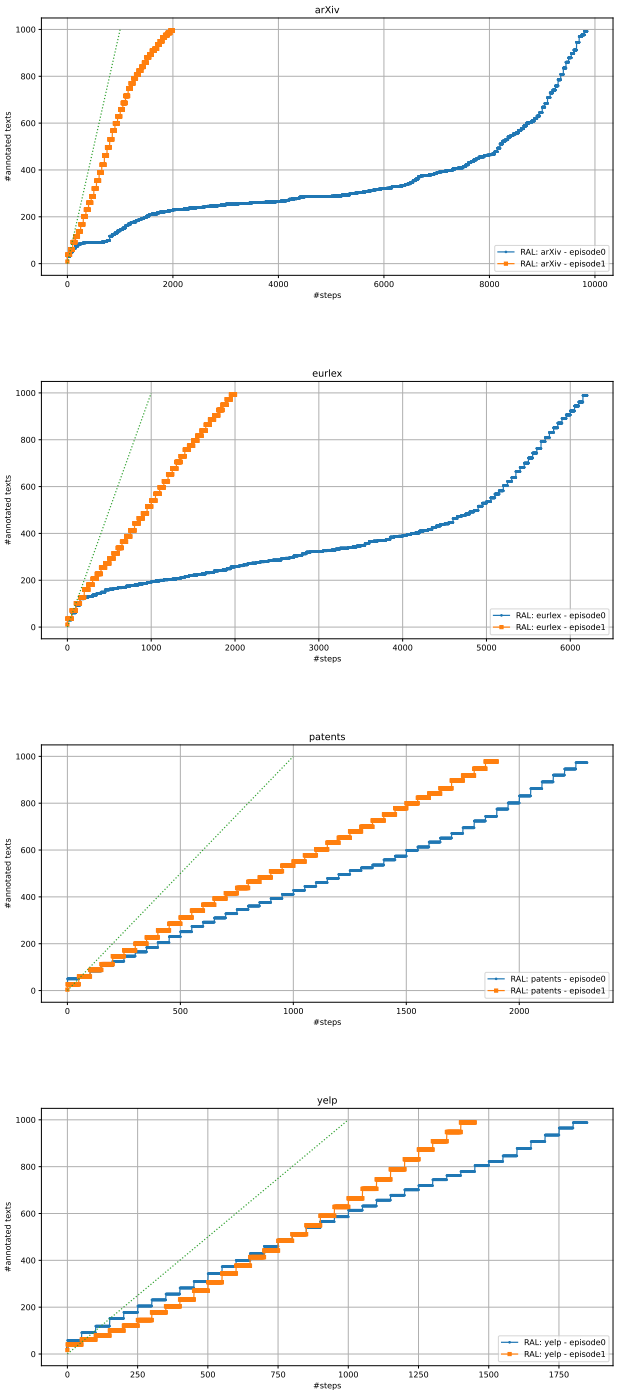

Figure 9: Number of steps (i.e. unlabeled documents seen) versus documents labeled. The dotted line represents the maximum of labeling a document each step.

Param	Value	Comment
Active Learning		
initial size	100	Size of the initial dataset for AL
budget	1000	How many samples will be labeled
AL batch size	50	Train the model after this many samples have been labeled
Text Classifier		
train epochs	15	Number of training epochs for the Text Classifier
BERT model	bert-base-uncased	using *transformers* library, max. sequence length 512 tokens
batch size	24	Number of samples used in each training step
learning rate	5e-5	Using ADAM optimizer
Reinforcement Learning (DeepQ Learning)		
GAMMA	0.99	Decay for future rewards
ALPHA	5e-4	Learning rate for policy update, using ADAM
EPSILON start	1.0	The probability of choosing a random action.
EPSILON end	0.2	The minimum value for EPSILON
EPSILON DECAY	2500	The number of steps needed for EPSILON to go from start to end.
TARGET UPDATE FREQ	50	Update the target net with the online net after this many steps
RL batch size	50	Update policy after this many steps
DQ batch size	32	The number of transitions used in each step of the policy training.

Table 3: List of hyperparameters for RAL. *Value* lists the value used for the experiments.