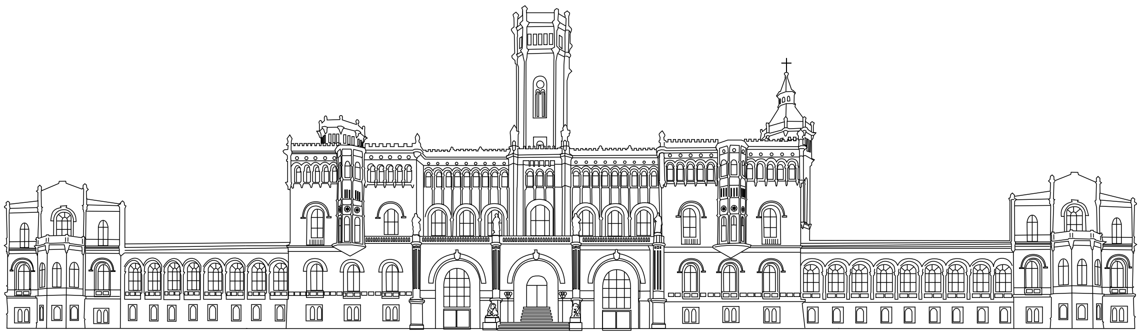
Gottfried Wilhelm Leibniz Universität Hannover
Institut für Verteilte Systeme
Fachgebiet Wissensbasierte Systeme

Master Thesis
Computer Science (M. Sc.)

Exploiting Rationale Data for Explainable NLP Models

Author:	Maximilian Jörg Reimer
First Examiner:	Prof. Dr. Avishek Anand
Second Examiner:	Prof. Dr. Wolfgang Nejdl
Supervisor:	Prof. Dr. Avishek Anand
Date:	September 15, 2021



Abstract

In recent years, deep learning models have become very powerful – even outperforming humans on a variety of tasks. This enables more real-world applications, including also sensitive fields such as medical diagnoses or jurisdiction. Besides achieving sufficiently good performance, the requirement to justify and explain the models' decisions is becoming increasingly important.

This work aims to enable a broader application of a specific model class that is inherently interpretable, namely explain-then-predict models, by reducing the annotation cost of the explanations. We focus on the ExPred model as a representative of explain-then-predict models.

We investigate its dependency on rationale annotations, a special kind of explanation, through training using gradually fewer rationale-labeled instances. Furthermore, we explore different approaches that aim to reduce the number of human-labeled instances required during training, such as active learning and weak supervision.

Our results show that even with only a fraction of instances annotated with rationales from the original dataset, ExPred still achieves good performance (within 95% of the performance when using 100% annotation). Depending on the dataset, only a few thousand annotated rationales are required. Using weak supervision, this can be further reduced, at least in specific settings. On the Movie Reviews dataset, we achieve good performance with only 5% of the original rational labels. The tested off-the-shelf active learning methods do not provide any benefit over randomly selecting instances to label. However, the extensive behavior analysis enables the future design of active learning methods that are tailored to explain-then-predict models. We start by proposing an active learning method that outperforms the random baseline on the Movie Reviews dataset.

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden, alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind, und die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen hat.

Hannover, den 15. September 2021

Maximilian Jörg Reimer

Contents

List of Figures

List of Tables

Glossary

APLS *Active Learning by Processing Surprisal* a deep-learning-specific active learning method introduced in [YLB20].

Anchors Model-agnostic, post-hoc interpretability method, introduced in [RSG18]. For more details see subsubsection 2.1.2.1 "Post-hoc Models".

AuxLTC *Auxiliary Head Least True Certain* a ExPred-specific active learning approach introduced in this work. For more details see subsubsection 3.4.4.8 "Auxiliary Head Uncertainty as an Indicator for Hardness of Instances".

BADGE *Batch Active learning by Diverse Gradient Embeddings* a deep-learning-specific active learning method introduced in [Ash+19].

BALD *Bayesian Active Learning by Disagreement* active learning method introduced in [She+21]. For more details see subsection 3.4.2 "Experimental Setup".

BERT A pre-trained transformer encoder introduced in [Dev+19].

BERT-to-BERT A variation of the model Lehman et al. introduced in [DeY+20]. For more details see subsection 3.1.1 "Model Selection: Explain-then-Predict Models".

Bastings et al. The model introduced in [BAT19]. For more details see subsection 3.1.1 "Model Selection: Explain-then-Predict Models".

Core-Set A purely diversity-based active learning method introduced in [SS18]. For more details see subsection 2.3.5 "Deep Learning".

DAL *Discriminative Active Learning* a diversity-based active learning method introduced in [GS19]. For more details see subsection 2.3.5 "Deep Learning".

DistilBERT The light-weight version of BERT introduced in [San+20].

EGL *Expected Gradient Length* a deep-learning-specific active learning method introduced in [Hua+16]. For more details see subsection 2.3.5 "Deep Learning".

ERASER The ERASER Benchmark [DeY+20]. For more details see subsection 2.1.3 "Quantifying Interpretability: Eraser Benchmark".

ExPred The model introduced in [ZRA21]. For more details see section 3.1.1 "Predict and Explain, and then Predict again".

FRESH The model introduced in [Jai+20]. For more details see subsection 3.1.1 "Model Selection: Explain-then-Predict Models".

FWL *Fidelity Weighted Learning* a weak supervision framework introduced in [Deh+18]. For more details see subsection 2.2.1 “Fidelity Weighted Learning”.

GCS *Greedy Core-Set* a fast approximation of CORE-SET, introduced in [SS18]. For more details see subsection 2.3.5 “Deep Learning”.

GRU *Gate Recurrent Unit* a recurrent layer introduced in [Cho+14].

GloVe Word embeddings introduced in [PSM14].

HPXL *Highest Predicted Explanation Loss* an active learning method introduced in this work. For more details see subsubsection 3.4.4.4 “Weighting of Tokens”.

HXL *Highest Explanation Loss* an oracle-based active learning method introduced in this work and only used for analysis. For more details see subsubsection 3.4.4.1 “Do the instances differ in their utility?”.

HXLDD *Highest Explanation Loss Dummy Diversity* an extension of HXL to study the diversity issue. For more details see subsubsection 3.4.4.2 “Diversifying Oracles”.

Integrated Gradients Model-specific, post-hoc interpretability method and an extension to input gradients introduced in [STY17]. For more details see subsubsection 2.1.2.1 “Post-hoc Models”.

LC *Least Certain* a classical active learning method describe in [Mol20]. For more details see subsection 2.3.4 “Classical Approaches”.

LIME Model-agnostic, post-hoc interpretability method, introduced in [RSG16]. For more details see subsubsection 2.1.2.1 “Post-hoc Models”.

LSTM *Long Short-Term Memory* networks a recurrent unit, introduced in [HS97].

Lehman et al. The model introduced in [Leh+19]. For more details see subsection 3.1.1 “Model Selection: Explain-then-Predict Models”.

Lei et al. The rationale-supervised version of LEI ET AL. (U) introduced in [DeY+20]. For more details see subsection 3.1.1 “Model Selection: Explain-then-Predict Models”.

Lei et al. (u) The model introduced in [LBJ16]. In this work we employ the revised version introduced in [DeY+20] . For more details see subsection 3.1.1 “Model Selection: Explain-then-Predict Models”.

MCD *Monte-Carlo Dropout* a deep-learning-specific active learning method that uses dropout to estimate a ensemble, introduced in [GIG17]. For more details see subsection 2.3.5 “Deep Learning”.

MNLP *Mean Normalized Log-Probabilities* a formulation of LC for sequences-tagging tasks, introduced in [She+17]. For more details see subsection 3.4.2 “Experimental Setup”.

MNTLP *Mean Normalized True Log-Probability* an oracle-based active learning method introduced in this work and only used for analysis. For more details see subsubsection 3.4.4.1 “Do the instances differ in their utility?”.

PE *Perceptron Ensemble* active learning method introduced in [Ein+20]. For more details see subsection 2.3.5 “Deep Learning”.

REINFORCE A algorithm to train neural model containing stochastic units, which can be used as discrete intermediate layers[Wil92].

Random Random active learning baseline that selects each batch randomly.

SP-LIME The an extension to LIME.

SmoothGrad Model-specific, post-hoc interpretability method and an extension to input gradients introduced in [Smi+17]. For more details see subsubsection 2.1.2.1 “Post-hoc Models”.

Snorkel A weak supervision framework introduced in [Rat+17b]. For more details see section 2.2 “Reducing Labeling Cost”.

VR *Variation Ratio* a query-by-disagreement active learning method introduced in [Bel+18]. For more details see subsection 3.4.2 “Experimental Setup” and subsection 2.3.4 “Classical Approaches”.

BoolQ Dataset from the eraser Benchmark. For more details see subsection 2.1.3 “Quantifying Interpretability: Eraser Benchmark”.

CoS-E Dataset from the eraser Benchmark. For more details see subsection 2.1.3 “Quantifying Interpretability: Eraser Benchmark”.

e-SNLI Dataset from the eraser Benchmark. For more details see subsection 2.1.3 “Quantifying Interpretability: Eraser Benchmark”.

Evidence Inference Dataset from the eraser Benchmark. For more details see subsection 2.1.3 “Quantifying Interpretability: Eraser Benchmark”.

FEVER Dataset from the eraser Benchmark. For more details see subsection 2.1.3 “Quantifying Interpretability: Eraser Benchmark”.

Movie Reviews Dataset from the eraser Benchmark used in our experiment. For more details see subsection 2.1.3 “Quantifying Interpretability: Eraser Benchmark” and subsection 3.1.3 “Datasets”.

MultiRC Dataset from the eraser Benchmark and used in our experiment. For more details see subsection 2.1.3 “Quantifying Interpretability: Eraser Benchmark” subsection 3.1.3 “Datasets”.

List of Symbols

L A labeled dataset used in the context of active learning and semi-supervision, see $\mathcal{D}$..

U Unlabeled dataset just containing the inputs $U = \{(x_i)\}_{i=1}^n$. or in case also the task labels but not the rationales $\mathbf{r}$ $U = \{(x_i, \mathbf{y}^*{}_i)\}_{i=1}^n$.

$\mathbb{E}$ The expected value.

$\mathbf{r}$ A rationale, a binary sequence $\mathbf{r} = (r_1, \ldots, r_n)$ of same length as the input sequence $\mathbf{x} = (x_1, \ldots, x_n)$, indicating which element x_i is relevant for solving the task.

$\mathbf{x}$ Input sequence $\mathbf{x} = (x_1, \ldots, x_n)$.

$\mathbf{y}^*$ The ground truth label.

$\mathbf{z}$ Some latent variable representation $\mathbf{z} = (z_1, \cdots, z_n)$ of the sequence $\mathbf{x}$.

$\mathcal{D}$ A dataset. Usually a set of tuples $\mathcal{D} = \{(x_i, \mathbf{y}^*{}_i)\}_{i=1}^n$ or in case of rationale producing models triples $\mathcal{D} = \{(x_i, \mathbf{y}^*{}_i, \mathbf{r}_i^*)\}_{i=1}^n$.

$\mathbb{1}$ The indicator function.

b Annotation budget i.e. the number of instances to be labeled. b_t is the budget at time t and $b_{t':t}$ the accumulated budget from time t' to t.

Chapter 1

Introduction

1.1 Motivation

In recent years, deep learning models have become the state of the art in various domains. In 2015, a model outperformed humans on ImageNet [Rus+15; He+16] for the first time. With the introduction of transformer models [Vas+17] and especially BERT [Dev+19] natural language processing, systems took another leap towards similar performance as humans [NB19]. This enables more and more real-world applications of these systems in sensitive fields such as medical diagnoses, jurisdiction, or social participation. But "with great power comes great responsibility" (attributed to Voltaire [Thi29]) and it has been shown that these models can reproduce social biases and display unwanted behaviors such as discrimination or semantically incorrect correlation [MPL19], which are hard to discover since these systems are black-box models.

This motivates the increased interest in making such models explainable. There are two broad categories of approaches to understanding why a model has come to a certain decision: *post-hoc* methods and *interpretability-by-design*. Post-hoc methods aim to analyze an already trained model in order to explain its prediction. In contrast, the interpretability-by-design scheme tries to build models that are inherently interpretable. Although post-hoc methods may seem appealing at first since the already existing high-performing models can be continued to be used, several methods have shown failure cases where they provide a plausible yet no causal explanation [JW19]. Thus, we focus on interpretability-by-design in this work.

More precisely, we will consider models that follow the *explain-then-predict* paradigm. This class of models first extracts a subset of the input serving as an explanation and then bases its final prediction only on this subset, therefore achieving faithful explanation by design. While there are models that do not require any additional training data for the explanation [LBJ16; BAT19; Jai+20], these do not achieve competitive performance to those who do [JW19; DeY+20; ZRA21]. Although acquiring rationale annotations alongside task labels poses not much more effort to some tasks [DeY+20], it may do to others, and there are already various datasets available with only task-level annotations. This work aims to reduce the requirement for human-annotated rationales. In particular, we investigate the effectiveness of active learning and weak supervision. Reducing the number of rational annotations needed can enable wider usage of interpretable models, leading to more reliable models, increased trust in to the models prediction and better human-model cooperation.

1.2 Problem Formulation and Research Questions

In this work, we aim to reduce the dependency on explanation annotations for explain-then-predict models. Put more formally, this means the following:

Given

(i) an explain-then-predict model $f(\mathbf{x}) = (\mathbf{y}^*, \mathbf{r})$ where $\mathbf{x}$ is the input sequence, $\mathbf{y}^*$ the task-level prediction and $\mathbf{r}$ the explanation also known as rationale. $\mathbf{r}$ is a binary sequence of the same length as the input sequence $\mathbf{x}$, where a 1 indicates that the token is important for the model's decision and 0 that it has no influence. Where $f(\cdot)$ requires annotation for the rationales to be trained. We will focus on ExPred [ZRA21] as a representative of this model class.

(ii) a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}^*{}_i, \mathbf{r_i})\}_{i=1}^{N} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$ with disjoint train, validation and test splits of sizes $N_{train}, N_{val}, N_{test}$ respectively.

(iii) further we divide $\mathcal{D}_{train}$ into two disjoint subsets L and U, where L is just some subset of $\mathcal{D}_{train}$ and U is the rest of the samples from $\mathcal{D}_{train}$, but without the explanation annotation ($U = \{(\mathbf{x}, \mathbf{y}^*) | (\mathbf{x}, \mathbf{y}^*, \mathbf{r}) \notin L\}$).

the goal is to find an approach that makes it possible to train f using only U and L and to reduce the size of L as much as possible while still achieving good task and explanation performance. Particularly, the performance should be better than that of unsupervised approaches such as Lei et al. (u) [LBJ16], which use $|L| = 0$. We call $|L| = b$ the *annotation budget*. We define good performance as 95% of the performance achieved using 100% supervision of the rationales (b=1).

We derive the following primary research questions:

RQ1 How much does the task and explanation performance for explain-then-predict models degenerate if fewer instances with rationale annotations are available at training time, i.e. if we reduce the size of L?

RQ2 Can we leverage weak labels, which are available almost for free, e.g. by using Lei et al. (u) [LBJ16] as weak labeler $\tilde{f}(\cdot)$?

RQ3 Given a fixed annotation budget b, how much can we improve test performance by selecting a specific L, i.e. acquiring rationals $\mathbf{r}$ for specifically useful instances, without knowing $\mathbf{r}$ beforehand, e.g. through active learning?

We emphasize that throughout the thesis we will assume that task-level annotations are always available for all instances. We only aim to reduce the number of rationale annotations. This is a practical scenario since there are a lot of task-level datasets available that lack explanation annotations.

1.3 Structure of the Thesis

In this chapter, we motivated this work and defined the problem and the research questions we are answering.

Chapter 2 gives an overview of the fundamentals and related works that are relevant to our research. This work builds on two pillars: Interpretable machine learning and approaches to reduce labeling cost for machine learning models. In section 2.1 “Interpretable Machine Learning” an introduction to the concepts of interpretable machine learning is given. Reducing the labeling cost (section 2.2) can be achieved in many ways including subsection 2.2.1 “Fidelity Weighted Learning” and section 2.3 “Active Learning”.

Corresponding to our three research questions, we perform three groups of experiments in chapter 3 “Experimental Evaluation”. For each of these groups, we give a short motivation, describe the experimental setup, and present the results. In order to make the description more compact, we describe the parts of the setup that are the same for all experiments in section 3.1. section 3.2 “Preliminary Analysis: The Baseline and its Need for Rational Supervision” answers **RQ1**, section 3.3 “Learning from Weakness: Semi-Supervision using Weak Labels” **RQ2** and section 3.4 “Active Learning” **RQ3**, respectively. For active learning, we additionally perform an extensive study on the behavior of the different approaches (subsection 3.4.4), which aims to provide insight on how to design improved active learning methods that are tailored to rationale extraction.

We conclude this work (chapter 4) by giving a résumé of our results in section 4.1. We are pointing out the limitations of our work as well interesting lines of future research in section 4.2

Chapter 2

Related Work

2.1 Interpretable Machine Learning

While machine learning models have become incredibly powerful, they have also become extremely complex, meaning that the reasoning behind a model's decision is in many cases not understandable for human beings, neither for the user nor the developer. In subsection 2.1.1 "Motivation and Definition" some scenarios will be highlighted in which explainability is a desired property of machine learning systems, and an intuitive definition for interpretability will be given. This is followed by subsection 2.1.2, in which the terminology interpretable machine learning is defined and available concepts are summarized. In the two last sections, we will focus on models specific to NLP, more precisely on how to measure interpretability (subsection 2.1.3 "Quantifying Interpretability: Eraser Benchmark") and on a model class that is interpretable by design (subsection 3.1.1 "Model Selection: Explain-then-Predict Models").

2.1.1 Motivation and Definition

There is neither a strict mathematical definition of interpretability nor a generally accepted informal one [Mol20], but there have been several attempts to specify it, e.g. "Interpretability is the degree to which an observer can understand the cause of a decision" [Mil18, p. 14] or "Interpretability is the degree to which a human can consistently predict the model's result." [KKK16]. If the second definition were accepted, a system could never be superior to a human.

Interpretability of machine learning systems has two beneficiaries: model developers and model users. In the latter case, imagine a system that predicts whether a patient is likely to have cancer based on X-ray images. Even if the overall model performance reported by the developer is better than a human expert, a doctor using such a system to support his/her diagnoses would probably not trust the model if he/she cannot obtain an explanation for the decision from the system as one would from a human member of a cancer board. An explanation, in this case, could be either in natural language describing which features of the input image have led to the prediction or simply some way of highlighting the parts of the image that were relevant for the decision. Using this additional information, the doctor can now make an informed decision about whether the model's forecast is plausible, and the system can even guide him/her to features indicating cancer that he/she might have overlooked at first. To sum up, better explainable models enable better collaboration between users and systems, improve certainty and help strengthen the user's trust in the system.

In addition to improving the operation of such systems, interpretable models or methods that make models interpretable also facilitate the developer's work. Deep learning models are incredibly good at picking up hidden patterns in the data, but they sometimes learn correlations that are not semantically correct or unwanted for other reasons. Understanding the individual prediction can help discover such undesired decision rules, which are not always revealed by the validation metrics. Carrying on the example from above, the training data could have been collected from databases of hospitals. One – bad – way of doing this would be to take all cancer positive samples from one and all negative samples from another hospital. This could lead to artifacts in the images caused by different hardware or configurations, which are much easier distinguishable for the model than actual symptoms of cancer. This can lead to a phenomenon called *right for the wrong reasons* [MPL19]. The model's predictions based on the dataset are correct, but using such semantically meaningless correlations for predictions based on new datasets would lead to massive misprediction since the new data points would most likely not fulfill the same conditions. Looking at the reasons for the model's decision can help to find such errors in the data as well as in the model [WGS20]. Additionally, questioning and understanding the model's "train of thought" can help to build better models in the future and enables us to learn from AI about the problems they are solving, e.g. in the cancer example one might discover new indicators for cancer.

In addition to these content-related advantages of interpretable models, there are also legal requirements for explaining and justifying a decision that has been made about a person [GF17], and there will probably be other similar laws in the future.

2.1.2 Categories of Interpretability Methods

The quality of an explanation is dependent on the question one wants to answer, e.g. what aspects of the model's behavior are of interest. Additionally, the methods are set apart by what kind of explanation is produced, e.g. highlighting important parts of the input vs. a textual explanation. The categorization of explanations we will look at is guided by [Mol20].

In [WGS20], current approaches are outlined and the following aspects are analyzed:

- The model's parameters: This provides insights about the learned structure of the model, e.g. representation probing.
- The model's input: This shows how the input influences the prediction, e.g. which parts of the input are most influential. This includes looking for (global) patterns in the model's decision process.

- The training data: This inspects the way the training samples influence the predictions, e.g. which training samples would change the current model's prediction if they were removed from the training data and the model was retrained with the resulting dataset.

When considering the model's predictions, one can distinguish between methods aiming at a global understanding of the model and those aiming at an explanation of individual predictions. Global explanations usually take the form of extracted decision rules from a trained model. But even simple if-this-then-that rules, which are often not strong enough to describe complex models globally, become quickly very unclear and therefore not understandable to a human because of the sheer number of rules. Thus, one typically looks at individual predictions, also known as local explanations, or at groups of predictions, e.g. the ones belonging to the same class. Examples of locally valid decision rules are anchors [RSG18] and triggers [Wal+21]. The question that is answered this way most of the time is: Which parts or features of the input were most influential for the prediction? A common alternative is to ask: Which parts of the input has to change or in what ways does the input have to change in order to change the model's prediction? Looking at a group of instances can reveal how the model separates one group from another. The last category of methods we are discussing tries to find data points in the training dataset that would, if removed, change the prediction in a specific test instance, e.g. [Yeh+18]. These methods are sometimes called data influence methods [WGS20] and can help find biases and errors in the dataset. Most local explanation methods – apart from data influence methods, which reveal the training instances that were most influential for a certain prediction – determine the degree of influence different parts of the input had on the model's predictions. This kind of local can be generated based on gradients [Smi+17; STY17; Sak20], attention scores [BCB16] or perturbations [RSG16; LMJ17].

So far, we have looked at the explanation itself. Another aspect that divides existing methods is whether they analyze an already trained model, in which case the methods are called post-hoc methods, or whether they build an intrinsically interpretable model. In the next section, we will look at specific examples from each of those categories that are commonly used in NLP.

2.1.2.1 Post-hoc Models

Post-hoc models can be divided into two kinds: the ones that are applicable independently of the model at hand, which are therefore called model-agnostic methods, and those that require models to have certain properties, e.g. access to gradients, which are known as model-dependent methods. In this section we will look at four methods, namely two representatives of each category. For a qualitative evaluation of most of the approaches see subsection 2.1.3 "Quantifying Interpretability: Eraser Benchmark".

The first method we consider is LIME, short for Local Interpretable Model-agnostic Explanations [RSG16]. It is a local, model-agnostic post-hoc method and hence applicable to any black-box model. The idea is to explain a single prediction (locally) to a human by approximating the complex black-box model f locally around the data point x with a simpler yet interpretable model $g \in G$, e.g. a decision tree or a linear model. The authors also provide an algorithm that aims to select a subset of instances that, when analyzed with LIME, explains the overall behavior of the model. This extension called SP-LIME is not discussed in this section. In order to create a training dataset $\mathcal{Z}_x$ for $g(\cdot)$, samples in the vicinity of x are created solely by perturbing x and $f(\cdot)$'s prediction. During training of $f(\cdot)$, the samples are weighted by the similarity of $\pi_x(z)$ to x, resulting in loss $\mathcal{L}(f, g, \pi_x) = \sum_{z \in \mathcal{Z}_x} \pi_(z)(f(z) - g(z))^2$. Even if the surrogate is conceptually interpretable, it can quickly become too complex to be comprehensible to a human, e.g. if the decision tree is too deep or the linear model has too many non-zero coefficients. The best explanation is therefore a model that minimizes the loss $\mathcal{L}(l, g, \pi_x)$ but also the surrogate's complexity $\Omega(g)$. Overall, we obtain the objective $\operatorname{argmin}_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$.

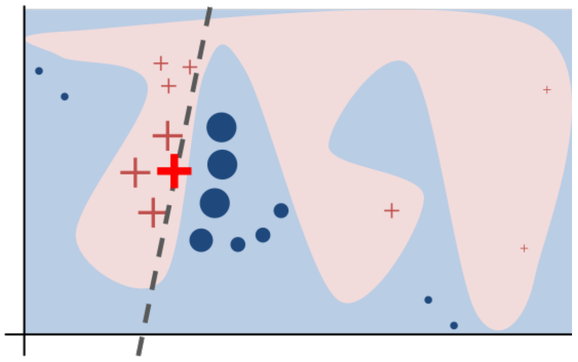


Figure 2.1: Concept of LIME in a two-feature example. The linear classifier (dashed line) serves as a local explanation for the big red cross and the blue circles, but it does not describe the full data space (background color). Source: [RSG16]

To wrap it up, LIME has three main components:

(i) A surrogate model g: In the paper, they used a sparse linear model achieved through lasso regularization.

(ii) A perturbation strategy to construct $\mathcal{Z}_x$ from x: For text tasks, the authors' choice is to randomly remove words from x.

(iii) A proximity function $\pi_x(z) = \exp(-D(x, z)^2/\sigma^2))$ with D being a distance function, e.g. cosine distance for text.

According to Molnar [Mol20, Chapter 5], these three main components are also the greatest weakness of LIME since choosing these hyperparameters is non-trivial and has a considerable influence on the resulting explanation.

There are several methods that try to generate locally valid decision rules around a given instance x which they have to apply in order to come to a certain decision. If we were able to find rules that can be applied globally, then we would probably not need a complex model. Anchors [RSG18], again a model-agnostic method and introduced by the same authors as LIME, are such an example. An anchor A is a set of predicates – in the paper it is always the presence of specific words, e.g. $A = \{\text{'not', 'bad'}\}$ – that returns true if all rules are fulfilled. Furthermore, A is a sufficient condition for a sample z from $D_x(z|A)$ to predict $f(z)$ if it predicts the same as $f(x)$ with a high probability. Hereby, $D_x(z|A)$ is the perturbation distribution around x of sample z where A applies. For example, if $x =$ 'This movie is not bad', then $A(x) = 1$ and the probability $p(f(x) = f(\text{'not bad'}))$ should be high. More formally, A is an anchor if $\mathbb{E}_{\mathcal{D}(z|A)}\mathbf{1}_{f(x)=f(z)} > \tau, A(x) = 1$. These anchors can be efficiently constructed using a greedy algorithm. Given a sample x and target precision τ (percentage of other examples that fulfill the anchor and lead to the same prediction), we first approximate D_x by perturbing x, which results in a set of samples $\mathcal{Z}_x$. In the case of language, this can be achieved by replacing words. We then start with candidates for A, which consist of the individual tokens of x and, in our example, are $\{\text{'This'}\}, \{\text{'movie'}\}, \{\text{'is'}\}, \{\text{'not'}\}, \{\text{'bad'}\}$. For each of the tokens, we estimate the precision of $\mathcal{Z}_x$ and take the best candidate. We then create all possible combinations of rules containing two tokens by using the token selected in the first iteration. We repeat these steps until we meet our τ.

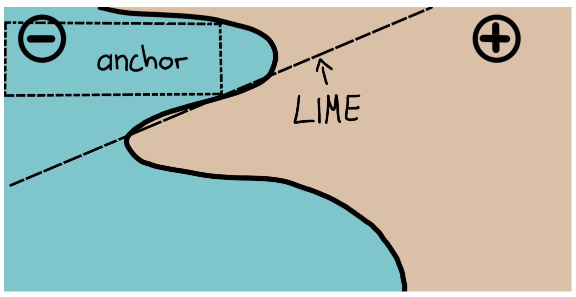


Figure 2.2: LIME vs. Anchors– A Toy Visualization. In contrast to LIME, Anchors find regions to which the rules apply. Source: [Mol20]

In contrast to LIME and Anchors, gradient-based methods are not model-agnostic since they require access to the gradients of the model. Gradient-based methods evaluate the importance of the input features by looking at the gradient of the predicted class i of classifier $f(x)$ with respect to the input x. The attribution map is therefore defined as $M(x) = \frac{\partial f(x)_i}{\partial x}$. This variant of gradient attributions are usually referred to as simple gradients. An alternative approach is to consider gradients w.r.t. the loss of the ground truth $\mathcal{L}(f(x), \mathbf{y}^*)$ or the predicted label $\mathcal{L}(f(x), \text{argmax}_i\, f(x)_i)$. The reasoning behind this is derived from the perspective that features have a big influence on the model's prediction if a small change in this feature results in a large change in the model's prediction. This naive approach is associated with several problems, thought, such as being too local and therefore sensitive to noise and not being able to cope well with the saturation of the activation functions in the network[WGS20]. There are several works that try to overcome the shortcomings of this vanilla method, e.g. SmoothGrad [Smi+17] and Integrated Gradients [STY17]. One has to point out that the feature attribution we are usually interested in is single value per feature. In the case of images, it is pixels, and in the case of text, words. Both kinds of features are often not represented by a single value because pixels consist of different channels and words of subword tokens. Therefore, some kind of

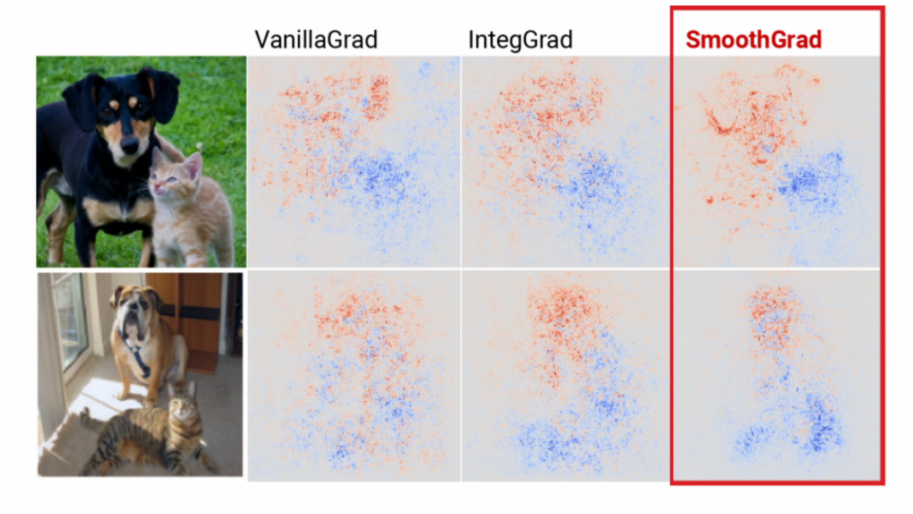


Figure 2.3: Exemplary result of vanilla input gradients (VanillaGrad), Integrated Gradients (IntegGrad) and SmoothGrad (SmoothGrad). Source: based on[Smi+17]

aggregation of the gradients for each of these scalars is needed. Common choices include the sum of the absolute values of the gradients for each component [WGS20] and the dot product of the gradient with the embedding. Another class of model-specific methods are attention-based ones [LBJ16; BCB16], which interpret the attention scores of the model as relevance scores of the features. For example, in the case of models that use a BERT encoder, the attention scores of the `[CLS]` tokens from the last or penultimate layer are usually aggregated to calculate the scores. Although this may seem plausible, it has been shown that attention scores are not very reliable [JW19]. This can be ascribed to the fact that the tokens to which attention is applied to are a mixed representation of the input tokens.

2.1.2.2 Interpretability by Design

In contrast to post-hoc methods, interpretability-by-design approaches create models that are at least partly comprehensible for humans. These approaches range from simply using inherently interpretable models such as sparse linear models, decision trees, and nearest neighbor classifiers [Mol20] over using such models on top of semantic features extracted using nontransparent models to only ensuring meaningful intermediate representations [Hig+16; Mar+21]. Incorporating explanations into the model and the training process allows for a additional kind of explanations for the model's decision, e.g. textual justification [Nar+20]. Although it has been shown that using explanations as a semantic regularization can boost the model's generalization performance [ZMW16; Raj+19], it does not necessarily results in causal explanations. This has been demonstrated in particular in the case of standard attention scores as indicators for the influence of individual features on the prediction [JW19]. Motivated by this, there are two lines of work: hard attention models, which are discussed in the section Model Selection: Explain-then-Predict Models, and models that use an opaque model to extract semantic features and use an interpretable model on top of it [Mar+21]. The latter is also called a semantic bottleneck.

In this work, we will focus on *explain-then-predict* models which provide an explanation in terms of *rationales*, a subset of the input sequence. They do so by first predicting the rationale $\mathbf{r}$ based on the input sequence $\mathbf{x}$ and then basing the actual task prediction only on the token contained in $\mathbf{r}$. They are therefore seen as hard attention models. More details can be found in subsection 3.1.1 "Model Selection: Explain-then-Predict Models".

2.1.3 Quantifying Interpretability: Eraser Benchmark

Since interpretability is a relatively nascent field in NLP, there are many lines of research seeking to provide desired properties and unified evaluation metrics. One of the most popular works on rationales is the ERASER benchmark [DeY+20], which will be used as a gold standard throughout this work. The benchmark revises and augments seven existing NLP datasets in order to provide a variety of tasks based on which models can be evaluated. Furthermore, metrics for measuring the accordance of rationale predictions with human annotations as well as metrics for assessing the faithfulness of the explanations were introduced.

The authors consider both hard and soft rationales for the selection of the tokens. We will focus mainly on the metrics defined for hard-selection of the input tokens since this kind of rationales is of major importance for this work. The hard explanation performance is measured by token-wise precision, recall, and derived F1 scores. In order to present a less rigorous metrics, partial matches via intersection over union using a threshold of 0.5 are introduced to calculate F1 scores. For soft-selection methods, Area Under the Precision-Recall Curve (AUPRC) is proposed as performance measure. In addition to measuring the conformity of the model's explanations to human annotations, also referred to as plausibility, another critical property of the predicted rationales is their faithfulness, meaning that the rationales should truly reflect the model's prediction. The authors call a rationale faithful if it is *sufficient* to make the task prediction and if it is also *comprehensive.* In order to be sufficient for the task level prediction $m(\cdot)_i$ of class i, $m(\cdot)_i$ should not change much when the prediction is only based on the tokens included in the rationale r instead of on the whole input sentence x. This property is therefore defined as sufficiency $= m(\mathbf{x})_i - m(\mathbf{r})_i$. Comprehensiveness measures how much information is in the non-rationale tokens that support the original prediction, leading to comprehensiveness $= m(\mathbf{x})_i - m(\mathbf{x} \setminus \mathbf{r})_i$. A faithful model achieves high comprehensiveness scores and low sufficiency scores.

The datasets consist of disjoint training, validation, and publicly available test sets. All datasets are based on datasets with task-level annotations and augmented with rationales. For datasets without comprehensive rationales, the authors acquired an additional test set containing comprehensive rationale annotations. Some of the datasets have been slightly changed, e.g. in CoS-E each instance has been converted from a multiple-choice question to multiple question-answer pairs for which the label is either `true` or `false` in accordance to the correctness of the answer. For more details on other tasks see the paper[DeY+20].

The datasets introduced in the paper are:

- Evidence Inference [Leh+19]: An NLI-task on full-text articles describing randomized controlled trials; the task is to predict whether a medical intervention improves over a comparator on a specific outcome.
- BoolQ [Cla+19]: A reading comprehension task with yes/no questions from Wikipedia.
- Movie Reviews [ZE08]: Sentiment analysis on movie reviews.
- FEVER [Tho+18]: The task is to predict whether a claim is supported or rejected by a text source.
- MultiRC [Kha+18]: A reading comprehension task in which the task is to predict whether a given reply answers a question.
- CoS-E [Raj+19]: A common-sense reasoning dataset.
- E-SNLI [Cam+18]: An NLI-task in which the model has to predict whether the given premise entails the hypothesis.

In the present work, we will focus on Movie Reviews and MultiRC. Therefore, more detailed descriptions of these datasets are given in section 3.1 “Common Experimental Settings”.

The authors evaluate multiple baseline interpretability methods for the task from above. For hard-selection models, they compare Lei et al. (u), Lehman et al., BERT-to-BERT and the supervised version of Lei et al. (u) introduced in this paper (for more details see subsection 3.1.1 “Model Selection: Explain-then-Predict Models”). For soft-selection, they apply an LSTM [HS97] decoder using additive attention on top of BERT-encoder [Dev+19] or GloVe [PSM14] vectors in the case of Evidence Inference and BoolQ in order to overcome quadratic memory consumption of the self-attention layer. They compare the following soft-scoring methods: vanilla input gradients, attention weights and LIME (see subsubsection 2.1.2.1 “Post-hoc Models”).

Hard-selection models are regarded as faithful by construction and therefore only agreement of the predicted rationales with human annotators and not comprehensiveness and sufficiency has been reported. The main result is that the supervised version of Lei et al. outperforms the unsupervised version Lei et al. (u) in most cases, both in terms of task performance and token F1 scores. In general, a good task performance seems to be associated with a good explanation performance. For soft-selection models, the main outcome is twofold. Firstly, the findings for attention-based attribution confirm that soft-selection models tend to yield rationales that are in accordance with human annotations but are not faithful. Secondly, LIME has been demonstrated to be the most faithful approach on these tasks.

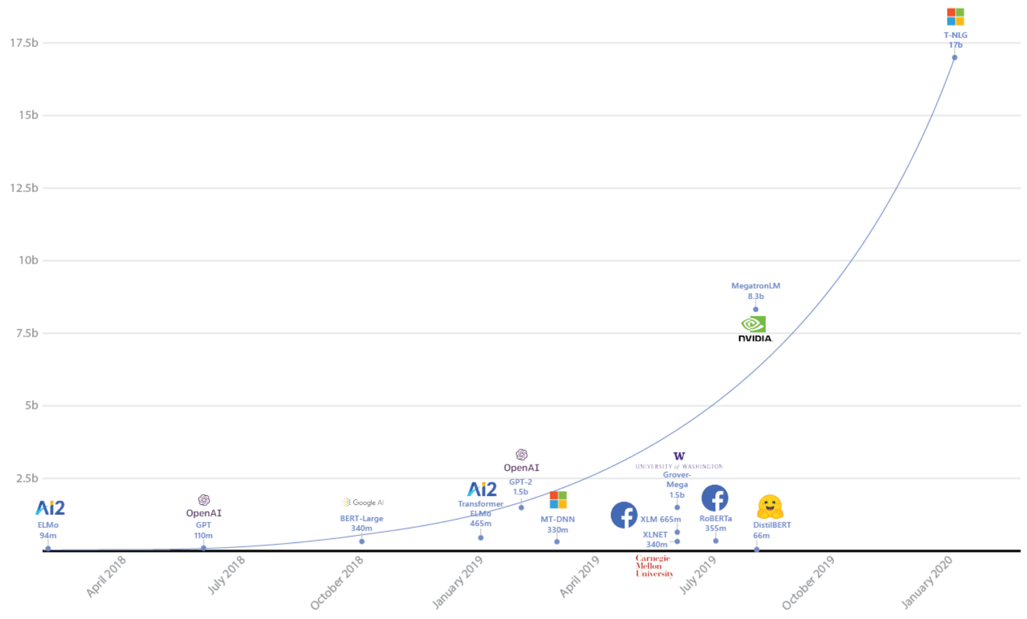


Figure 2.4: The trend towards larger language models. Source: [Ros20]

2.2 Reducing Labeling Cost

Most deep learning models require massive amounts of labeled training data annotated by *subject matter experts* (SMEs). This is especially an issue in NLP since language is sparse, which is why more data is required to cover the input space. This issue becomes more and more important due to the trend of building models with an increasing number of parameters (Figure 2.4).

[Rat+17a] provides a good overview of different techniques that reduce the labeling cost. The following list is guided by this work.

Active learning selects – given a labeling budget b, a small initially labeled dataset L_0, and a large unlabeled dataset U – those instances from U that, if labeled next by an oracle $o(\cdot)$, are estimated to be most helpful for the training of the model. $o(\cdot)$ is usually a human annotator. The labeling cost is set a fixed to budget b and the usefulness of the resulting dataset is optimized. Multiple formal objectives and other details can be found in section 2.3 "Active Learning".

Semi-supervised learning starts off with a small labeled dataset L_0 and a large unlabeled dataset U, but instead of labeling additional samples, it utilizes assumptions about smoothness or low dimensional structure in order to use unlabeled data, e.g. through unsupervised representation learning.

Transfer learning takes a model trained on a different task and applies it to the task at hand. A commonly used approach is to pre-train the model on an available, large, and relatively similar dataset and fine-tune it on a task of interest [Dev+19; Sha+14]. Another example is multi-task learning where a model is simultaneously optimized for different tasks using some shared representation. Lastly, there are some approaches that do not use the parameters of an already trained model, but its predictions to train another model, e.g. model distillation [San+20]. This can be considered a combination of transfer learning and weak supervision.

Weak supervision makes use of high-level annotations or noisy labels for training. More formally, in weak supervision we can use multiple weak supervision sources $j = 1 \dots m$ to label $U = \{x_1, \dots x_n\}$, resulting in weak datasets $L_{weak,1}, \dots L_{weak,m}$ where each $L_{weak,j}$ is associated with a coverage set C_j, which is the set of points $L_{weak,j}$ is defined over. For each $x_i \in C_j$, the probability of the label y_i is $p_j(y_i|x_i) < 1$, although it may also be unknown. Weak supervision approaches try to denoise the labels, to combine different weak sources or to model $p_j(y_i|x_i)$ in order to weight the samples during the training accordingly.

Forms of weak annotations may include:

Weak labels from sources like crowd workers, heuristic rules, distant supervision (labels constructed from existing databases), or predictions of other models.

Constrains which encode domain knowledge.

Distributions from which instances can be sampled, e.g. available through generative models.

Invariances like geometric transformations of images that probably retain their class. Based on this categorization, *data augmentation* can be seen as a form of weak supervision.

With Snorkel [Rat+17b], a weak supervision framework has been introduced that reduces the annotation cost by writing heuristic labeling functions also known as *data programming* instead of asking annotators for instance-wise labels. This leads to a massive annotation speed-up, but also reduces label quality. To mitigate this, the authors treat the labeling functions as multiple weak sources and model the accuracies without ground truth access by considering the overlaps and conflicts of the different rules. The results of the pipeline are probabilistic labels that the actual model is trained with.

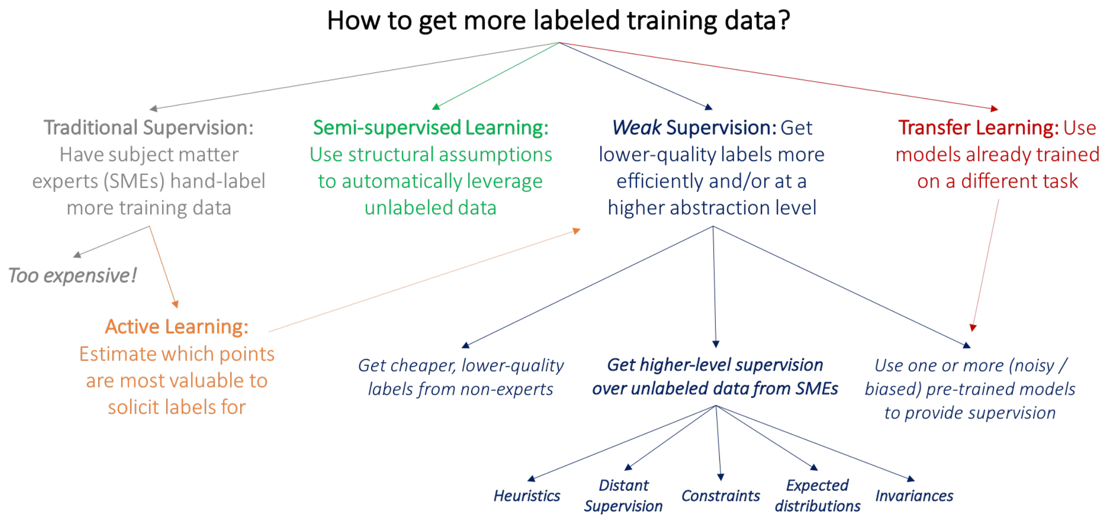

How to get more labeled training data?
Traditional Supervision: Have subject matter experts (SMEs) hand-label more training data
Semi-supervised Learning: Use structural assumptions to automatically leverage unlabeled data
Weak Supervision: Get lower-quality labels more efficiently and/or at a higher abstraction level
Transfer Learning: Use models already trained on a different task
Too expensive!
Active Learning: Estimate which points are most valuable to solicit labels for
Get cheaper, lower-quality labels from non-experts
Get higher-level supervision over unlabeled data from SMEs
Use one or more (noisy / biased) pre-trained models to provide supervision
Heuristics
Distant Supervision
Constraints
Expected distributions
Invariances

Figure 2.5: Overview of different approaches to reducing labeling cost. Source: [Rat+17a]

2.2.1 Fidelity Weighted Learning

A more simplistic approach than [Rat+17b] has been introduced in [Deh+18]. The objective is to train a model $f(x) = y$ as well as possible given

(i) a small strongly labeled dataset L_{strong}

(ii) a large unlabeled dataset U with $|L_{strong}| \ll |U|$

(iii) and access to some weak annotation function $\tilde{f}(x) = \tilde{y}$

Hereby $\tilde{y}$ is a noisy label for x. Since it is assumed that evaluating $\tilde{f}$ is cheap, it can then be used to create a weak dataset $L_{weak} = \{(x, \tilde{f}(x))|x \in U\}$ with negligible cost. Depending on what one considers to be the input to the approach ($\tilde{f}(x)$ or L_{weak}), this approach can be seen as a semi-supervised or weakly-supervised approach.

The authors argue that treating the weak labels or even all labels including the strong ones as equal ignores the different fidelities of the labels and therefore disregards information that could be used during training. In their approach, called *Fidelity Weighted Learning (FWL)*, a so-called soft dataset $L_{soft} = \{(x, \bar{y}, \Sigma)|x \in L_{weak} \cup L_{strong}\}$ where $\bar{y}$ is the soft label and Σ is the associated uncertainty estimate. To simplify, the model $f(\cdot)$ is then trained on this soft dataset by using the uncertainty estimates to weight the samples based on their estimated quality.

The method consists of three steps:

Firstly, pre-train the *student model* $f(x) = \phi(\psi(x))$ on the weak dataset L_{weak}. This results in a trained representation layer $\psi(x)$ that is in accordance with the weak labels and a task specific layer $\phi(\cdot)$.

Secondly, train the *teacher model* on the learned representations using the strong labels $\{(\psi(x), y)|(x, y) \in L_{strong}\}$. In the paper, they use a clustered Gaussian Process $\mathcal{GP}$ that provides good uncertainty estimates $\Sigma(x)$ and soft labels $T(x) = \bar{y}$ and creates $L_{soft} = \{(x, \bar{y}, \Sigma(x))|x \in L_{weak} \cup L_{strong}\}$. The intuition is that the teacher learns to correct the student's mistakes since it has to use the contaminated representations learned on the weak data.

Lastly, fine-tune the student on L_{soft} by using the soft labels $\bar{y}$ as targets and $\Sigma(x)$ to weight samples, i.e. scale the learning rate sample-wise accordingly.

The authors evaluate the framework on a toy task, on document ranking and sentiment classification, and they compare it to multiple baselines including:

- WA: The weak annotator $\tilde{f}(\cdot)$.
- NN$_{weak}$: The student trained on the weak data only.
- NN$_{strong}$: The student trained on the strong data only.
- NN$_{weak,strong^+}$: The student jointly trained on weak and strong annotations by oversampling the strong data.
- NN$_{weak \to strong}$: Here, transfer learning is used by pre-training the model with the weak data and fine-tuning it with the strong data.
- NN$_{weak^\omega \to strong}$: Pre-train on weak data and fine-tuning on strong data, but weighting all samples using a fixed value ω during pre-training, which results in a smaller pre-train learning rate. ω is estimated as the mean ω over all samples from FWL.
- FWL: The fidelity-weighted learning scheme described above.

The results show that FWL outperforms all baselines consistently across the different tasks. Among the simple baselines, NN$_{weak^\omega \to strong}$ is slightly ahead of NN$_{weak,strong^+}$ and NN$_{weak \to strong}$, which show no significant differences between them.

2.3 Active Learning

In this section we look at another technique to reduce labeling cost: active learning. We first describe the motivations to use this class of approaches (subsection 2.3.1 "Motivation and Definition"), continue by presenting different application scenarios (subsection 2.3.2 "Scenarios for Active Learning") and follow up with subsection 2.3.3 "Optimization Goals", in which we formally define different possible optimization goals for active learning. In subsection 2.3.4 "Classical Approaches", we give examples of active learning approaches optimizing for the different goals, and we conclude with discussing deep learning tailored methods (subsection 2.3.5 "Deep Learning")

2.3.1 Motivation and Definition

According to [Set12, p. xi], "Active Learning is the study of machine learning systems that improve by asking questions". The intuition is that a learner can learn better if it is able to actively explore the input space. This is analogous to humans learning a new language better and with less explanation (data) needed if he/she can ask for the meaning of specific words or phrases that he/she does not know about or is unsure about in certain contexts.

This becomes even clearer when looking at a 1D classification example. In this example, we assume that we want to learn to predict whether a certain person, Mia, would say that the weather is good or bad based on the hours of sunshine per day. We assume that there is one threshold $\tau^* \in [0\text{h}, 24\text{h}]$ that separates bad from good weather. In the classical supervised setup, we would now acquire a dataset $\mathcal{D}$ by presenting Mia with different hours of sunshine and asking her for each value whether she thinks that it is nice weather or not, e.g. by uniformly sampling $\mathcal{D} = \{(x, \text{MiaLikesTheWeather}(x)) | x = 0, 2, 4, \dots, 24\}$, where $\text{MiaLikesTheWeather}(x)) = x > \tau^*, \tau^* = 4$. We then end up with a classifier $\tau = x^+ + \frac{x^+ - x^-}{2} = 4 + \frac{5-4}{2} = 4.5$, where x^+ is the smallest positive value and x^- the biggest value in the negative class in $\mathcal{D}$ using a labeling budget of $b = 13$. In contrast to this classical approach, an active learner is able to ask Mia iteratively. We start, e.g. with the two boundary values $L_0 = \{(0,0), (24,1)\}$, which results in $\tau_0 = 12$. Step $t = 1$ and $t = 2$ result in $L_1 = \{(0,0), (12,1), (24,1)\}$ with $\tau_1 = 6$ and $L_1 = \{(0,0), (6,1), (12,1), (24,1)\}$ with $\tau_2 = 3$, respectively. The active learner performs a bisection search resulting at step 4 in $L_4 = \{(0,0), (3,0), (4.5,1), (6,1)(12,1), (24,1)\}$ and $\tau_4 = 3.75$ only using a budget of $b = 6$. Here, the active learner always chose the sample that was closest to the decision boundary τ_t of the classifier at time t as the sample to be labeled next.

The active learning setup usually involves the following components:

(i) A *model* or *learner* $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$ that requires labeled training data $L \subseteq \mathcal{X} \times \mathcal{Y}$ in order to be trained and that yields parameters: $\theta = \mathbf{train}(f, L)$.

(ii) Some source of unlabeled data $U \subseteq \mathcal{X}$ from which the model can choose samples. U may be a fixed set or it may change over time. More details are discussed in subsection 2.3.2 “Scenarios for Active Learning”.

(iii) An *oracle* $o : \mathcal{X} \mapsto \mathcal{Y}$ that provides access to the true labels, but is expensive to evaluate. In most cases, this will be a human expert.

(iv) An *annotation budget* b: The number of times $o(\cdot)$ can be asked to provide a label.

(v) An inital set of labeled samples L_0.

(vi) An initial parameter vector θ_{init}.

Algorithm 1 provides a high-level view on how most active learning algorithms work. Active learners differ mainly in their assumptions about U and the way they select new instances. θ_{init} is either set to $\theta_{init} = \theta_{t-1}$ or kept constant in order to avoid overfitting to the instances selected early on.

Algorithm 1: High-level view on the active learning framework

```
Input: $U$, train$(\cdot,\cdot)$, $o(\cdot)$, $b$, $L_0$
Output: $L_b$ and $\theta_b$
1 for $t = 1$ to $b$ do
2     $\theta_t = \mathbf{train}(f_{\theta_{init}}, L_{t-1})$
3     Select an instance $x_t$ from $U$ to label next
4     $L_t = L_{t-1} \cup \{(x_t, o(x_t))\}$
```

2.3.2 Scenarios for Active Learning

Active learning algorithms can be distinguished by three types of scenarios they assume to be given. These differ mainly in the nature of the source of unlabeled data [Set12, pp. 5].

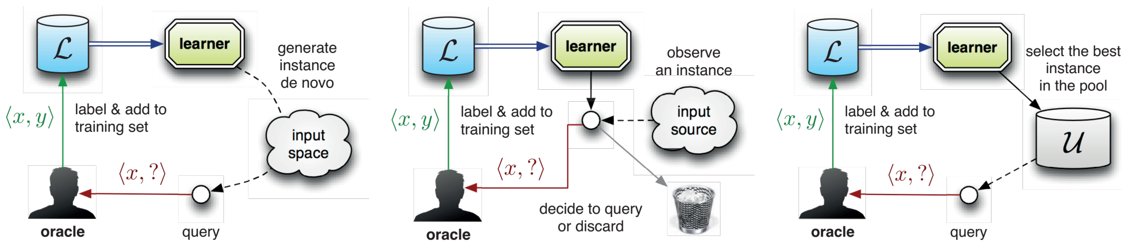
The first scenario is *query synthesis*, which entails that the active learner creates samples from the input space $\mathcal{X}$ that have to be labeled next. This allows the active learner to explore the input space, but may result in out-of-distribution examples if the input space is not well-defined. This is a common occurrence since defining the input space is often difficult in practice. Imagine a binary image classification task of the classes dogs and birds. One approach could be to ask for annotation of samples that the model is most unsure about. If the input space is well-defined, one could imagine that images of dogs in typical bird environments will be generated, e.g. a dog setting in a tree. But if the input space is ill-defined, generated instances could combine features differently, producing instances that lie between the two classes but are not realistic. This could result in hybrids of birds and dogs, e.g. a dog with wings. These unrealistic examples can be hard to label for a human and are therefore not helpful. There are works that try to overcome these problems by using generative models like variational autoencoders [SR19].

Another possible setting is *selective sampling*, in which the source of unlabeled data is a stream. The active learner only has access to one sample at a time and has to decide whether it should be labeled or not. But deciding whether a sample is informative enough without being able to compare it to others is a difficult task.

The most popular setup is *pool-based sampling*, which means that a large pool U of unlabeled examples is available. The active learner scores all samples in U and selects the most promising one based on an *acquisition function* or *utility function* $u : \mathcal{X} \mapsto \mathbb{R}$ that estimates the usefulness of labeling a sample (or a set of samples) and re-training the model on this new dataset $L' = L \cup \{(x, o(x))\}$. In many cases, u is actually dependent on the trained model $u(f_\theta, L)$. Note that in order for this method to be useful, evaluating $u(\cdot)$ must be much cheaper than actually labeling x and retraining the classifier. This is the most common scenario since nowadays we can easily crawl the web for most data types and find large collections of samples that can be used to retrain U. In this work, we will focus on the pool-based scenario since it is also the most common one in NLP.

Algorithm 2: Pool-based sampling of active learning

```
Input: $U_0$, train$(\cdot,\cdot)$, $o(\cdot)$, $b$, $u(\cdot)$, $L_0$
Output: $L_b$ and $\theta_b$
1 for $t = 1$ to $b$ do
2     $\theta_t = \textbf{train}(f_{\theta_{init}}, L_{t-1})$
3     $x_t = \text{argmax}_{x \in U}\, u(f_{\theta_t}, U_{t-1})$
4     $L_t = L_{t-1} \cup \{(x_t, o(x_t))\}$
5     $U_t = U_{t-1} \setminus \{x_t\}$
```



(a) Query Synthesis (b) Selective Sampling (c) Pool-based Sampling

Figure 2.6: Overview of Active Learning Scenarios. Source [Set12]

2.3.3 Optimization Goals

As discussed in the previous sections, active learning aims to select or generate the instances for being labeled next that are the most helpful. We will now look at different ways of formalizing this. The current section is strongly guided by [Set12, pp. 11, pp. 21, pp. 37, pp. 47] if not stated otherwise.

As we have already seen in the example in subsection 2.3.1 "Motivation and Definition" concerning the case of a 1D binary classifier, we can reduce the number of samples needed to achieve the same error rate quadratically by performing a binary search. To generalize to higher dimensions or multi-class settings, we can interpret the distance to the decision boundary as the certainty of the model's prediction. If the model is very certain about a prediction, it does not need the label for the instance since it is far away from the decision boundary and we assume that the model from the iteration before already provides some useful information. We, therefore, want to select the instance the model is unsure about. The family of utility functions used for this selection process is thus referred to as *uncertainty sampling*. If the model at hand provides a posterior distribution $P(y|x)$, we can formulate the objective as being the selection of those instances x for which the posterior distribution is closest to a uniform distribution. This makes this kind of utility function easily applicable to a wide range of models.

Whenever we train a model, the objective is to find a model f_θ from a hypothesis space $\mathcal{H}$ that describes our data distribution $\mathcal{D}$ as well as possible. For example, f_θ could be a neural network with parameters θ and $\mathcal{H}$ the set of all possible parameter values.

Usually, $\mathcal{D}$ is not fully available and therefore we try to find the model f_θ that fits a set of samples $\mathcal{D}_{train}$ from $\mathcal{D}$. Let $\mathcal{V} \subset \mathcal{H}$ be the *version space*, in which $\mathcal{V}$ is the set of possible models that are in accordance with our training data $\mathcal{D}_train$. By adding more instances to $\mathcal{D}_train$, the version space becomes – potentially – smaller. Therefore, we can formulate the active learning objective as selecting instances that reduce $\mathcal{V}$ the most and therefore diminish the classification error. This is known as *version space search*. Note that there are model classes for which version space search and uncertainty sampling are equivalent, e.g. for *support vector machines*. This kind of approach is limited to cases for which we can calculate or at least approximate the version spaces.

Another perspective on active learning is *minimizing expected error*. Here, we want to select the instances that, when added to the training set, reduce the future error or the error an unseen samples. This can be formulated as the expected error over all possible labels:

$$\begin{aligned} x^* &= \operatorname{argmin}_x \mathbb{E}_{Y|\theta,x} \left[\sum_{x' \in U} \mathbb{E}_{Y|\theta^+,x'} \mathcal{L}(y, \hat{y}) \right] && (2.1) \\ &= \operatorname{argmin}_x \sum_y p_\theta(y|x) \left[\sum_{x' \in U} \sum_{y'} p_{\theta^+}(y'|x') \mathcal{L}(y', \hat{y}) \right] && (2.2) \end{aligned}$$

Hereby, θ represents the model parameters resulting from training on L, and θ^+ represents the parameters resulting from re-training the model on $L \cup (x, y)$ after adding x from U and labeling it with labeled y'. $\hat{y}$ refers to the predicted class of θ^+. To calculate the utility function, we need to add each $x \in U$ with all possible labels to L and retrain the classifier. This makes the approach, although theoretically appealing, unpractical in most cases except for some model classes like Naive Bayes, Gaussian random fields, or nearest neighbor methods. There are related methods such as *variance reduction* which try to overcome this issue, but they are outside of the scope of this work.

In uncertainty sampling and version space search, each data point is considered individually and is not analyzed as part of the larger structure of the data space, which can lead to selecting outliers that are not representative of the distribution. This issue is addressed by *density-weighted methods* by adding a term to the utility function that measures the similarity to all other data points in U.

$$x^* = \operatorname{argmax}_x u(x) \cdot \left(\sum_{x' \in U} \phi(x, x') \right)^\beta \qquad (2.3)$$

Here, $u(\cdot)$ is some utility function, $\phi(x, x')$ is a similarity measure of x, and x' and β are hyperparameters trading off between similarity and utility.

The last set of approaches, *cluster-based methods*, tries only to optimize covering the input space well. The reasoning is that if we can choose a representative set of samples, this is sufficient. To do so, one needs a similarity measure in order to select diverse (dissimilar) instances. If the diversity measure is not dependent on the model, e.g. because a hidden layer of the network is used, the same methods can also be used to create the initial training dataset L_{init}. This is called warm starting. Alternatively, purely diversity-based methods can even be used as a single step selection method[Hua+16].

2.3.4 Classical Approaches

Having formalized different objectives for active learning, we will now present three classical examples for uncertainty sampling [Set12, pp. 14] and two for version space search[Set12, pp. 15 ff.]. Later, in subsection 2.3.5 "Deep Learning", we will also introduce a representative of the class of cluster-based methods.

The simplest uncertainty-based approach is *least confident* (LC):

$$x^* = \operatorname{argmax}_x 1 - P_\theta(\hat{y}|x) \qquad \text{with } \hat{y} = \operatorname{argmax}_y P_\theta(y|x) \tag{2.4}$$

This method selects the instances with the highest believe to being mislabeled while ignoring information about the rest of the distribution. The method *max-margin* utilizes the probabilities of the classes predicted to be the most likely $\hat{y}_1$ and the second most likely $\hat{y}_2$ to occur and judges instances as most important when the model cannot clearly discriminate between them:

$$x = \operatorname{argmax}_x (P_\theta(\hat{y_2}|x) - P_\theta(\hat{y_1}|x)) \tag{2.5}$$

Max-entropy is the most general and most common uncertainty method and takes the whole posterior distribution into account:

$$x^* = \operatorname{argmax}_x H_\theta(Y|X) = \operatorname{argmax}_x - \sum_y P_\theta(y|x) \log P_\theta(y|x) \tag{2.6}$$

This utility function is the most appropriate for non-binary classification settings because it can be interpreted as optimizing the expected log-loss, which is what we usually do in classification. For a binary classification setting, it does not matter which of the three methods is chosen, which is why many works also use uncertainty sampling [Ein+20].

Query by disagreement (QBD) is one of the earliest approaches from the realm of version space search. It works in the stream-based selective sampling scenario and selects an instance x to be labeled next when there are two models $h_1(\cdot)$ and $h_2(\cdot)$ in the current version space $\mathcal{V}$ that do not agree on x – put formally, if $\exists h_1, h_2 \in \mathcal{V} : h_1(x) \neq h_2(x)$. In most scenarios, this condition has to be approximated since $\mathcal{V}$ is directly available or too big to test all possible pairs of $h_1(\cdot)$ and $h_2(\cdot)$. For instance in the binary classification case, this can be done by maintaining two models: one that is the most specific to L $h_s(\cdot)$ and one that is the most general $h_g(\cdot)$. This can be achieved by sampling random points from the data distribution, creating two new datasets by adding them to L and labeling one of them as positive and the other as negative. $h_s(\cdot)$ is then trained on the first dataset and $h_g(\cdot)$ on the second one. The modified selection criterion is now: select x if $h_s(x) \neq h_g(x)$.

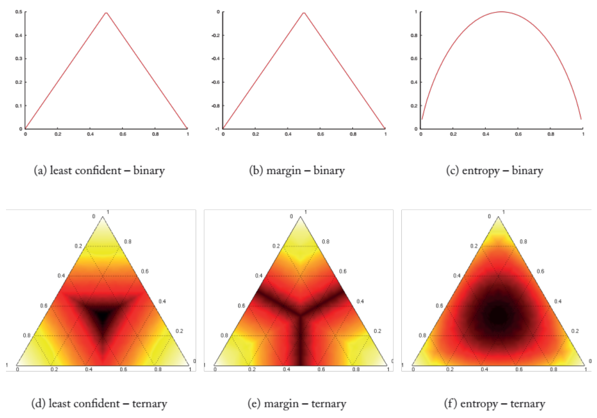


Figure 2.7: The three uncertainty-based utility functions for binary and ternary cases. Source: [Set12]

A crucial limitation that QBD exhibits is that it can only be easily applied in the binary case and in a stream-based setting. *Query by committee* (QBC) overcomes this constraint by using an ensemble or committee $\mathcal{C}$ to estimate the version space and by incorporating the information of how many of the models in the committee agree on an instance. One possible utility function is then *variation ratio* (VR)[Bel+18]:

$$x^* = \text{argmax}_x \, 1 - \frac{\text{vote}_{\mathcal{C}}(y, x)}{|\mathcal{C}|} \tag{2.7}$$

Here, $\text{vote}_{\mathcal{C}}(\hat{y}, x) = \sum_{\theta \in \mathcal{C}} \mathbb{1} y = h_\theta(x)$ and $\hat{y}$ is the class predicted by the ensemble. In another concept, the prediction of the committee is used as a Bayesian estimate of the uncertainty, which can be used to formulate alternative versions of the uncertainty-based utility functions. This allows their usage for models that do not provide good uncertainty estimates. For entropy, we can either obtain *hard vote entropy*:

$$x^* = \text{argmax}_x - \sum_y \frac{\text{vote}_{\mathcal{C}}(y, x)}{|\mathcal{C}|} \log \frac{\text{vote}_{\mathcal{C}}(y, x)}{|\mathcal{C}|} \tag{2.8}$$

or *soft vote entropy*:

$$x^* = \text{argmax}_x - \sum_y p_{\mathcal{C}}(y|x) \log p_{\mathcal{C}}(y|x) \tag{2.9}$$

Hereby, $p_{\mathcal{C}} = \frac{1}{|\mathcal{C}|} \sum_{\theta \in \mathcal{C}} p_\theta(y|x)$.

2.3.5 Deep Learning

Classical active learning algorithms are, for two reasons, not the perfect fit for deep learning:

Firstly, they operate with the method of selecting one instance at a time and training the model with the new dataset $L \cup (x, o(x))$. While this might be a realistic scenario for classical machine learning algorithms, which typically require relatively few training samples and are trained quickly, this is not feasible for deep neural networks. Training can take hours or even days, which makes it impossible to have a human annotator waiting for the training to finish. Also, adding a single instance does not improve the model's performance significantly due to the sheer number of samples needed. Therefore, deep learning models require active learning methods that select batches of samples instead of single instances [GS19; Ein+20]. Just taking the top-k samples based on one of the classical utility functions, such as Least Confident, bears the risk that very similar samples could be chosen, which are individually helpful but add redundant information to the labeled training set. Approaches like BADGE [Ash+19] and APLS [YLB20] try to trade off between uncertainty and diversity.

Secondly, the softmax activation is known to provide bad uncertainty estimates[Ein+20; GS19]. One attempt to overcome this is to use QBC methods. Since most of the modern neural networks use dropout layers, one natural way to simulate ensembles without the need to train multiple heavy-weighted networks is to run the instances through multiple forward passes with the dropout layers in training mode. This method is called MCD [GIG17] and has been shown to approximate Bayesian inference. Other methods such as perceptron ensembles (PE)[Ein+20] involve training multiple decoders on top of a shared encoder.

The acquisition function of (2.10) of Expected Gradient Length (EGL) [Hua+16] uses the gradient as an indicator for the change in the model parameters that is to be expected when adding a sample $x \in U$ with a label y to L instead of relying directly on the uncertainty estimates. This method is also motivated by variance reduction, which in itself is an implicit form of expected error reduction.

$$x^* = \text{argmax}_x \, \mathbb{E}_{Y|x,\theta} \left[\nabla\theta \mathcal{L}(x, y, \theta) \right] \tag{2.10}$$

Core-Set [SS18] and Discriminative Active Learning (DAL) [GS19] were created to introduce a batch setup. They solely optimize for sampling a representative subset of the data distribution. Due to only requiring access to some hidden representation layers but not the task-solving layer, those methods are easily transferable to other settings apart from the classification scenario, which is the most common setting in active learning.

A Core-Set of size B is defined as "B center points such that the largest distance between a data point and its nearest center is minimized" [SS18]. Here, the distance is measured in some representation space, usually some hidden layer of the network. Unfortunately, constructing such a Core-Set is NP-hard. In the paper, they propose a greedy algorithm that maximizes the criterion in each iteration and applies a refinement algorithm afterwards if estimated to be feasible. This first step is often applied on its own [Ein+20] and referred to as GCS.

Core-Set results in a set of points with roughly the same distance between any two points but ignores the density of the distribution, which leads to oversampling in sparse regions [GS19]. Furthermore, the DAL-authors argue that Core-Set does not scale to large datasets. They measure the usefulness of a sample as the difficulty to be classified to belong to U vs. L. This can also be interpreted as optimizing for the objective of making the distributions U and L indistinguishable and therefore selecting L to be representative of the whole data distribution, assuming U represents it well. Formally, the model is decomposed into $f : \mathcal{X} \rightarrow \mathcal{Y}$ and $f' \circ \psi$ with $\psi : \mathcal{X} \rightarrow \mathcal{Z}\prime$ and $\mathcal{Z}\prime \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is the input, $\mathcal{Y}$ the label and $\mathcal{Z}\prime$ a hidden representation space. In each active learning iteration, they train a discriminator $d : \mathcal{Z}\prime \rightarrow \{u, l\}$, which models $p(y = u|\psi(x))$ where u is the label for samples belonging to U and l the label for samples belonging to L, respectively. The final utility function is:

$$x^* = \text{argmin}_x \, p(\hat{y} = u|\psi(x)) \tag{2.11}$$

In order to acquire diverse batches, $d(\cdot)$ is retrained after selecting a sample and pseudo-labeling it as l. This suffices since the embedding $\psi(x)$ does not change considerably whether a single instance or several are labeled.

Chapter 3

Experimental Evaluation

In this chapter we are going to investigate our research questions (**RQ1-RQ3**, see section 1.2 “Problem Formulation and Research Questions”) through experiments.

In section 3.1 “Common Experimental Settings”, we describe the experimental details that stay the same throughout the analysis. This includes the choice of the model (subsection 3.1.1 “Model Selection: Explain-then-Predict Models”), the hyperparameters used during training (subsection 3.1.2 “Model Hyperparameters and Training”), and the datasets we apply the model to (subsection 3.1.3 “Datasets”).

We start our experiments in section 3.2 “Preliminary Analysis: The Baseline and its Need for Rational Supervision”, where we investigate how the rationale performance degrades when gradually decreasing the number of rationale annotations used during training. This lays the foundation for the following research and already answers **RQ1**.

Having established this baseline, we attempt to reduce the labeling budget even further by applying semi-supervision through the labels given by LEI ET AL. (U) (section 3.3 “Learning from Weakness: Semi-Supervision using Weak Labels”). To answer **RQ2**, we test a simple approach to determine if weak-supervision is promising line of research.

Finally, we turning to **RQ3**, explore different active learning approaches in section 2.3 “Active Learning”, and provide an extensive analysis on the behavior of the different approaches.

3.1 Common Experimental Settings

In this section an overview of the general experimental setup is given. First, we look at the choice of the model that we consider in the context of this work by analyzing the available candidates from the explain-then-predict class. We choose the explain-then-predict family in the first place because it is faithful by construction and exhibits state-of-the-art results.

3.1.1 Model Selection: Explain-then-Predict Models

In this section, we justify the selection of the model and lay out the details of its inner workings. Explain-then-predict models provide rationales and aim to be faithful by construction. They all follow the same structure: Given an input sequence $\mathbf{x} = (x_1, \ldots, x_l)$, they first apply the so-called *explanation generator* $g(\mathbf{x}) = \mathbf{r}$ to predict the *rationale* $\mathbf{r} = (r_1, \ldots, r_l) \in \{0, 1\}^n$, a vector indicating the subset of the input tokens $\mathbf{x}$ that are relevant to solve the task at hand. Then only the tokens included in the rationale are given to the actual classifier $f(\mathbf{x} \otimes \mathbf{r}) = y$, where $\otimes$ is a masking operator. For brevity we denote $\mathbf{x} \otimes \mathbf{r} = \widetilde{\mathbf{x}}$

The implementation of $g(\cdot)$, $f(\cdot)$, and $\otimes$ and the way $g(\cdot)$ and $f(\cdot)$ are being trained sets the methods apart.

$\otimes$ can either be implemented as replacing the tokens where $r_i = 0$ with a special `[MASK]` token retaining the structure of the input or it can simply delete the non-rationale tokens. Consequently for this sentiment analysis example

$$\text{('Even', 'though', 'it', 'was', 'sometimes', 'funny', 'I', 'did', 'not', 'like', 'the', 'movie')}$$
$$\otimes$$
$$(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)$$

evaluates to

(`[MASK]`, `[MASK]`, `[MASK]`, `[MASK]`, `[MASK]`, `[MASK]`, 'I', 'did', 'not', 'like', 'the', 'movie')

or

$$\text{('I', 'did', 'not', 'like', 'the', 'movie')}$$

.

Providing $f(\cdot)$, only the reduced version of $\mathbf{x}$ assures that the final prediction is based only on the tokens present in the rationale $\mathbf{r}$. Therefore, the explanation can be regarded as faithful.

The training of $g(\cdot)$ and $f(\cdot)$ can differ in two ways: On the one hand, they can either be trained jointly as end-to-end models or independently as pipeline models. On the other hand, some of the implementations require supervision on the rationales, others do not.

In Table 3.1, we present an overview of the approaches discussed in this chapter, including the discussed properties of the different models as well as exemplary performance on the datasets MultiRC and Movie Reviews, where available. Additionally, FULL INPUT is displayed which is a BERT-based model that gets the full-input and is equivalent to the second stage of ExPred. This serves as a reference for the model performance. Token F1 refers to the macro F1 on token matches of the rationale predictions, macro F1 to the classification performance of the task head. The results for FRESH are taken from the original paper and are not comparable since they use the original version of MultiRC and Movie Reviews and not the modified ones from ERASER benchmark. Furthermore,

Table 3.1: Approach comparison with regard to their properties and results are taken from[ZRA21]. † indicates degenerated runs due to REINFORCE. * results are taken from [Jai+20] and therefore not directly comparable.

Approach	Supvis. on r	E2E	MultiRC		Movie Reviews	
			Macro F1	Token F1	Macro F1	Token F1
FULL INPUT	-	-	0.708	-	0.894	-
ExPred	✓	✗	**0.698**	**0.640**	0.915	**0.348**
BERT-to-BERT	✓	✗	0.633	0.412	0.860	0.145
Lehman et al.	✓	✗	0.614	0.140	0.750	0.139
FRESH *	(✓)	✗	0.63-0.67	-	<u>0.93-0.94</u>	-
Lei et al.	✓	✓	0.665	0.331	0.914	0.285
Lei et al. (u)	✗	✓	0.648	†	**0.920**	0.322
Bastings et al.	✗	✓	-	-	-	-

they can not provide performance results on the rationales since the annotations were not available when the experiments were conduced. Overall, ExPred achieves the best results but requires supervision on the rationales, which makes it a perfect candidate for our investigations. In the following sections, we will discuss the model in more detail. In the end, we focus on ExPred since it is the model most relevant to this work. The fact that it provides state-of-the-art results and requires training data on the rationales make it the perfect fit for our research questions.

End-to-End Models The end-to-end models we are looking at, Lei et al. (u) [LBJ16] and Bastings et al. [BAT19], are trained jointly (expressed as $f(g(\mathbf{x}) \otimes \mathbf{x})$), reducing the need for an explicit supervision signal on rationales and using only task-level annotations plus some constraint on the selected rationales. This is not trivial since the hard selection $g(\mathbf{x}) \otimes \mathbf{x}$ is a non-differentiable operation.

In the original version of Lei et al. (u), the authors employ a unidirectional recurrent encoder to implement $f(\mathbf{x}) = p(y|\mathbf{x} \otimes \mathbf{r})$ in conjunction with a classification layer on top of the last hidden state and a bi-directional recurrent encoder with a shared classification layer at each time step to model $g(x) = p(\mathbf{r}|\mathbf{x})$. The objective to be minimized is

$$\text{cost}(\mathbf{r}, \mathbf{x}, y^*) = \mathcal{L}(\mathbf{r}, \mathbf{x}, y^*) + \Omega(\mathbf{r}) \tag{3.1}$$

where $\mathcal{L}(\mathbf{r}, \mathbf{x}, y^*)$ is simply the classification loss, here chosen to be

$$\mathcal{L}(\mathbf{r}, \mathbf{x}, y^*) = \|f(\tilde{\mathbf{x}}) - y^*\|_2^2 \tag{3.2}$$

and

$$\Omega(\mathbf{r}) = \lambda_1 \underbrace{\|\mathbf{r}\|_2}_{(*)} + \lambda_2 \underbrace{\sum_i |r_i - r_{i-1}|}_{(**)} \tag{3.3}$$

acts as a regularizer on the rationales. It enforces short $(*)$ and continuous $(**)$ selections. Optimizing this directly using some SGD variant would lead to empty rationales since the gradient signal from $\mathcal{L}$ to the explanation generator is always zero. To solve this issue, the excepted loss is optimized instead:

$$\min_{\theta_f, \theta_g} \sum_{(x,y^*) \in \mathcal{D}} \mathbb{E}_{\mathbf{r} \sim g(\mathbf{x})} [\text{cost}(\mathbf{r}, \mathbf{x}, \mathbf{y}^*)] \tag{3.4}$$

This is approximated using sampling, resulting in a REINFORCE-style algorithm [Wil92]. DeYoung et al. [DeY+20] replaced both encoders with BERT models in their implementation, followed by a LSTM and introduced a supervised version by simply adding the cross-entropy loss on the rationales as a third loss term. In this work we use Lei et al. (u) to refer to the improved, BERT-based version.

REINFORCE-style training exhibits unstable convergence, sometimes leading to degenerated results. To overcome this issue, Bastings, Aziz, and Titov [BAT19] utilize differential binary variables, which allow end-to-end training without the need for REINFORCE. Instead of predicting a discrete distribution (selected/unselected) for each token, they use a continuous distribution over the domain $[0, 1]$ that has almost all its density mass at 0 and 1, namely the rectified Kumaraswamy, and use the re-parameterization trick to predict the parameters of the distribution. They then sample from this distribution to get the rationale prediction for each token. This allows non-zero gradients. During training, the selection is not necessarily hard ($r_i \in]0, 1[$), but most predictions are almost 0 or 1, since most of the probability mass of the rectified Kumaraswamy distribution is at or close to 0 and 1. During the evaluation, the values are converted to hard selections by thresholding. They optimize a similar objective to [LBJ16] but introduce a target length for the rationales. In order to solve this constrained optimization problem, they use the Lagrangian relaxation.

Pipeline Models In contrast to end-to-end models, pipeline models train $g(\cdot)$ and $f(\cdot)$ independently which makes the training procedure much more simple.

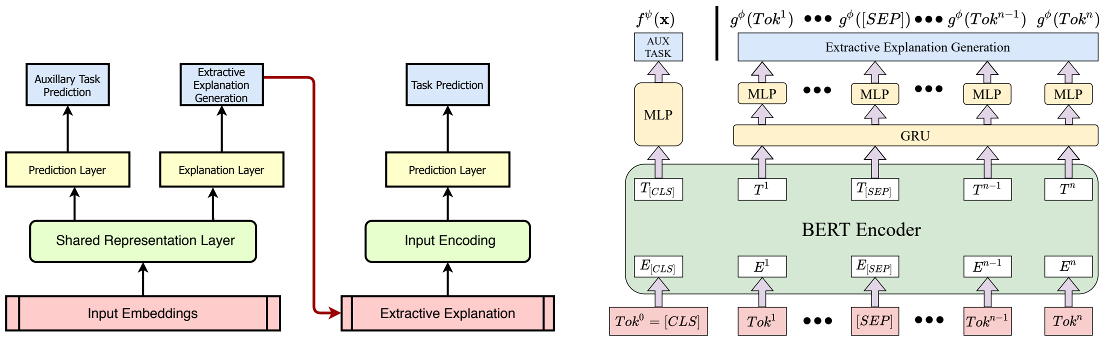
One of the earliest models was introduced by Lehman et al. [Leh+19]. In this work, multiple models were proposed. We will focus on the neural pipeline model which is referred to as Lehman et al. since it is the model that ensures faithful rationales and has been used regularly as a baseline [DeY+20; ZRA21]. $g(\cdot)$ is implemented as a sentence-wise independent classifier and trained using human supervision. It uses a GRU-encoder to get a vector representation of a sentence which is then concatenated with a prompt representation created by a bag-of-word model and passed to a classification layer to predict whether this sentence should be contained in the rationale. $f(\cdot)$ follows a similar architecture except that is trained only on the concatenated sentences that have been identified by the first stage. The ERASER paper [DeY+20] proposed a similar model where both stages are implemented using a BERT model, allowing better performance and token-wise instead of sentence-wise rationale selection. This model is called BERT-TO-BERT.

Jain et al. [Jai+20] approach the problem differently. To overcome the issue of post-hoc explanation methods not being faithful, they realized the so-called *support model* $g(\cdot)$ by first training on the target task, in their case different BERT variants depending on the tasks. The support model is then analyzed using post-hoc methods like simple gradients or attention scores to create continuous attribution scores for each token. In order to construct hard rationales, the attribution scores are discretized by one of two heuristics, either taking the top-k token or the sequence of k consecutive tokens which maximize the sum of scores. k is a hyperparameter. $f(\cdot)$ is realized, again, as a BERT variant following the same architecture as the support model but trained on the extracted rationales rather than the whole text. In order to allow mixing in human supervision on the rationales, they propose another version that trains an explicit extractor model on human annotations when given and on the pseudo labels created by one of the methods described above otherwise.

Predict and Explain, and then Predict again All of these pipeline models exhibit the same issue: the stage selecting the rationale ($g(\cdot)$) has very little information of the target task and relies mainly on the human annotations or the handcrafted heuristics to create the rationales. Zhang, Rudra, and Anand [ZRA21] try to overcome this issue using EXPRED, a multi-task learner (MTL), in the first stage which simultaneously optimizes the target task and the selection of rationales based on a shared representation. The second stage ($f(\cdot)$), again, only uses the predicted rationales for training.

More precisely, $g(\cdot)$ shares its BERT-encoder with an auxiliary task head $f'(\cdot)$ that solves the same task as the actual $f(\cdot)$ but uses a different set of parameters. The decoder of the generator is a unidirectional GRU followed by a shared token-wise classifier. This MTL is trained using the combined loss $\mathcal{L} = \mathcal{L}_{task} + \lambda\mathcal{L}_{exp}$, where $\mathcal{L}_{task}$ is the cross-entropy loss on $f'(\cdot)$ and $\mathcal{L}_{exp}$ is a weighted form of the cross-entropy loss on the tokens where weights are inverse of the prior probabilities for each class (3.5)[1]. λ is a hyperparameter that trades off rationale against auxiliary performance. To avoid propagating errors from the first into the second stage, only instances where the prediction of the auxiliary head matches the ground truth are used during training of the second stage.

[1]Note: The formula differs to the one in [ZRA21] due to a typo in the original paper but reflects the actual implementation.



(a) The overall structure.

(b) Architecture of MTL

Figure 3.1: ExPred model architecture. Source: [ZRA21]

$$
\begin{aligned}
\mathcal{L}_{exp} &= \frac{1}{|\mathbf{r}|}\sum_{i=1}^{|\mathbf{r}|} w_i \,\mathrm{BCE}(r_i, r_i^*) && (3.5)\\
&= \frac{1}{|\mathbf{r}|}\sum_{i=1}^{|\mathbf{r}|} \frac{|\mathbf{r}|}{|\mathbf{r}_{r_i^*}|}\mathrm{BCE}(r_i, r_i^*) && (3.6)\\
&= \sum_{i=1}^{|\mathbf{r}|} \frac{1}{|\mathbf{r}_{r_i^*}|}\mathrm{BCE}(r_i, r_i^*) && (3.7)
\end{aligned}
$$

where r_i^* is the ground truth rationale token at position i, $\mathrm{BCE}(\cdot,\cdot)$ is the binary cross-entropy and $|\mathbf{r}_{r_i^*}| = \begin{cases}|\mathbf{r}_1| & r_i^*=1\\ |\mathbf{r}_0| & r_i^*=0\end{cases}$, with $|\mathbf{r}_1| = \sum_{i=1}^{|\mathbf{r}|} r_i^*$ and $|\mathbf{r}_0| = |\mathbf{r}| - |\mathbf{r}_1|$. Therefore, $\frac{|\mathbf{r}|}{|\mathbf{r}_{r_i^*}|} = w_i$ corresponds to the inverse prior probability of token i to be r_i^* and the loss to the token-wise weighted mean BCE.

3.1.2 Model Hyperparameters and Training

In our experiment, we follow the model and training settings introduced by Zhang, Rudra, and Anand [ZRA21].

The models $g(\cdot)$ and $f(\cdot)$ use a standard BERT$_{base}$ uncased model as the encoder (12 layers, 768 hidden dimensions, 12 attention heads) along with the associated pre-trained tokenizer from the huggingfaces library [Wol+20]. In the case of $g(\cdot)$, the contextualized representations are then fed into a unidirectional GRU with 128 hidden dimensions. Each token is then individually classified to be part of the rationale or not through a classification layer with 256 hidden dimensions and a sigmoid non-linearity. The classification head used in both $f(\cdot)$ and $g(\cdot)$ is stacked on top of the representation of the `[CLS]`-token produced by the last encoder layer and employs two fully connected layers with a hidden dimensionality of 256 and output dimensionality 2 (for Movie Reviews and MultiRC). A dropout layer [Hin+12] with dropout probability 0.5 is placed before the first of the two layers, tanh is the non-linearity between the two, and a softmax follows after the last one. The classification and explanation heads are initialized using the Xavier initializer [GB10].

Sequences longer than 512 tokens are truncated and any tokens beyond that boundary are classified not to be part of the rationale.

During training, a batch size of 16, the Adam optimizer with a learning rate of $1e^{-5}$, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 1e^{-8}$, and weight decay of zero are used. Additionally, gradient clipping to 0.5 is applied. Learning rate scheduling is not applied.

The training for both stages ($f(\cdot)$ and $g(\cdot)$) is performed over ten epochs. Additionally, for $f(\cdot)$, early stopping is applied based on the validation macro F1 score with a patience of ten. For $g(\cdot)$, it has turned out to be helpful if it slightly overfits the rationale annotation since this leads to better rationales that are passed on to the second stage during training. Therefore, no early stopping is used during the training of $g(\cdot)$.

We use $\lambda = 5$ for Movie Reviews and $\lambda = 20$ for MultiRC as these values were found to be the best by Zhang, Rudra, and Anand [ZRA21] who utilized parameter sweeping.

The model is implemented using Pytorch 1.8.0+cu111 [Pas+19].

3.1.3 Datasets

We start our experiments on the same datasets used in the ExPred paper [ZRA21] but discard FEVER.

In FEVER, the task is to predict whether a given claim is supported or not by a text passage from Wikipedia. The instances follow the format `[CLS][claim][SEP][document]`. The rationales are only defined on the document, not on the claim. Results from Lei et al. (u) on FEVER from the ERASER paper show that even if no rationale is extracted, the task is still solvable with reasonable performance, which makes the dataset less interesting for rationale extraction. This fact and time constrains (the dataset consists of 97,957 instances) ruled FEVER out.

Movie Reviews contains movie reviews with associated sentiment labels `positive` or `negative`. The input format is `[CLS]["What is the sentiment of this review?"][SEP][review]`. The rationales are provided at token level and are relatively short as can be seen in Table 3.2.

MultiRC is a reading comprehension task. Given a string `[CLS][question]||[answer] [SEP][document]`: Predict whether the `answer` answers the `question` based on the document. Note that there are multiple instances containing the same question and document but different answers as this was originally a multiple choice question format. The rationales are only defined on the document and are on a sentence level.

Table 3.2: Dataset statistics. Source [DeY+20]

Datset	Documents	Instances	Rationale %	Evidences	Evidence Lengths
MultiRC					
Train	400	24029	17.4	56298	21.5
Val	56	3214	18.5	7498	22.8
Test	83	4848	-	-	-
Movie Reviews					
Train	1599	1600	9.35	13878	7.7
Val	150	150	7.45	1143.0	6.6
Test	200	200	-	-	-
FEVER					
Train	2915	97957	20.0	146856	31.3
Val	570	6122	21.6	8672	28.2
Test	614	6111	-	-	-

3.2 Preliminary Analysis: The Baseline and its Need for Rational Supervision

Table 3.1 compares unsupervised and supervised explain-then predict models. One can see that the supervision on the rationales improves the task and the explanation performance. To get a more detailed picture of the extent of the problem, experiments are conduced to figure out the performance loss if different amounts of the rationale annotations are available during training. Based on the results, we select a fixed annotation budget which is assumed to be given in the rest of the experiments.

3.2.1 Experimental Setup

We conduct our experiments on ExPred using the training and model details described in section 3.1 “Common Experimental Settings”.

For both datasets $\mathcal{D}$ = Movie Reviews and $\mathcal{D}$ = MultiRC, we randomly sample a fraction or budget $b \in [0, 1]$ of instances for which we keep the rationale annotation and remove them for the rest. This results in $\mathcal{D}' = U_{1-b} \cup L_b$, with $U_{1-b} = \{(\mathbf{x}_i, \mathbf{y^*}_i) | (\mathbf{x}_i, \mathbf{r}_i, \mathbf{y^*}_i) \in \mathcal{D} \wedge i \notin S_b\}$ and $L_b = \{(\mathbf{x}_i, \mathbf{y^*}_i, \mathbf{r}_i) | (\mathbf{x}_i, \mathbf{r}_i, \mathbf{y^*}_i) \in \mathcal{D} \wedge i \in S_b\}$ where S_b is the sampled index set. For each fraction, we sample two different subsets and train the model for each of these modified datasets twice using two random seeds to account for variation in usefulness of the sample and in the training procedure, likewise. This is done for the training set only. Validation and test sets always contain 100% of the rationales.

In order to train ExPred using instances where no rationale annotations are available, we modify the loss $\mathcal{L} = \mathcal{L}_{task} + \lambda\mathcal{L}_{exp}$ to

$$\mathcal{L} = \mathcal{L}_{task} + \begin{cases} \lambda\mathcal{L}_{exp}, & \text{if evidence is available} \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

Since λ has been selected based on the full dataset, one could argue that the influence of the explanation loss $\mathcal{L}_{exp}$ decreases by a factor b. This is the case since the loss per batch changes from

$$\mathcal{L}_{batch} = \frac{1}{B}\sum_{i=1}^{B}\mathcal{L}_i \tag{3.9}$$
$$= \frac{1}{B}\sum_{i=1}^{B}\mathcal{L}_{cls_i} + \frac{1}{B}\sum_{i=1}^{B}\lambda\mathcal{L}_{exp_i} \tag{3.10}$$
$$= \mathcal{L}_{cls} + \mathcal{L}_{exp} \tag{3.11}$$
$$\tag{3.12}$$

on average, to

$$\mathcal{L}'_{batch} = \mathcal{L}_{cls} + b\mathcal{L}_{exp} \tag{3.13}$$

because the explanation loss is only non-zero in $b\%$ of all cases. Experiments showed that compensating for this fact by $\mathcal{L}'_{batch} = \mathcal{L}_{cls} + \frac{1}{b}b\mathcal{L}_{exp}$ had no effect so we stick to (3.8). B refers to the batch size.

We run the experiments for $b = 1, 0.9, \ldots 0.1$, resulting in $9 * 2 + 1$ runs per dataset and report the test macro F1-score on class prediction for the auxiliary head of $g(\cdot)$, the predictions of $f(\cdot)$, and the token F1-score on the explanation head. After inspecting the results, we added $b = 0.5, 0.01$ to investigate the performance in very low data regimes.

3.2.2 Results

Figure 3.2 depicts the results of this initial experiment. Generally speaking, the performance of the explanation head (token F1 $g(\cdot)$) decreases with fewer annotations available. But shape of the curves differs considerably between Movie Reviews and MultiRC. The macro F1 score of the auxiliary task head of $g(\cdot)$ slightly increases but remains relatively stable. This could hint at the fact that the explanation loss $\mathcal{L}_{exp}$ gains slightly more influence during training; as discussed above. The macro F1 of the overall model $f(\cdot)$ also stays mostly stable. This is somewhat surprising since one would expect the task performance to suffer from bad rationales, leading to less meaningful inputs to the second stage of the model.

Taking a closer look at the rationales performance, one can see that on Movie Reviews, the token F1 stays stable until reaching $b = 0.8$ at roughly 0.34 and drops almost linearly after that until it reaches a score of 0.25 at $b = 0.05$. In contrast, on MultiRC, the token performance stays around 0.64 until $b = 0.4$ and slowly decreases to 0.59 at $b = 0.05$, then drops massively to 0.525 at $b = 0.01$. Although this seems like a big difference at first, one has to remember that MultiRC is much bigger than Movie Reviews while the model size stays the same. Meaning, the absolute number of instances with rationale annotations differs massively. e.g. $b = 0.1$ is $0.1 \cdot 1600 = 160$ for Movie Reviews and $0.1 \cdot 24029 = 2402$ for MultiRC, which is still twice as much the whole Movie Reviews dataset. Another aspect is that the variance – here shown as one standard deviation – is bigger in the case of Movie Reviews. This aligns with the fact that the absolute number of instances is much lower in Movie Reviews.

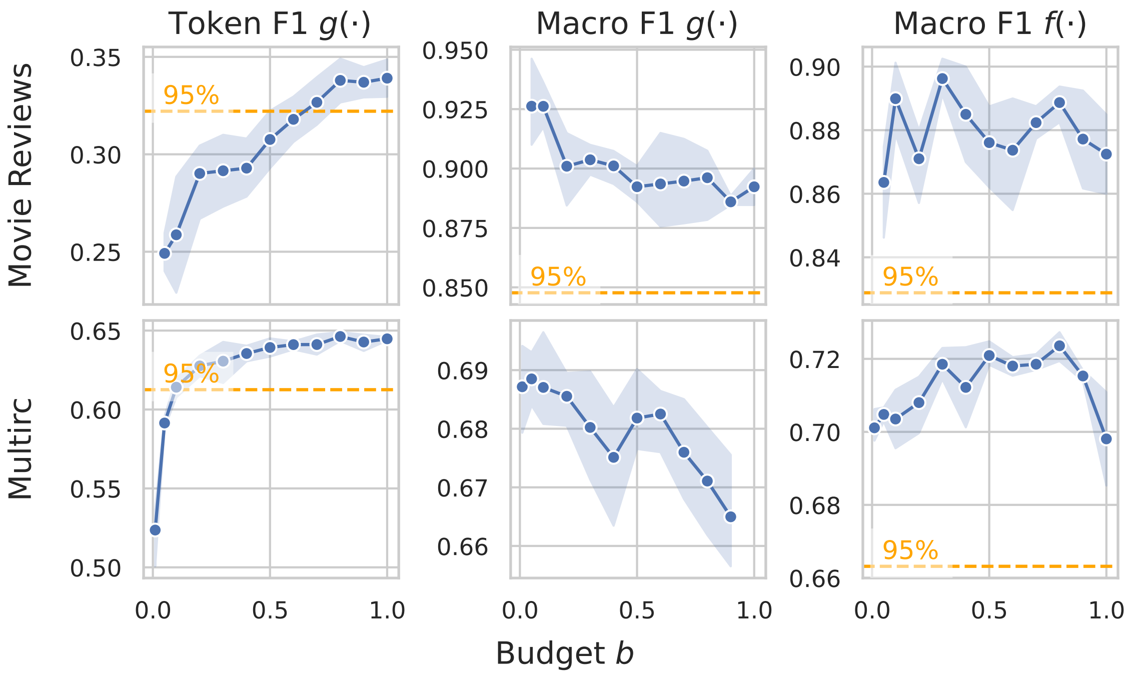


Figure 3.2: Test scores of ExPred if trained on different amounts of rationales.

The main results show that especially the token performance but also the other metrics stay within 95% of the performance at $b = 1$ down to $0.6 < b < 0.7$ ($\hat{=} 960 < b_{abs} < 1120$) for Movie Reviews and $0.1 < b < 0.2$ ($\hat{=} 2400 < b_{abs} < 4800$) for MultiRC, which might already be good enough for a lot of applications. This means that, depending on the task, a few thousand samples are sufficient. Therefore, we consider **RQ1** to be answered.

3.3 Learning from Weakness: Semi-Supervision using Weak Labels

Since solely reducing the amount of annotated rationales only works well enough (within 95% of performance at 100%) down to a certain point as shown in section 3.2 "Preliminary Analysis: The Baseline and its Need for Rational Supervision", one possible approach to tackle the issue of dropping performance is to apply semi-supervision or weak supervision techniques. This line of approaches is especially appealing since we acquire weak annotations from unsupervised models like FRESH, Lei et al. (u), or Bastings et al. almost for free.

Inspired by Dehghani et al. [Deh+18], we try to answer **RQ2** by investigating if semi-supervision using weak labels improves performance over just using strong labels. Since we want to establish if semi-supervision works and not necessarily aim to achieve state-of-the-art results, we choose the $\text{NN}_{weak \rightarrow strong}$ baseline from [Deh+18] as a starting point. This approach has low implantation cost and can serve as an indicator for deciding whether weak supervision is a promising direction. It is up to future research to investigate more sophisticated approaches like FWL.

3.3.1 Experimental Setup

The setting is as follows: given a labeling budget b, a large unlabeled data pool, more precisely with no annotations on the rationales, $U = \{(\mathbf{x}_i, \mathbf{y^*}_i)\}_{i=1}^{n}$, and a weak labeler $\tilde{g} : \mathbf{x} \rightarrow \mathbf{r}$ that produces noisy or low fidelity labels. In our case these labels are the rationale annotations. First, we randomly select an index set $|S_b| = b$ from which the corresponding instances are being labeled by an oracle $o(\cdot)$ that provides strong labels, resulting in $L_{strong,b} = \{(\mathbf{x}_i, o(\mathbf{x}_i), \mathbf{y^*}_i) | (\mathbf{x}_i, \mathbf{y^*}_i) \in U \wedge i \in S_b$. This is simulated by selecting instances from the original dataset (Movie Reviews or MultiRC). Then $\tilde{f}(\cdot)$ is utilized to create a weakly labeled dataset $L_{weak,1-b} = \{(\mathbf{x}_i, \tilde{g}(\mathbf{x}_i), \mathbf{y^*}_i) | (\mathbf{x}_i, \mathbf{y^*}_i) \in U \wedge i \notin S_b\}$. Notably, we only employ weak labels for the instances that were not strongly labeled in order to avoid conflicting information.

$\text{NN}_{weak \rightarrow strong}$ takes $L_{strong,b}$ and $L_{weak,1-b}$ and pre-trains $f(\cdot)$ on $L_{weak,1-b}$ and fine-tunes/trains on $L_{weak,1-b}$. Both training steps exactly follow the setting described in section 3.1 "Common Experimental Settings". We denote the dataset used for pre-training as $\mathcal{D}_{pre-train}$ and the one used for fine-tuning as $\mathcal{D}_{train}$. If no pre-training dataset is given $(\emptyset)$, this is simply equivalent to the standard training procedure.

To investigate if the model tends to forget information from the pre-training stage during fine-tuning, we run variants of the experiment where the BERT-encoder has been frozen after pre-training up to layer ten.

We use ExPred as $f(\cdot)$ and Lei et al. (u) as $\tilde{g}(\cdot)$ [DeY+20][2]. We run the experiments on Movie Reviews and MultiRC. We run the experiment for $b = 0.05$ and $b = 0.2$ since Movie Reviews already exhibits a huge performance drop at $b = 0.2$, but MultiRC stays stable until $b = 0.05$. We run each setting from two random seeds and report the average score as well as one standard deviation.

As a baseline, we include the performance of $f(\cdot)$ trained on $L_{strong,b}$ at the corresponding budgets as well as at $b = 1$. Additionally, we list the results of the weakly labeled Lei et al. (u) and ExPred trained on 100% weak labels $L_{weak,1-b}$.

[2]The implementation and parameters can be found `https://github.com/successar/Eraser-Benchmark-Baseline-Models`

3.3.2 Results

Table 3.3 and Figure 3.3 show the detailed results. First of all, we look at the performance of the weak labeler Lei et al. (u) in comparison to ExPred at 100% supervision. This marks the upper border we want to stay within 95% of. On both datasets, the task score on the actual task (Macro F1 $f(\cdot)$) is comparable among the different runs; roughly 0.81 for Movie Reviews and 0.7 for MultiRC. In contrast, the token F1 score does differ. In the case of Movie Reviews, it is slightly (0.44 vs. 0.31) and in case of MultiRC considerably lower (0.64 vs 0.2) for Lei et al. (u) than for ExPred. These results are to be expected since Lei et al. (u) is designed for sentiment analysis tasks where the rationales are usually short and fine-grained, i.e. on token level. Both properties do not apply in the case of MultiRC.

As a lower border, we report ExPred trained only on weak labels. For Movie Reviews, the token performance is, as one would expect, lower than the weak labeler itself. This is not the case for MultiRC, where the trained ExPred beats the performance of the weak labels by approximately at 0.04. This indicates that the weak labels exhibit symmetric noise around the true labels which can be smoothed out by ExPred. It is also remarkable that the task performance macro F1 ($f(\cdot)$) drops compared to 100% strong labels. This means that the rationales are not good enough for the second stage of ExPred to achieve good task performance.

The results of MultiRC at $b = 0.2$ are less interesting since even with only strong supervision, ExPred achieves strong results above 95% of the performance at $b = 1.0$, and does not improve with weak supervision. In fact, it decreases when the encoder is frozen during fine-tuning. Meaning that, it only achieves reasonable performance if most of the information learned during pre-training can be overwritten during fine-tuning. On Movie Reviews, we see a drop in token performance which can be mitigated using weak supervision in combination with freezing the BERT-encoder during fine-tuning, in which case we achieve comparable results to 100% supervision.

On a budget of $b = 0.05$ on Movie Reviews, we are still able to achieve performance, in particular token performance, above our target. On MultiRC, the performance on only strong labels is only slightly below this threshold but weak supervision neither helps nor damages performance, if the encoder is not frozen.

Overall, the results show that weak supervision can help to achieve comparable results to the performance achieved using 100% of annotation with only 20% strong labels and good performance with only 5% strong labels if the weak labels are of high quality. If the performance of the weak labeler is considerably lower than the results of the model using only strong data during training, it hurts the performance. Freezing the encoder amplifies the effect, since it helps to preserve information learned during pre-training. This effect is much stronger for the token F1 but also visible in the other metrics.

Using a more sophisticated weak supervision approach like FWL, where labels with estimated low quality are weighted less during training, might help to alleviate this issue for MultiRC.

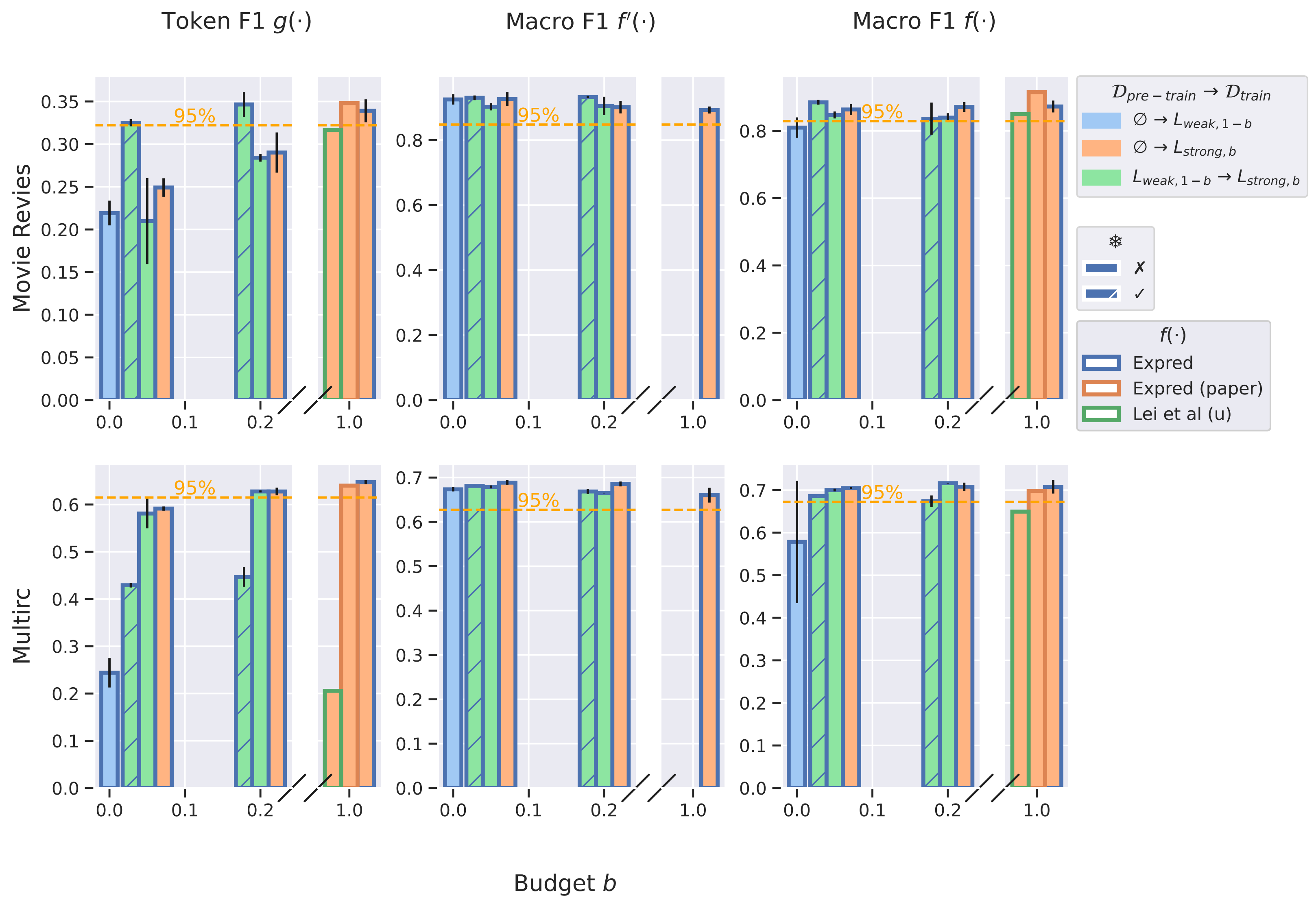


Figure 3.3: Weak supervision results on Movie Reviews and MultiRC. $\emptyset$ implies that the stage is not executed at all. ❄ indicates that the BERT-encoder is frozen during fine-tuning.

Table 3.3: Weak supervision results on Movie Reviews and MultiRC. $\emptyset$ implies that the stage is not executed at all. ❄ represents if the BERT-encoder is frozen during fine-tuning. The performance metrics follow the format 'mean (± standard deviation)'

$f(\cdot)$	$\mathcal{D}_{pre-train}$	$\mathcal{D}_{train}$	b	❄	Macro F1 $f(\cdot)$	Token F1 $g(\cdot)$	Macro F1 $f'(\cdot)$
MultiRC							
ExPred	$\emptyset$	$L_{strong,b}$	1.00	✗	0.708 (± 0.016)	0.647 (± 0.004)	0.660 (± 0.017)
ExPred (paper)	$\emptyset$	$L_{strong,b}$	1.00	✗	0.698	0.640	
Lei et al. (u)	$\emptyset$	$L_{strong,b}$	1.00	✗	0.649	0.206	
ExPred	$\emptyset$	$L_{weak,1-b}$	0.00	✗	0.578 (± 0.144)	0.244 (± 0.031)	0.673 (± 0.005)
ExPred	$\emptyset$	$L_{strong,b}$	0.20	✗	0.708 (± 0.01)	0.628 (± 0.008)	0.686 (± 0.006)
ExPred	$L_{weak,1-b}$	$L_{strong,b}$	0.20	✗	0.716 (± 0.001)	0.628 (± 0.002)	0.665 (± 0.001)
ExPred	$L_{weak,1-b}$	$L_{strong,b}$	0.20	✓	0.674 (± 0.013)	0.447 (± 0.021)	0.669 (± 0.006)
ExPred	$\emptyset$	$L_{strong,b}$	0.05	✗	0.705 (± 0.002)	0.591 (± 0.004)	0.689 (± 0.006)
ExPred	$L_{weak,1-b}$	$L_{strong,b}$	0.05	✗	0.700 (± 0.003)	0.581 (± 0.031)	0.679 (± 0.003)
ExPred	$L_{weak,1-b}$	$L_{strong,b}$	0.05	✓	0.686 (± 0.000)	0.429 (± 0.005)	0.681 (± 0.000)
Movie Reviews							
ExPred	$\emptyset$	$L_{strong,b}$	1.00	✗	0.872 (± 0.018)	0.339 (± 0.013)	0.892 (± 0.011)
ExPred (paper)	$\emptyset$	$L_{strong,b}$	1.00	✗	0.915	0.348	
Lei et al. (u)	$\emptyset$	$L_{strong,b}$	1.00	✗	0.850	0.317	
ExPred	$\emptyset$	$L_{weak,1-b}$	0.00	✗	0.810 (± 0.030)	0.219 (± 0.014)	0.925 (± 0.016)
ExPred	$\emptyset$	$L_{strong,b}$	0.20	✗	0.871 (± 0.015)	0.290 (± 0.024)	0.901 (± 0.019)
ExPred	$L_{weak,1-b}$	$L_{strong,b}$	0.20	✗	0.839 (± 0.013)	0.284 (± 0.005)	0.905 (± 0.028)
ExPred	$L_{weak,1-b}$	$L_{strong,b}$	0.20	✓	0.836 (± 0.048)	0.347 (± 0.014)	0.932 (± 0.004)
ExPred	$\emptyset$	$L_{strong,b}$	0.05	✗	0.864 (± 0.016)	0.249 (± 0.011)	0.926 (± 0.021)
ExPred	$L_{weak,1-b}$	$L_{strong,b}$	0.05	✗	0.847 (± 0.010)	0.210 (± 0.050)	0.902 (± 0.011)
ExPred	$L_{weak,1-b}$	$L_{strong,b}$	0.05	✓	0.885 (± 0.007)	0.325 (± 0.004)	0.930 (± 0.007)

3.4 Active Learning

Active learning is a common technique to reduce the annotation cost. In this section we investigate **RQ3**, namely "Given a fixed annotation budget b, how much can we improve by selecting a specific L without knowing $\mathbf{r}$ beforehand, e.g. through active learning?". Again we assume the task-level labels to be given and use active learning only to determine the instances for which we acquire the rationale annotations. We base our research mainly on three works: [Ein+20], [She+21], and [SC08], which will be shortly introduced in subsection 3.4.1 "Selection of Active Learning Methods". subsection 3.4.2 "Experimental Setup" lines out the implementation details and describes adaptations we had to make to transfer from a classification to our sequence tagging setup. Subsequently, in subsection 3.4.3 "Results", we first discuss the performance results followed by an extensive analysis on why active learning does not improve much in our setting as well as possible changes to tackle the arising issues (subsection 3.4.4 "Analysis: Reason for the Failure of Active Learning").

3.4.1 Selection of Active Learning Methods

Over the years, various active learning approaches have been introduced, which are often specifically designed to work well for one model class in combination with specific datasets.Lowell, Lipton, and Wallace [LLW19] points out that this poses a limitation on the practical usage of active learning because it is hard to know which method is going to work for new models or datasets. We attempt to overcome this by selecting approaches that have been successfully applied to similar models, here specifically BERT, and comparable tasks, in our case sequence tagging, if possible.

[Ein+20] performs an extensive study of comparing six different active learning approaches on BERT on ten different datasets. A limitation of this paper is that all considered datasets are binary classification tasks. They investigate a simple least confidence approach (LC), based on the uncertainty predicted by the model, two ensemble-based methods (MCD and PE), expected gradient length (EGL) and two purely diversity-based methods (GCS and DAL), which were introduced in subsection 2.3.5 "Deep Learning". The results show that all of the methods, with the exception of EGL and PE, consistently outperform the random baseline but none shows a significant advantage over the others.

In order to apply these methods to a sequence tagging task, some modifications have to be made. We use [SC08] as references, which provides an analysis of active learning strategies for sequence labeling tasks but on conditional random fields (CRFs) and not on BERT.

The most similar work to ours is [She+21], which examines four different active learning methods using different models including BERT on two sequence tagging datasets on the task of named entity recognition (NER). They compare one uncertainty-based approach *Maximum Normalized Log-Probability* (MNLP), which is an improved version of the least confident strategy for sequences with two approaches that use MCD to estimate an ensemble. More precisely, the authors apply *variation ratio* (VR) and *Bayesian active learning*

Table 3.4: Comparison of active learning approaches that have been applied to BERT. $p(\hat{y}|x)$ describes if the approach uses the probability of the predicted class, $p(y|x)$ if it makes use of the whole distribution.

Approach	Uses				App. to Seq. on BERT	Better than random
	$p(\hat{y}\|x)$	$p(y\|x)$	Diversity	MCD		
MNLP	✓	✗	✗	✗	BERT [She+21]	✓
VR	✗	✗	✗	✓	BERT [She+21]	✓
BALD	✗	✓	✗	✓	BERT [She+21]	✓
LC	✓	✗	✗	✗	as MNLP	✓
LC (MCD)	✓	✗	✗	✓	✗	✓
PE	✓	✗	✗	✗	✗	✗
EGL	✗	✓	✓	✗	✗	✗
DAL	✗	✗	✓	✗	✗	✓
GCS	✗	✗	✓	✗	✗	✓

by disagreement (BALD) which is especially appropriate for multi-class classification. The main result is that all methods work comparably well on BERT but outperform the random baseline. Additionally, they show that for strategies using MCD the performance degrades massively when only using the dropout layers of the classifier head. Another interesting insight is that for Transformer models, one can use more lightweight models like DistilBERT [San+20] as a surrogate model during active learning which leads to speed up but depending on the scenario sometimes too small performance drops of the final models.

Table 3.4 gives an overview of different properties of all methods that have been applied using BERT. EGL and PE are ruled out since they have not outperformed the random baseline in some cases. To cover a variety of aspects, we choose one diversity-based, one uncertainty-based, and one Monte-Carlo-dropout-based strategy and prefer approaches that have been applied to sequence tagging, leading to our final selection of MNLP, VR, and GCS. For more details on the approaches see section 2.3 "Active Learning" and especially subsection 2.3.5 "Deep Learning". GCS is chosen over DAL due to its lower implementation cost.

One can observe that the selection does not contain any strategy that trades off diversity against uncertainty like EGL, BADGE, or APLS would do. In this work we focus on methods that have already been successfully applied to BERT and it is up to future research to investigate the utility of these methods.

3.4.2 Experimental Setup

We apply active learning to ExPred on two datasets: Movie Reviews and MultiRC. We follow the training procedure described in section 3.1 "Common Experimental Settings".

We simulate the oracle $o(\cdot)$ by removing the rationale annotations from the fully labeled datasets and re-add them if selected by the active learner.

Notation In order to describe the implementation of the active learning approaches, we first define the following notation. As in previous sections, the first stage of the model, namely the explanation generator, is referred to as $g(\mathbf{x}) = p(\mathbf{r}|\mathbf{x})$ and maps from the input sequence $\mathbf{x}$ to the rationale sequence $\mathbf{r}$. It shares its BERT-encoder $\psi(\mathbf{x}) = \mathbf{z} = (z_{[CLS]}, z_1, \ldots, z_n)$ with the auxiliary head $f'(x) = p(y|\mathbf{x})$. Here $\mathbf{z}$ represents the sequence of contextualized tokens returned by BERT. The second stage $f(\tilde{\mathbf{x}}) = p(y|\tilde{\mathbf{x}})$ predicts the label distribution of y based on the reduced input sequence $\tilde{x}$ ($\tilde{x} = \mathbf{x} \otimes \mathbf{r}$).

$\hat{\cdot}$ refers to the predicted value and $\cdot^*$ describes the ground truth, while the plain variable refers to the random variable. For example, $p(r_i^*|\mathbf{x})$ refers to the predicted probability of rationale token i to be the same class as given by the human annotation, while $\hat{r}_i = \operatorname{argmax}_r p(r|\mathbf{x})$ is the predicted class.

Details of Query Strategies We apply the active learning methods in a batched setup where b_t instances are selected per active learning iteration t. If not specified, the top-b_t instances are selected according to the acquisition function. Iteration $t = -1$ refers to the warm-start iteration and b_{-1} to warm-start budget respectively.

In each iteration, we train the model from scratch to avoid overfitting to the instances selected early on [Ein+20; She+21].

The acquisition function of MNLP was introduced by Shen et al. [She+17] and is derived from the LC (3.14) method by assuming independence of the individual token of the sequence (3.15) and taking the sum of the log-probabilities instead of the product of the probabilities them-self. (3.16), however, favors longer sequences since they are less likely [She+21]. In order to compensate for this, a normalization factor is introduced, leading to (3.17):

$$\begin{aligned}
\mathbf{x}^* &= \operatorname{argmax}_{\mathbf{x}} -p(\hat{\mathbf{r}}|\mathbf{x}) && (3.14)\\
&= \operatorname{argmax}_{\mathbf{x}} -\prod_{i=1}^{n} p(\hat{r}_i|\mathbf{x}) && (3.15)\\
&= \operatorname{argmax}_{\mathbf{x}} -\sum_{i=1}^{n} \log(p(\hat{r}_i|\mathbf{x})) && (3.16)
\end{aligned}$$

$$\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x}} -\frac{1}{n}\sum_{i=1}^{n} \log(p(\hat{r}_i|\mathbf{x})) \qquad (3.17)$$

We follow [She+21] and apply Variation Ratio (already introduced in subsection 2.3.4 “Classical Approaches”) by simulating the ensemble $\mathcal{C}$ using Monte-Carlo dropout by performing $|\mathcal{C}| = m$ forward passes with all dropout layers set to evaluation mode. The final utility of VR is defined by:

$$\mathbf{x}^* = \text{argmax}_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^{n} VR_i \tag{3.18}$$

with $VR_i = 1 - \frac{\text{vote}_{\mathcal{C}}(\hat{r}_i, \mathbf{x})}{|\mathcal{C}|}$. These methods selected instances where the average token disagreement amongst the simulated ensemble is maximized.

Analogous to [Ein+20], we employ the greedy version of Core-Set (GCS) – described in subsection 2.3.5 "Deep Learning" – by using the embedding $z_{[CLS]}$, which is returned by the last layer of the BERT-encoder as the representation of the sequence.

Additionally, we report the results of the random baseline Random.

Parameters In order to speed up the whole process, we only train the first stage – the explanation generator – in each active learning iteration and the second stage only after the last epoch. Therefore we report the token F1 of the whole pipeline $f(\cdot)$ only after the last epoch. All reported scores are test scores.

We run each active learning method over six iterations, each from four different seeds. We use different warm-start budgets b_{-1} and select instances per iteration b_t as well as per dataset (see Table 3.5 "Shared parameters of the active learning strategies per dataset"). The training seeds of the random number generator are reset to the same value before the training in each active learning iteration, more precisely before the parameter initialization of the model, in order to reduce variance that is not introduced by the data selection.

Table 3.5: Shared parameters of the active learning strategies per dataset.

Dataset	b_{-1}		b_t		$b_{-1:5}$	
	rel.	abs.	rel.	abs.	rel.	abs.
Movie Reviews	0.05	80	0.1	160	0.65	1040
MultiRC	0.02	480	0.04	961	0.26	6247

For VR we set $m = 5$.

3.4.3 Results

The results of our main experiments are reported in Figure 3.4 and Figure 3.5 for Movie Reviews and MultiRC, respectively. Each column contains three metrics for one active learning method and the Random baseline plotted against the active learning iteration. The first row shows the token F1 on the rationales predicted by $g(\cdot)$, the second row the classification macro F1 of the auxiliary head f', and the last row the size of the labeled dataset $|L_{-1:t}| = b_{-1:t}$ at time t.

For both datasets, none of the methods outperforms the Random baseline regarding their token F1 clearly. For VR on Movie Reviews, one could argue that it is slightly better, but for all time steps, the performance is within one standard deviation of the baseline therefore it is not much stronger. On MultiRC, one can even see a degradation compared to Random, which lies outside of one standard deviation. Moreover, we observe that GCS is performing the worst on Movie Reviews, while on MultiRC it is performing the best compared to the other methods. The values of the macro F1 of the auxiliary head are less stable than the token F1 for all methods on both datasets and stay roughly the same.

Table 3.6 displays the results of the final models after the last active learning iteration, as well as the scores of the ExPred model train on the full datasets ($b = 1$) as a reference. The results confirm what we already know from the experiment in section 3.2 "Preliminary Analysis: The Baseline and its Need for Rational Supervision": we only need a fraction of the full dataset to get relatively close to the performance at 100%.

One difference between the two datasets one has to bear in mind is that the absolute number of instances added in each iteration to the labeled training set is much bigger for MultiRC than for Movie Reviews. This and the fact that GCS does comparably well on MultiRC suggests that diversity is an issue for the other approaches. The reason why GCS is not working could potentially lie in the chosen representation. [Ma+19] and [Cho+21] show that there are more sophisticated approaches that yield better embeddings than just taking the $z_{[cls]}$ as sequence representation, like averaging the representation of all tokens and the output of different layers.

Overall, one can say that none of those active learning methods clearly improves over the Random baseline. In the case of MultiRC the performance even drops a little bit. This answers **RQ3** negatively. Active learning or at least none of the tested off-the-shelf active learning methods is helping to improve given a fixed annotation budget.

In the next sections we will further investigate potential reasons for this behavior in order to enable the design of active learning methods that are tailored to rationale extraction using ExPred.

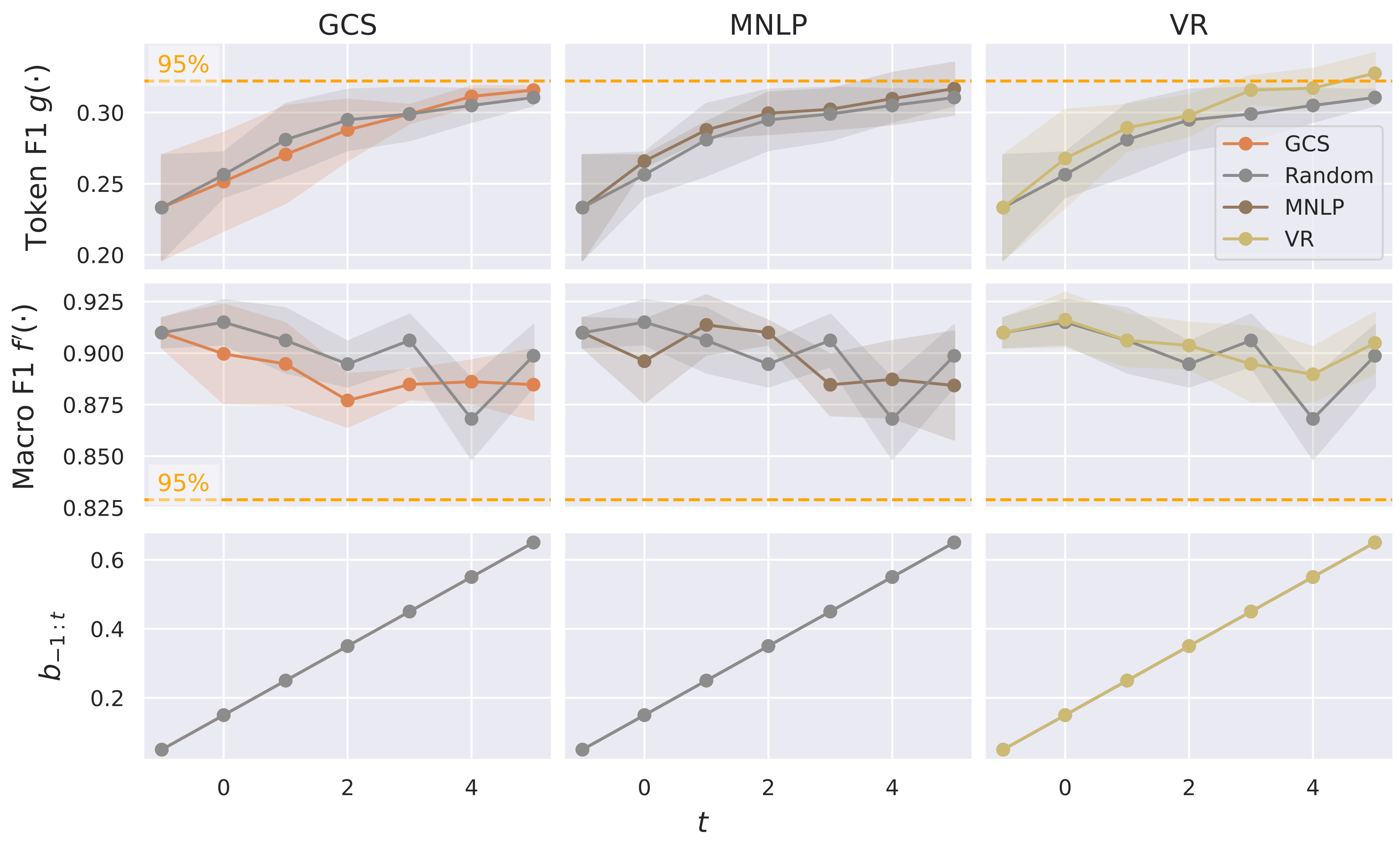


Figure 3.4: Active Learning Methods vs RANDOM on Movie Reviews. The orange line indicates 95% of the performance of EXPRED trained on the full dataset.

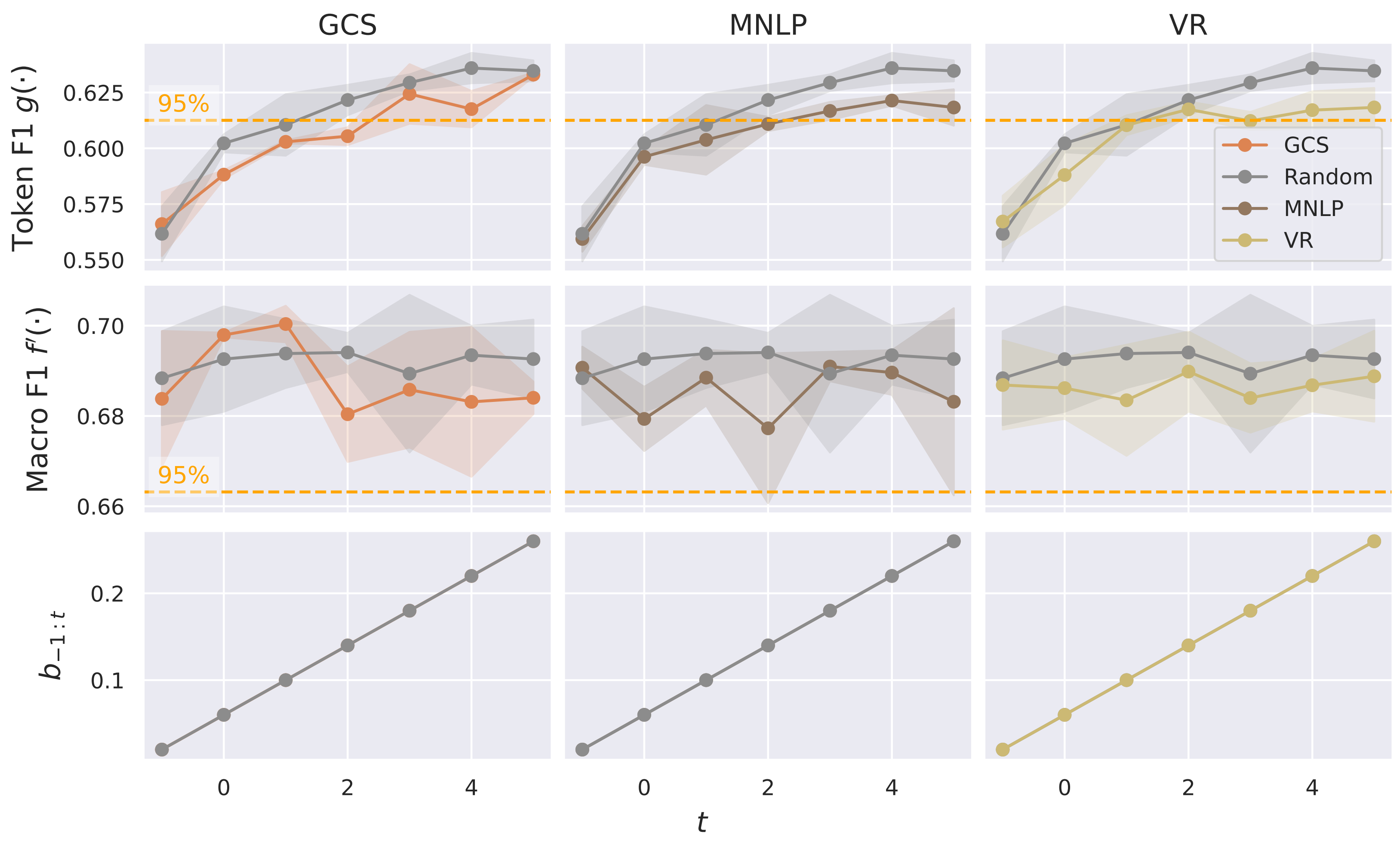


Figure 3.5: Active Learning Methods vs RANDOM on MultiRC. The orange line indicates 95% of the performance of EXPRED trained on the full dataset.

Table 3.6: Test scores of the final models after the last active learning iteration. The results are in the format "mean (± std. dev.)".

Active Learner	Token F1 $g(\cdot)$		Macro F1 $f(\cdot)$	
Movie Reviews				
ExPred ($b = 1$)	0.339	(± 0.013)	0.872	(± 0.018)
GCS	0.316	(± 0.003)	0.885	(± 0.015)
MNLP	0.317	(± 0.018)	0.872	(± 0.020)
Random	0.311	(± 0.005)	0.884	(± 0.022)
VR	0.328	(± 0.014)	0.889	(± 0.017)
MultiRC				
ExPred ($b = 1$)	0.647	(± 0.004)	0.708	(± 0.016)
GCS	0.633	(± 0.001)	0.717	(± 0.002)
MNLP	0.618	(± 0.008)	0.718	(± 0.005)
Random	0.635	(± 0.005)	0.716	(± 0.005)
VR	0.618	(± 0.009)	0.719	(± 0.009)

3.4.4 Analysis: Reason for the Failure of Active Learning

There are several cases when active learning fails. In this section we list different hypotheses for this and conduct experiments in the subsequently sections to support or refute them.

Hypothesis 1: All instances are equally helpful. The most trivial case occurs when all instances are equally helpful. If all instances provide the same utility for training the model it does not matter which ones are selected and therefore it exists no acquisition function that can do better (or worse) than random. The slight but consistent drop of performance on MultiRC of all active learning methods below Random suggests otherwise.

Hypothesis 2: The acquisition function fails to measure the utility. Assuming that some instances are more helpful than others, active learning fails if the selection strategy is not able to discover the usefulness. This can either be the case because the aspect measured by the utility function, like diversity (GCS) or uncertainty (MNLP or VR), is not correlated with the utility or if the function fails to measure it correctly.

GCS can fail if the chosen embedding space is not semantically meaningful. Another issue of GCS pointed out by [GS19] is that it tends to under-sample high density regions of the embedding space and to over-sample sparse regions because its optimization is solely based on the instances' distances.

Uncertainty-based methods like MNLP and VR are unsuccessful if the estimated uncertainty or ensemble agreement does not reflect the models' true certainty. One has to point out that MNLP uses the probability of the class predicted by the model. If the model predicts the wrong class but with high probability, the utility score produced by MNLP is low, even though it would be helpful to label this particular example.

Hypothesis 3: The batch construction leads to non-representative batches. Lastly, VR and MNLP could fail even if they measured the utility of each individual instance correctly. This is the case if the top-b_t instances were very similar, i.e. they similar new information to the model. This is the case because they measure the utility for the current model when adding one single instance and they do not take the other instances into account already selected in the same active learning iteration. Besides the dissimilarity within batches, one has also to assure diversity across batches to ensure representatives for the data distribution.

3.4.4.1 Do the instances differ in their utility?

In order to assess the upper bound a purely uncertain-based method can achieve, we propose two oracle methods that have access the ground truth labels of the rationales. This can be seen as estimating the ground truth of the utility of the instances. We emphasize that this is not a practical scenario and that those acquisition functions are only used to analyze the behavior of the active learning methods and properties of the datasets.

As emphasized above, MNLP fails to produce good scores if the model is overconfident on wrong decisions. We mitigate this by modifying MNLP to use the ground truth labels $\mathbf{r}^*$ instead of the predicted ones $\hat{\mathbf{r}}$, leading to *Maximum Normalized True Log-Probability* (MNTLP):

$$\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x}} -\frac{1}{n}\sum_{i=1}^{n}\log(p(r_i^*|\mathbf{x})) \tag{3.19}$$

If the ground truth is available, another natural choice is just to select the instances for which the models current predictions differs most from the annotations, which is measured by the loss:

$$\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x}} \mathcal{L}_{exp}(\hat{\mathbf{r}}, \mathbf{r}^*) \tag{3.20}$$

where $\mathbf{r}$ is the model's prediction associated with $\mathbf{x}$. We call this method *Highest Explanation Loss* (HXL).

We use the same training setup as described in subsection 3.4.2 “Experimental Setup” and report the results in Figure 3.6.

Results For Movie Reviews, we see a considerable improvement in token performance for both, HXL and MNTLP, over Random, where HXL outperforms MNTLP. The gains are naturally higher in earlier active learning iterations. The auxiliary head performance stays stable and is comparable for all three methods. These results show that for Movie Reviews there are instances that provide a much higher utility for the models training than others and that the non-oracle query strategies are not able to reveal it.

The results in the case of MultiRC paint another picture: the token performance of HXL and MNTLP is much worse than Random, where MNTLP is a little worse than HXL. This shift is also visible for the macro F1, where both oracle versions perform worse than Random. Here the drop in performance is not as big as for the token F1, but it is still substantial. The fact that both metrics decrease, indicates that the selected instances do not provide much semantically new information since it affects both tasks and therefore probably the encoder. Nevertheless, the change in performance shows that the instances have different values. Among other things, these results could be caused by selecting instances that are not representative of the distribution. One the one hand this would be the case if they are outliers. Outliers would be probably hard to classify but not helpful if added to the training set. On the other hand this can happen if the selected instances are very contently similar or share a lot of tokens, individually providing new but similar information to the model. We will investigate the latter hypothesis in the next subsubsection 3.4.4.2 "Diversifying Oracles", since the effect seems likely because it also happens on MultiRC where larger batches are selected.

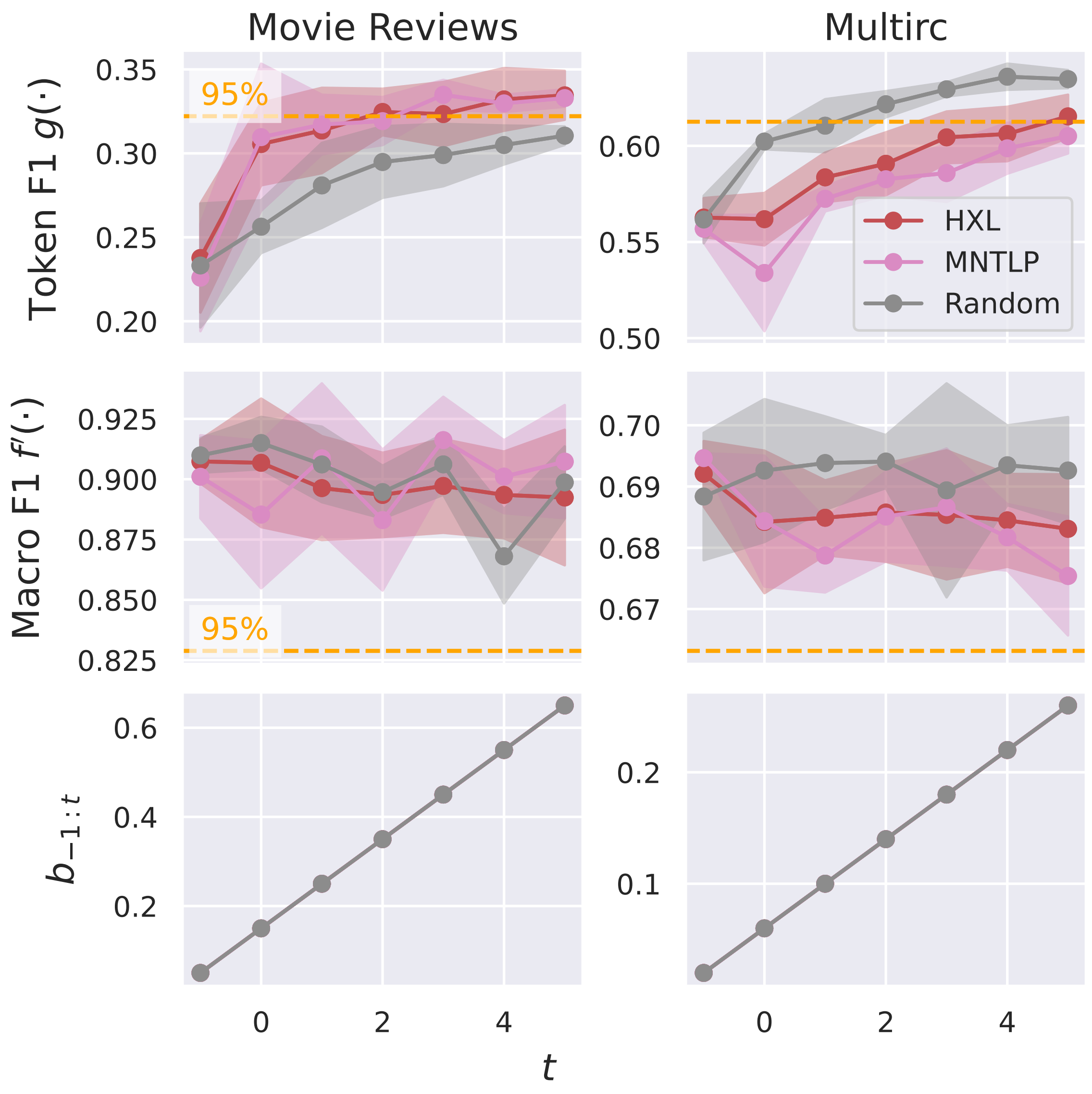


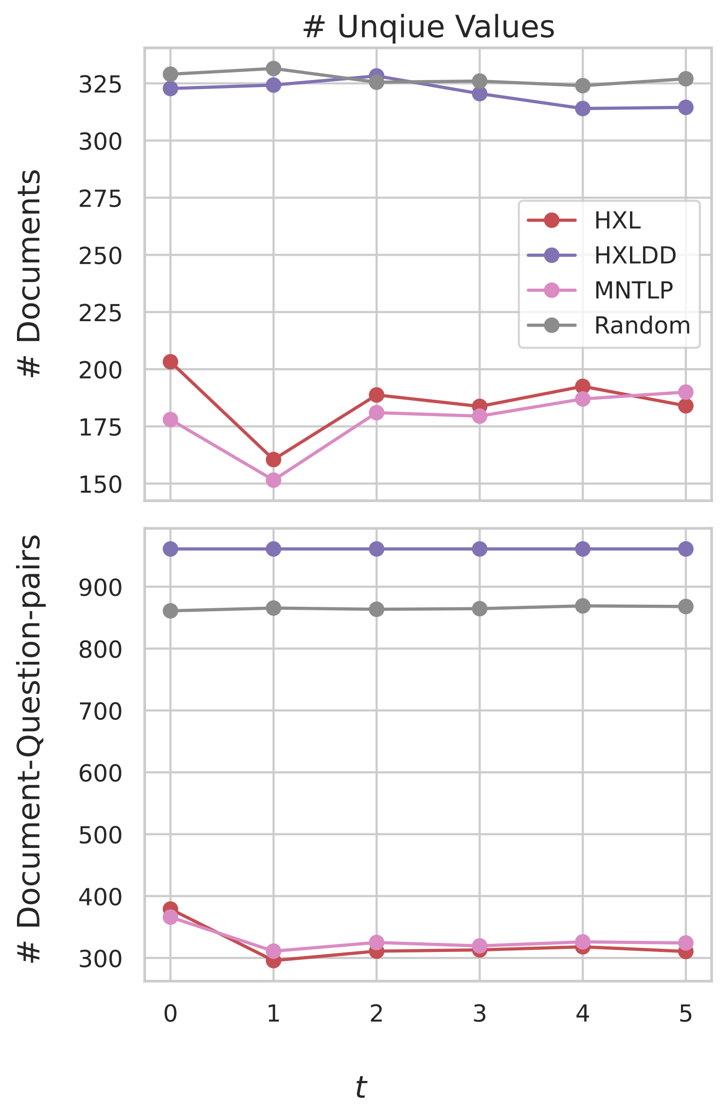
Figure 3.6: Results of the oracles vs Random baselines.

3.4.4.2 Diversifying Oracles

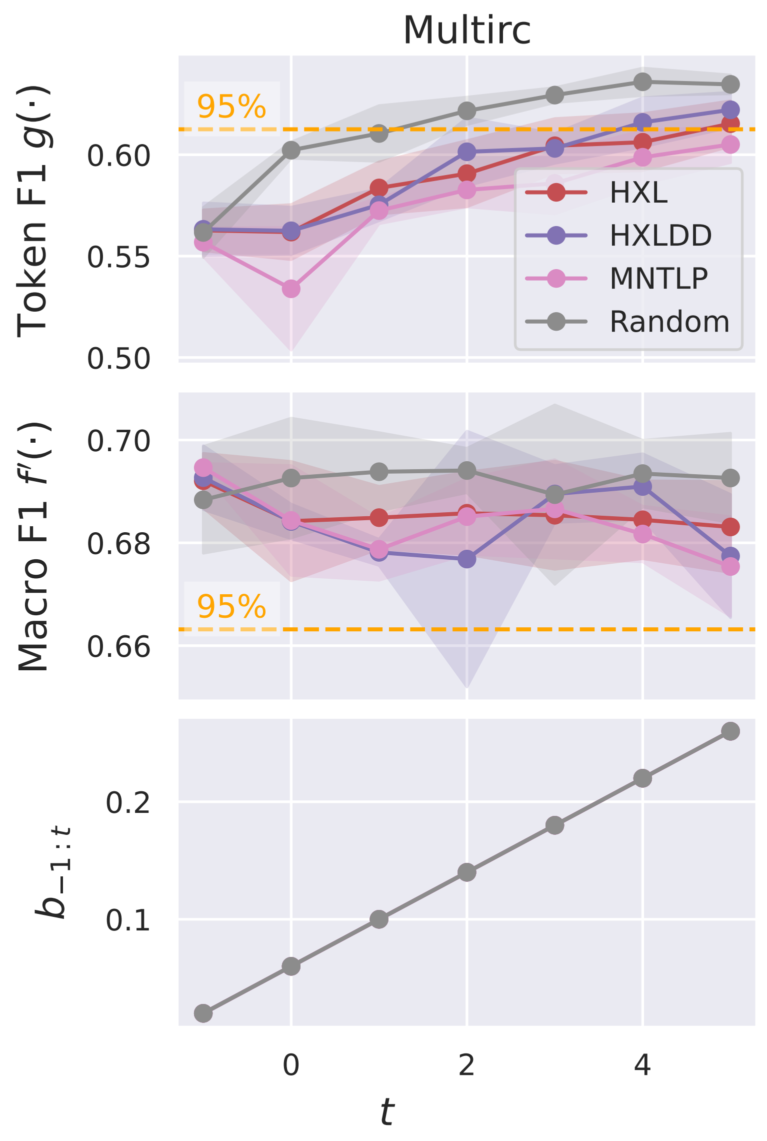
The way the instances in the dataset are constructed in MultiRC but produces greater risk of selecting homogeneous batches. The instances follow the structure: `[question]||[answer][SEP][document]`. The ERASER version of the dataset was derived from the original MultiRC dataset introduced in [Kha+18]. The instances there consist of a `document`, a `question` and multiple `answers`, where several of them are correct. On average, each question is associated with 2.6 correct and 5.4 answers in total. Additionally, each document appears in multiple instances with different questions (on average 11.3). Since the documents are longer than the questions and answers, it means that instances just sharing the document are relatively similar and they become even more similar if they share the question too.

Results Figure 3.7a shows the average number of unique documents and document-question-pairs selected in one iteration. One can see that the batches selected by RANDOM are much more diverse in terms of these two metrics than for HXL and MNTLP. We added another method *Highest Explanation Loss Dummy Diversity* (HXLDD) that uses the same acquisition function as HXL but adds a post-processing step. After scoring all instances according to HXL, we first find the instance with the highest score per document-question-pair and set the scores of all others in the group to 0, enforcing diversity in a very naive way. This results in much more diverse batches.

Figure 3.7b displays the performance results. The token F1 increases a little bit compared to HXL and MNTLP but does not reach the performance of RANDOM. Although the results are not very strong, they suggest that diversity is an issue here and that using a more complex diversification methods could potentially improve the results. Even though the interaction between effects of diversity and uncertainty is hard to disentangle. One has to point out that the way we are creating diversity only enforces diversity within the current batch L_t but does not take the instance already labeled in $L_{-1:t-1}$ into account.



(a) Diversity of the selected instances on MultiRC



(b) Results of HXLDD vs oracles and RANDOM on MultiRC

3.4.4.3 Rank Correlation of Different Methods

Following [GS19] and [Hua+16], we compare and analyze the way the methods select the instances by converting the scores produced during the first iteration ($t = 0$) to ranks and plot them for each pair of active learning methods. Lower ranks are better than higher ranks, while higher scores indicate higher utility.[3] Due to a large number of instances we prefer a 2D-histogram with a logarithmic color scale over a pure scatter plot. The lightest color corresponds to the maximum number of instances per bin and the darkest to the minimum. Additionally, we report Spearman's rank correlation coefficient and on the diagonal a distribution plot of the scores. For GCS, we interpret the selection order as score. Instances that are not selected are scored with zero are not included in the plot. We also exclude instances scored by HXLDD with zero since they do not reflect the uncertainty.

A limitation of this procedure is that during the conversion to ranks the distance between the consecutive score is not preserved. An alternative would be to normalize the scores and look at the density, but we keep it consistent with [GS19].

Results Figure 3.8 and Figure 3.9 compare the ranking of the non-oracle methods. The score distribution of GCS is not very informative since an instance selected in the corset creation at iteration i has the score $b_t - i$ and therefore each score appears exactly once. (For more details see subsection 2.3.5 "Deep Learning"). In general, the ranks produced by GCS are not correlated with the ranks of VR and MNLP, neither for MultiRC nor for Movie Reviews. This is not very surprising since the score are constructed artificially and the selection order is not necessarily meaningful because it is highly dependent on the initial selection. Regarding the rankings of VR and MNLP, one can observe that correlation is quite strong for Movie Reviews ($\rho = 0.74$) but slight for MultiRC ($\rho = 0.36$). The difference in correlation indicates that the scores for Movie Reviews are more ambiguous than MultiRC. For both datasets, they agree on a lot of instances that are very useful or not at all. But especially for MultiRC, there is also a great number of instances where they do not agree on the middle ranks. For both datasets and for both uncertainty-based methods most of the instances get a low score and just a small proportion is rated very useful.

The ranks for the oracle-based methods HXL and MNTLP (Figure 3.10 and Figure 3.11) are highly correlated ($\rho = 0.52$) for both datasets. This is not surprising since the acquisition functions do not differ much (see (3.21) and (3.27)) and the influence of the model's prediction and therefore the variation is reduced by using the ground truth data. On MultiRC, HXLDD is similarly correlated to MNTLP ($\rho = 0.52$) but stronger to HXL ($\rho = 0.52$). One difference between Movie Reviews and MultiRC is that for MultiRC the methods do very strongly agree on high and low ranks while for Movie Reviews the methods agree mainly on high ranks.

[3]The picture does not change much in a later iteration.

Since we discovered from Figure 3.6 that both oracle methods improve compared to Random on Movie Reviews, we can regard the scores produced by them as possible ground truth values. First of all, we observe the same phenomenon for GCS as before – it is not correlated with the other methods at all. For Movie Reviews MNTLP is strongly correlated with MNLP ($\rho = 0.57$) and with VR ($\rho = 0.6$). This shows, that VR is somewhat better in estimating the true scores. In contrast, HXL is almost not correlated with MNLP ($\rho = 0.15$) and VR ($\rho = 0.1$). Again, they largely agree with HXL on the high ranks (low utility instances), but there are also a large number of instances ranked with a high score by MNLP and VR which exhibit low scores produced by HXL forming a "C"-shape. This either means that a) these are the instances where the model is wrong but highly confident or b) that this relates to the difference in weighting to tokens (see subsubsection 3.4.4.4 "Weighting of Tokens"). The former reason is unlikely because this must have affected MNTLP as well. For MultiRC, the correlations are lower for MNTLP but higher for HXL, compared to Movie Reviews. That fits the mold that this is a problem related to the weighting of the tokens in the acquisition function since the difference here is lower than for Movie Reviews, where the rationales are shorter and therefore the weighting has a stronger influence.

Overall, one can observe that the correlations are stronger (except for the oracle methods) for Movie Reviews than for MultiRC. The higher consistency across methods supports what the performance results already indicated, i.e. the utility of the instances is easier to measure for Movie Reviews than for MultiRC.

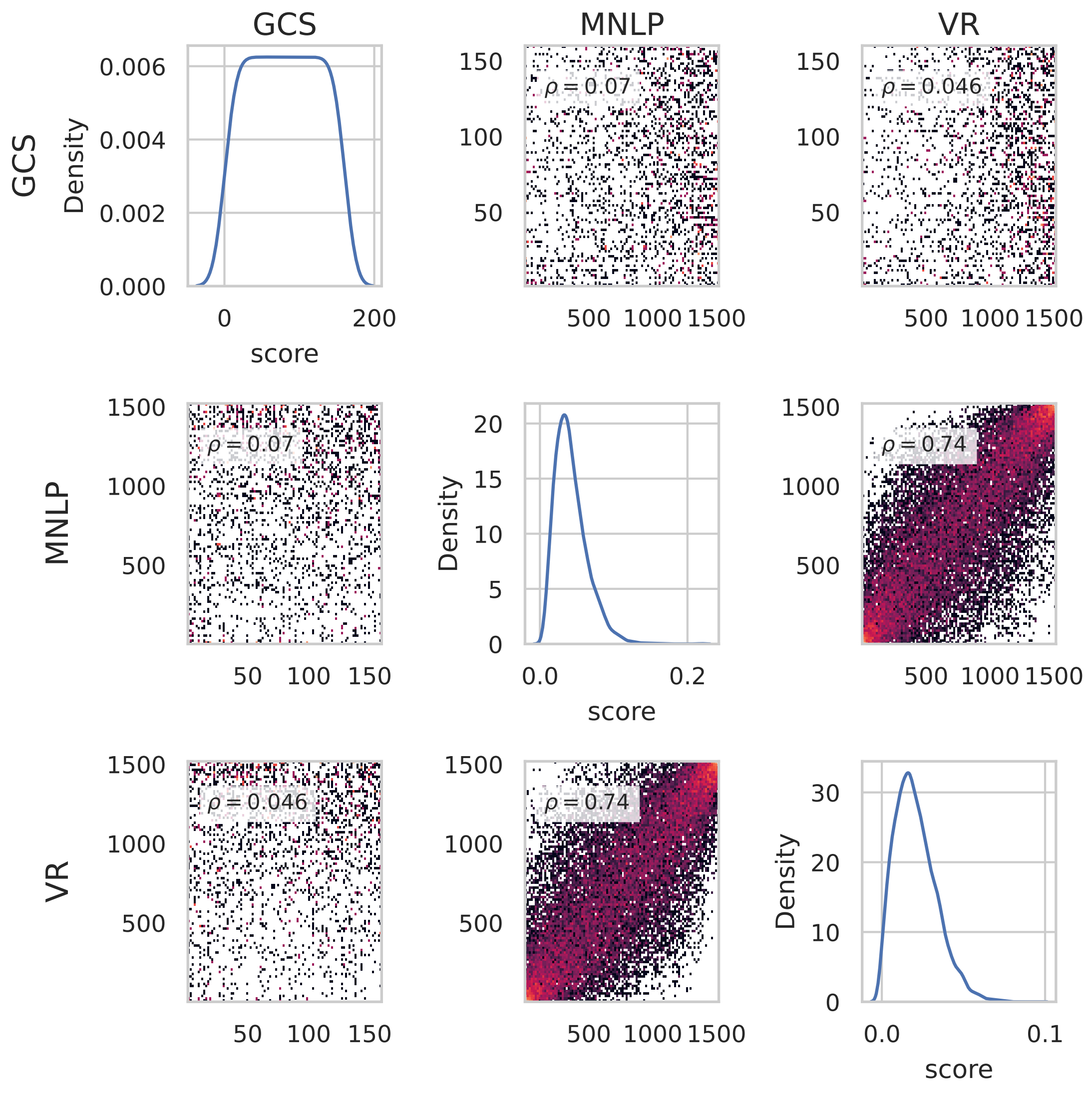


Figure 3.8: Pair-wise rank comparisons of the active learning methods GCS, MNLP and VR on Movie Reviews

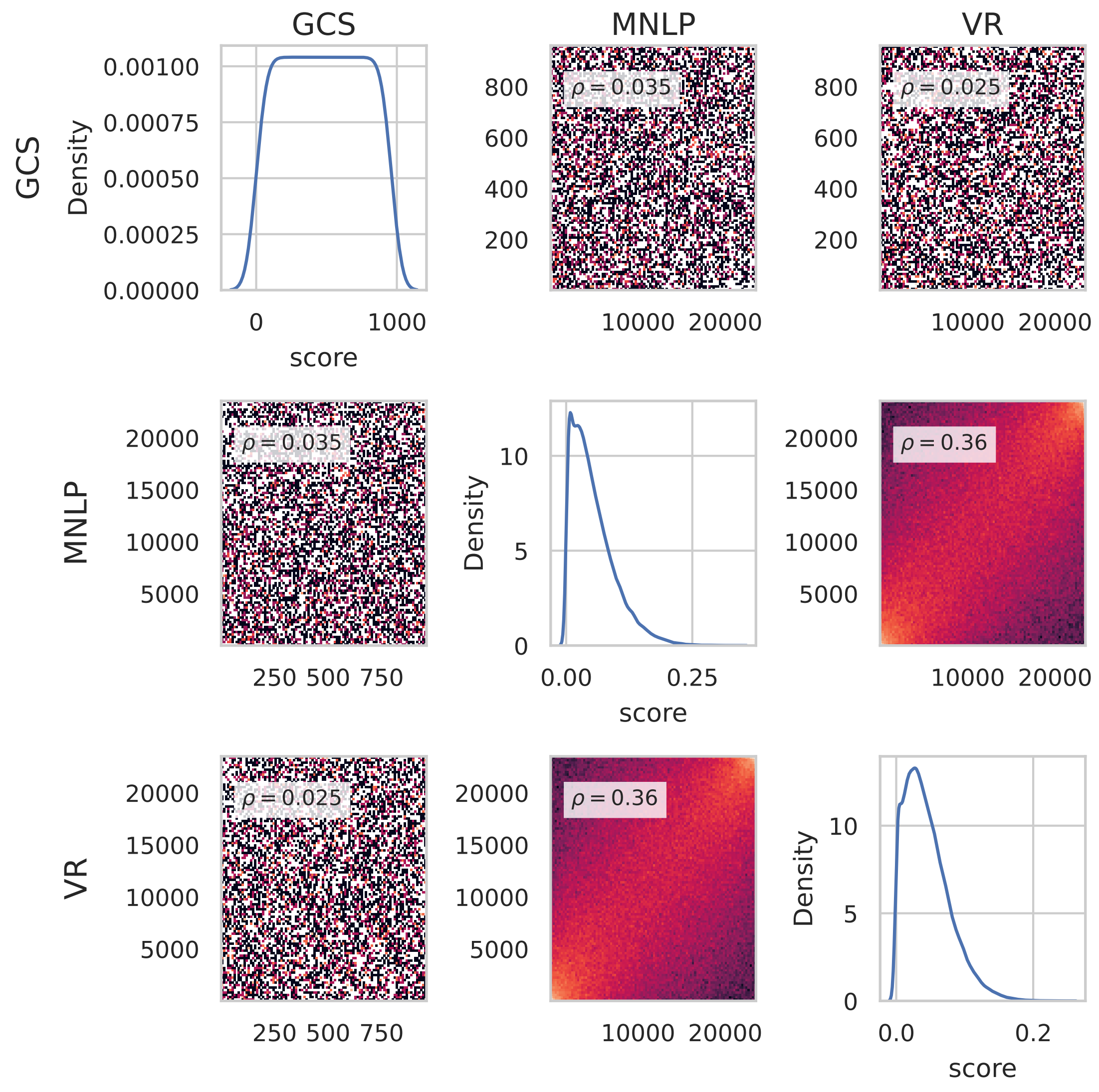


Figure 3.9: Pair-wise rank comparisons of the active learning methods GCS, MNLP and VR on MultiRC

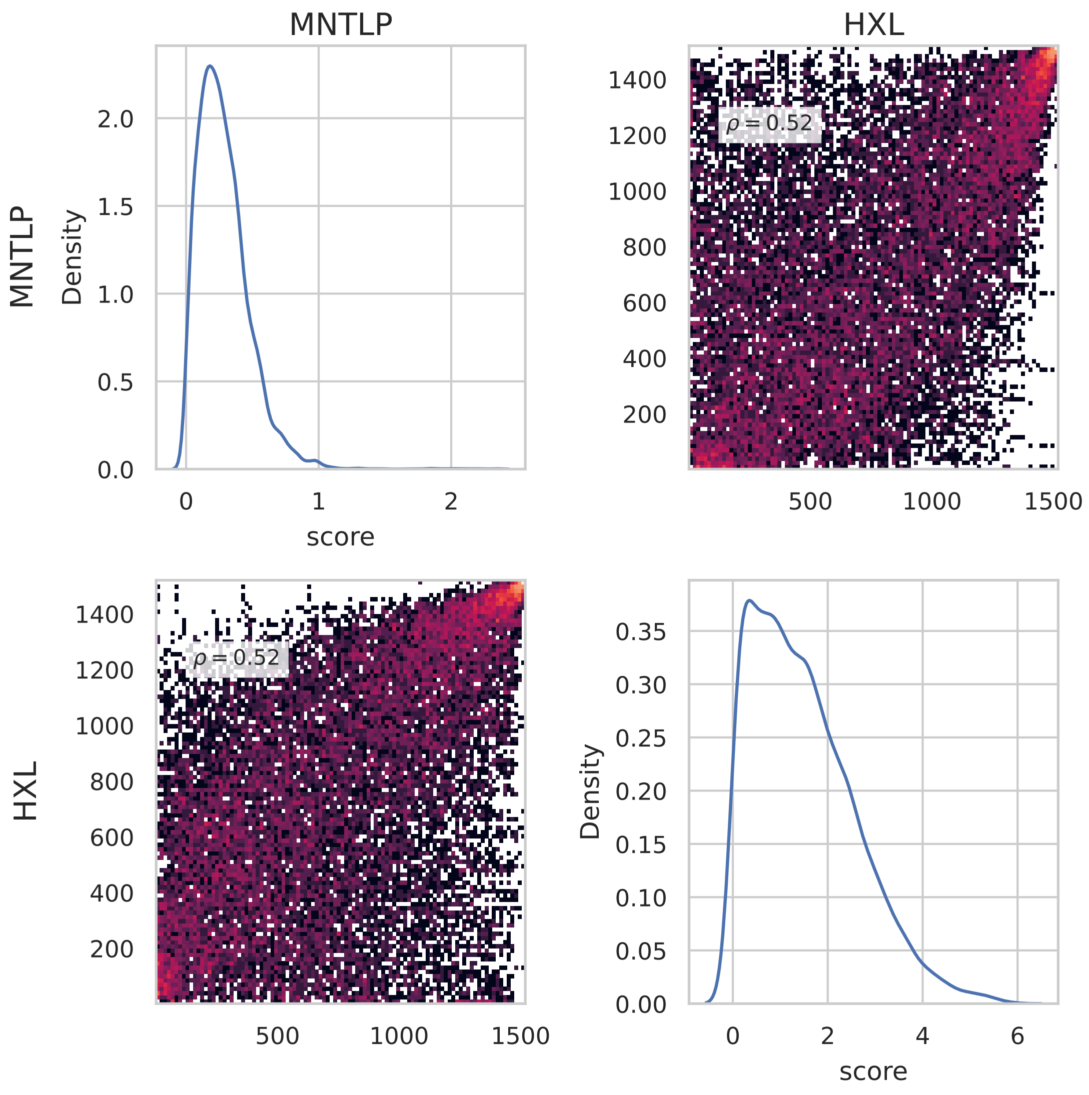


Figure 3.10: Pair-wise rank comparisons of the oracle active learning methods GCS, MNLP and VR on Movie Reviews

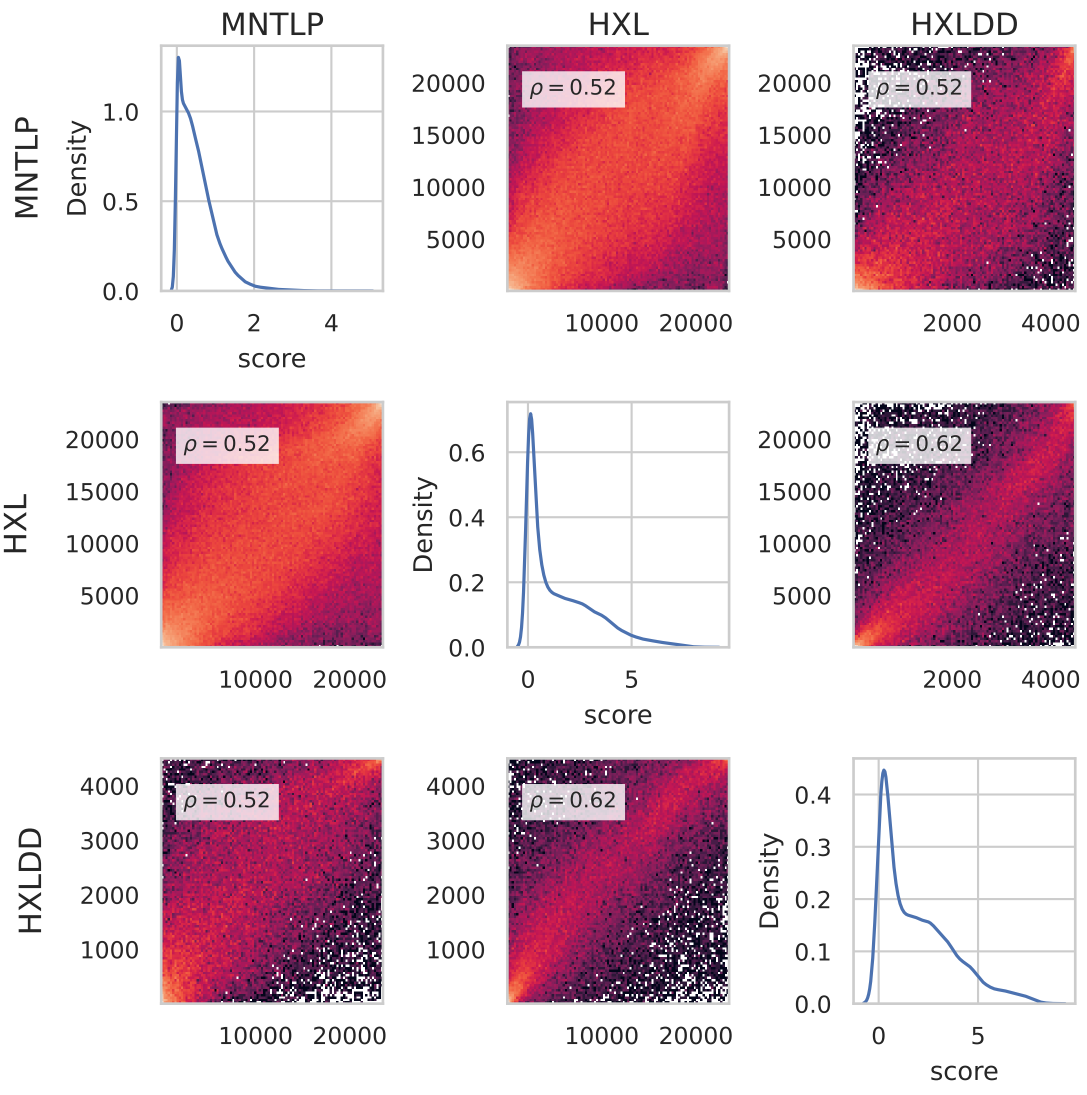


Figure 3.11: Pair-wise rank comparisons of the oracle active learning methods GCS, MNLP and VR on MultiRC

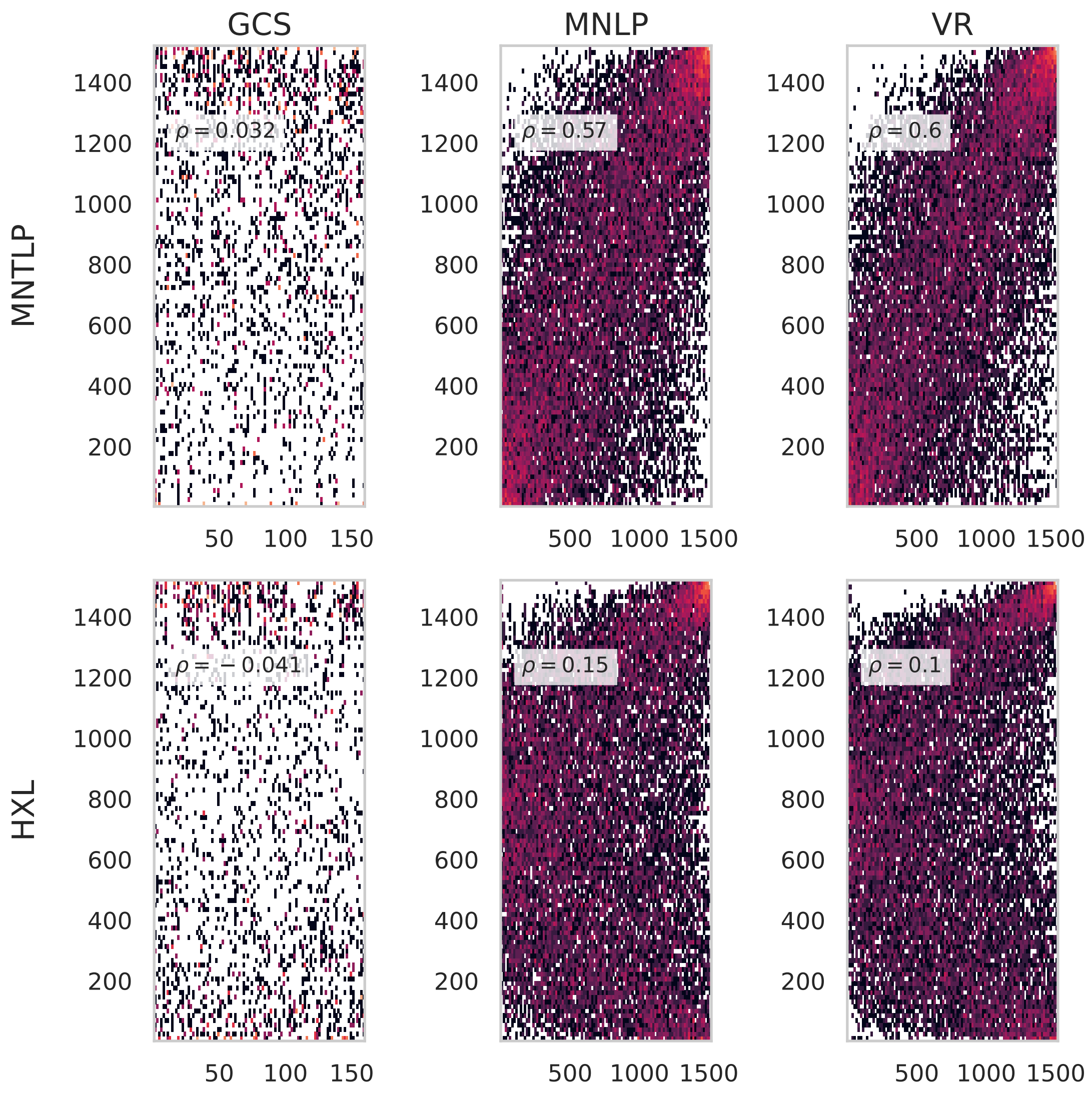


Figure 3.12: Pair-wise rank comparisons of the oracle vs normal active learning methods GCS, MNLP and VR on Movie Reviews

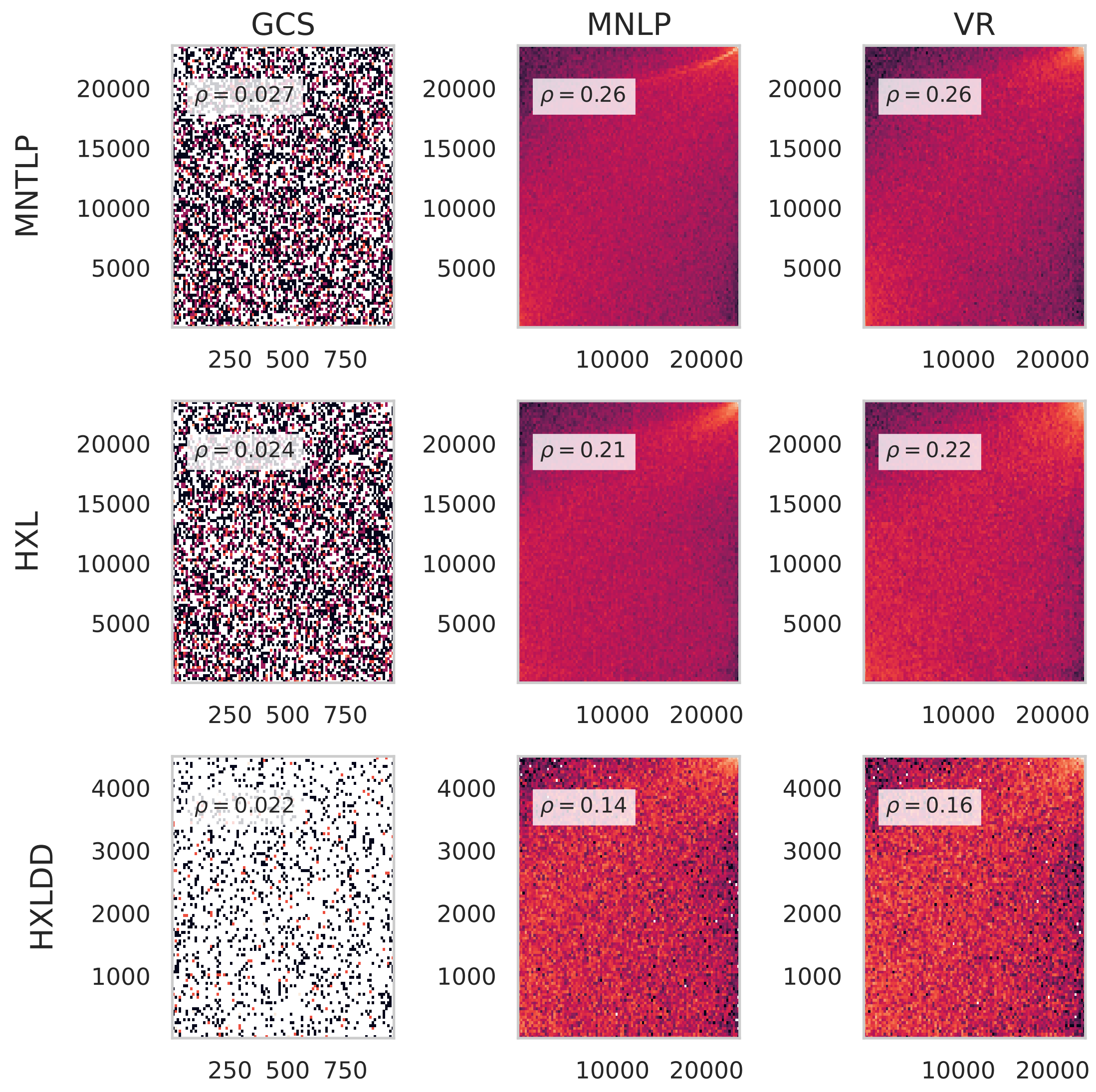


Figure 3.13: Pair-wise rank comparisons of the oracle vs normal active learning methods GCS, MNLP and VR on MultiRC

3.4.4.4 Weighting of Tokens

Another potential issue of the active learning methods we look at is that they are weighting each token in one sequence equally. For the training of ExPred it has been shown to be beneficial if to use the weighted average of the token-wise binary cross-entropy loss instead of an unweighted version.

By revisiting the formulation of the acquisition functions of MNTLP (3.21) and HXL (3.22), it is apparent that the only difference lies in the weights w_i.

$$\mathbf{x}^* = \text{argmax}_{\mathbf{x}} -\frac{1}{n}\sum_{i=1}^{n} \log(p(r_i^*|\mathbf{x})) \tag{3.21}$$

$$\begin{aligned}
\mathbf{x}^* = \text{argmax}_{\mathbf{x}} &= \mathcal{L}_{exp}(r_i, r_i^*) && (3.22)\\
&= \frac{1}{n}\sum_{i=1}^{n} w_i \,\text{BCE}(r_i, r_i^*) && (3.23)\\
&= \frac{1}{n}\sum_{i=1}^{n} w_i(-1)(r_i^* \log(p(r=1|\mathbf{x})) + (1-r_i^*)\log(1-p(r=1|\mathbf{x}))) && (3.24)\\
&= -\frac{1}{n}\sum_{i=1}^{n} w_i(r_i^* \log(p(r=1|\mathbf{x})) + (1-r_i^*)\log(p(r=0|\mathbf{x}))) && (3.25)\\
&= -\frac{1}{n}\sum_{i=1}^{n} w_i \begin{cases}\log(p(r=1|\mathbf{x})) & \text{if } r^*=1\\ \log(p(r=0|\mathbf{x})) & \text{otherwise}\end{cases} && (3.26)\\
&= -\frac{1}{n}\sum_{i=1}^{n} w_i \log(p(r_i^*|\mathbf{x})) && (3.27)
\end{aligned}$$

Additionally, in Figure 3.6 one sees that HXL outperforms MNTLP at least on Movie Reviews. Assuming the bad performance on MultiRC is diversity-related one could at least expect some improvement on Movie Reviews when using the loss of the predicted class (HPXL):

$$\begin{aligned}
\mathbf{x}^* &= \text{argmax}_{\mathbf{x}} \mathcal{L}_{exp}(r_i, \hat{r}_i) && (3.28)\\
&= -\frac{1}{n}\sum_{i=1}^{n} w_i \log(p(\hat{r}_i|\mathbf{x})) && (3.29)
\end{aligned}$$

We apply HPXL using the same setting as in previous experiments and report macro F1 and token F1. In Figure 3.14 the results of HPXL for Movie Reviews and MultiRC are presented. The macro F1 stays relatively stable for both datasets. For MultiRC, the token F1 increases over time but performs worse than random. Although one has to point out that is among the better methods on MultiRC. On Movie Reviews, however, HPXL outperforms Random substantially. The low performance on MultiRC could be caused by the diversity issue discussed earlier on. Additionally, the rationales are shorter in the case of Movie Reviews and therefore the weighting is more important than for MultiRC.

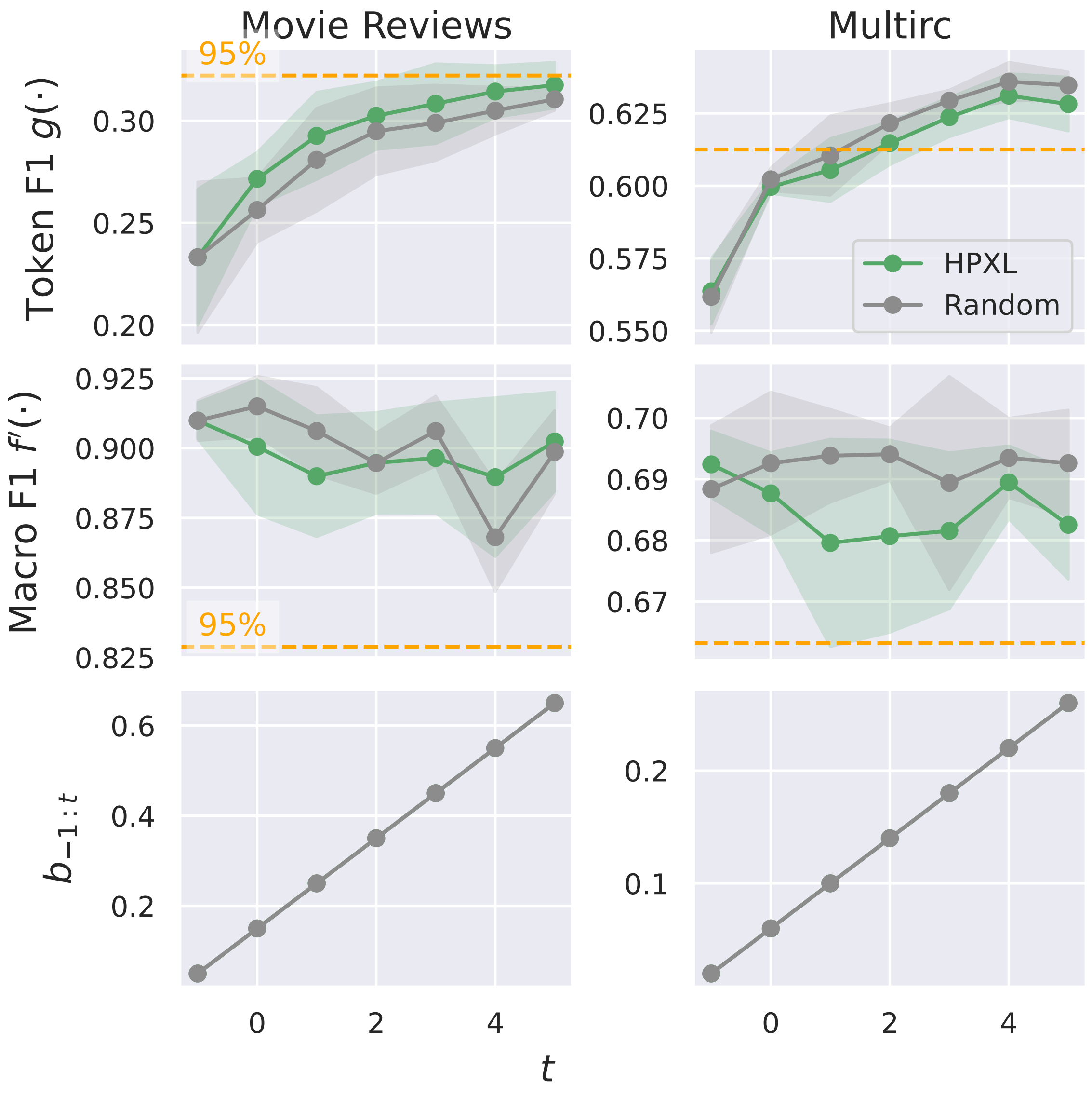


Figure 3.14: HPXL vs Random on Movie Reviews and MultiRC

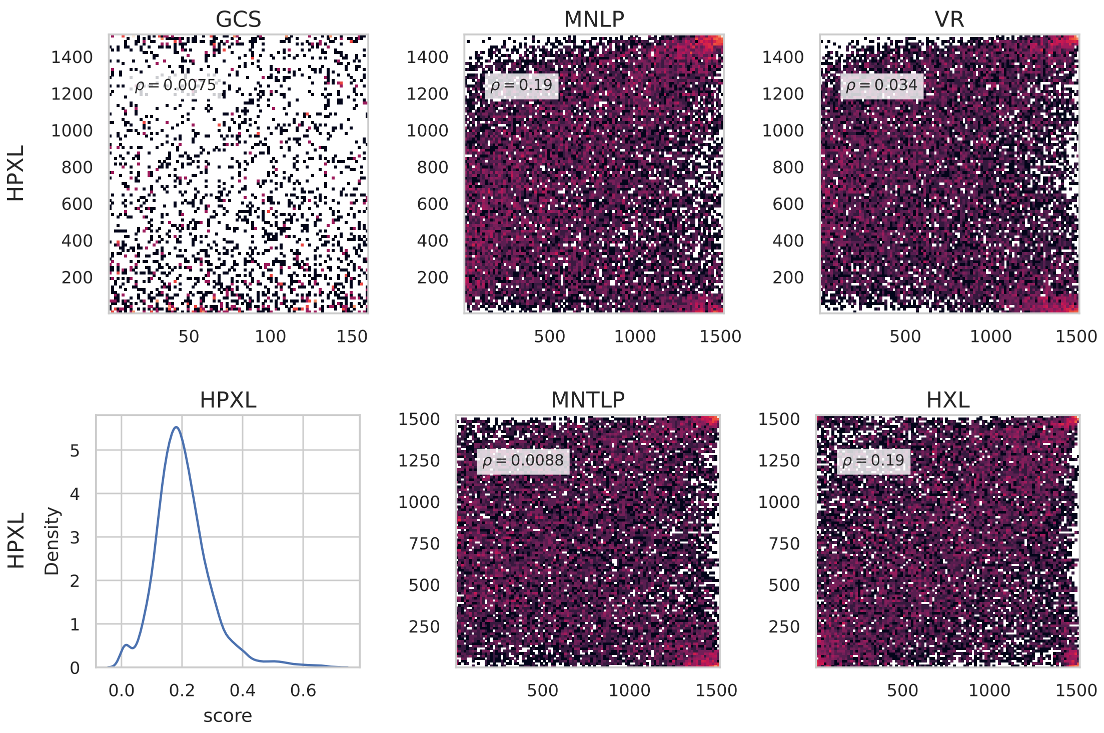


Figure 3.15: Pair-wise rank comparisons of HPXL on Movie Reviews

Results Figure 3.15 and Figure 3.16 depict the rank comparisons of HPXL to the other methods. The plots follow the same schema as described in subsubsection 3.4.4.3 “Rank Correlation of Different Methods”. GCS shows a completely different behavior than HPXL, independent of the dataset. The results on MultiRC do not show any salience. The oracle methods MNTLP and HXL are slightly less correlated ($\rho = 0.22$ and $\rho = 0.24$) than the non-oracle methods MNLP and VR ($\rho = 0.32$ and $\rho = 0.21$). On Movie Reviews, all methods (except GCS) show a cluster with low ranks assigned by HPXL but high ranks assigned by the other methods. For the oracle methods, especially for HXL, these are probably the instances where the model’s predictions differ from the ground truth. For the non-oracle methods and also for MNTLP, instances with very short rationales are most likely located there. Interestingly, VR is also stronger correlated than MNLP with HPXL.

Overall, we can hypothesize that weighting up the uncertainty of the underrepresented class does improve the results primarily in cases where we have short rationales.

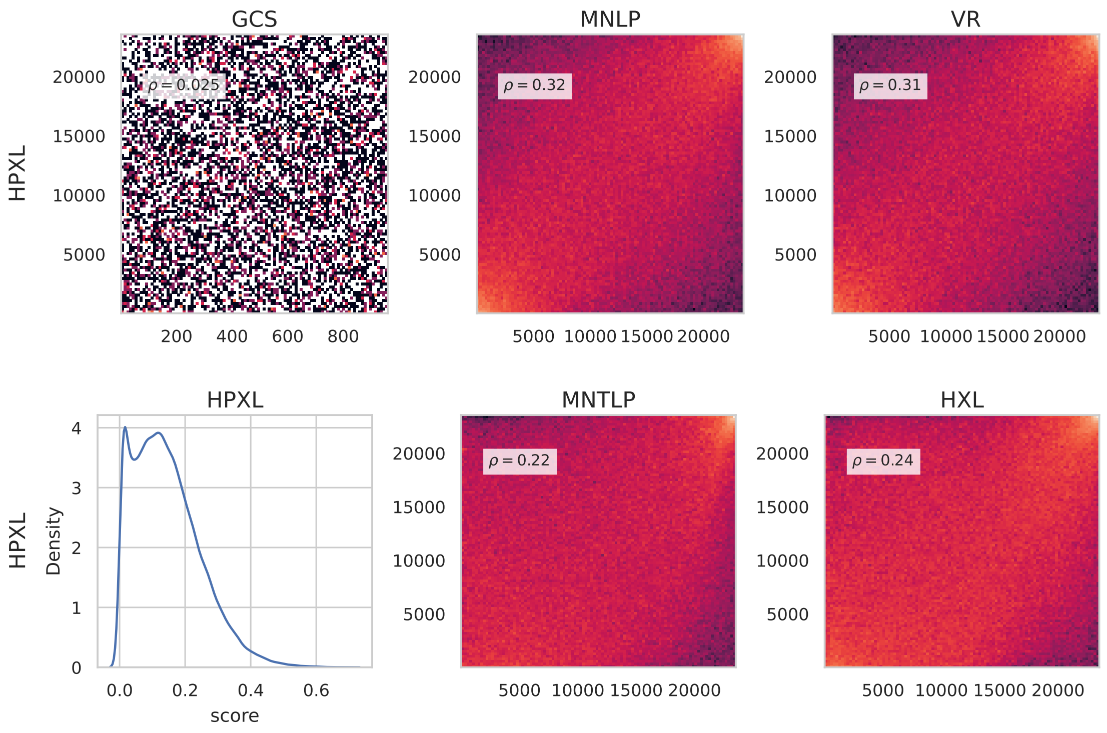


Figure 3.16: Pair-wise rank comparisons of HPXL on MultiRC

3.4.4.5 Diversity

Following Yuan, Lin, and Boyd-Graber [YLB20] we use the Jaccard-similiarity $J(T_1, T_2)$ between two sets of tokens T_1 and T_2 where we assume $|T_2| \gg |T_1|$ and T_2 to represent the token distribution to measure the diversity of T_1, i.e. to measure the representativeness of T_1 for T_2.

$$J(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|} \tag{3.30}$$

We report $J(T_{L_t}, T_{U_t})$ as "Diversity L_t" and $J(T_{L_{-1:t}}, T_{U_t})$ as "Diversity $L_{-1:t}$", where T_S is the union of all tokens in instances in S. U_t is the set of unlabeled instances at time t, L_t the instances which are being labeled at iteration t, and $L_{-1:t}$ that have been labeled from iteration -1 to (including) iteration t.

Results In Figure 3.17 one can see the diversity over time for all different active learning methods, both oracle-based and non-oracle-based approaches. As one would expect, GCS and HXLDD result in the most diverse batches. Interestingly, they do not produce more diverse batches than RANDOM on MultiRC. This indicates either that the selected batches are large enough to cover the diversity when selecting randomly or that the diversification does not work as well as it could. In the case of GCS, this adds another piece of evidence that the used representation is optimal. On Movie Reviews however, GCS does manage to construct significantly more diverse batches than RANDOM, meaning that the representation seems better. On Movie Reviews the relative size of the gap to the next method is much smaller than for MultiRC, giving another lead that diversity could be a bigger issue for the latter. HXL and MNTLP result in relatively less heterogeneous batches on both datasets, which as already mentioned in subsubsection 3.4.4.2 "Diversifying Oracles", this is not surprising since the scores produced by these methods are less influenced by random factors like model initialization or the warm-start dataset,since they use the ground truth labels instead of the predicted classes and therefore exhibit less noise. More noise would contribute to more diverse batches. MNLP and VR are producing less diverse batches on Movie Reviews than on MultiRC. Additionally, the variance of those two methods is smaller on Movie Reviews than on MultiRC. This could indicate that the model's predictions are more stable on Movie Reviews, leading to less noise and therefore less diverse batches. Interestingly, HPXL produces batches with a comparable diversity to RANDOM on Movie Reviews but not on MultiRC. Together with the fact that HPXL performs well on Movie Reviews but worse than random on MultiRC, we can conclude that HPXL finds a good trade-off of good scores and diversity on Movie Reviews but not on MultiRC and works better for skewed label distributions.

Overall, the lower scores for Movie Reviews compared to the ones on MultiRC are caused by the longer sequence in MultiRC, leading more quickly to almost full coverage of all tokens.

The fact that the order produced diversity by the different acquisition function for the current batch L_t and the order of diversity of the labeled dataset $L_{-1:t}$ does not differ and is increasing shows that the selected batches always contain a great number of new tokens, meaning that not only the batches but also the labeled dataset is diverse.

3.4.4.6 Batch Construction

Following the hypothesis that the diversity of the constructed batches is an issue on MultiRC, we conduct another experiment to find supporting evidence. To prove this, one has to find a diversification method that performs better than RANDOM. Just getting closer to random than by taking the top-k instances could also mean that the scores are less important for the selection. We choose HPXL and HXL to test an alternative batch construction method since those are the best performing non-oracles and oracle methods on Movie Reviews. Furthermore we assume that their good results indicate that the main issue in MultiRC is the batch construction.

The batches are sampled by dividing the range of scores into $2k$ bins of equal width and sampling from each of the top-k bins one instance each. The intuition is that, as one can see from Figure 3.16 and Figure 3.11, most of the instances get assigned a score in the lower half of the score range. By discarding them we ensure that only select instances that are reasonable useful. Additionally, randomly selecting from each bin ensures diversity in terms of the score.

Results In Figure 3.18 and Figure 3.19 we compare the performance results of the two different sampling methods. Figure 3.20 depicts the diversity of the selected batches and the labeled dataset (for details see subsubsection 3.4.4.5 "Diversity").

Performance-wise in the case of Movie Reviews, we can say that using stratified sampling instead of top-k sampling for batch construction reduces the performance but stays slightly above RANDOM. Meanwhile, it only increases the diversity for HXL but not for HPXL. One has to emphasize that the diversity of the constructed batches of HXL is much lower than for HPXL using top-k sampling. That supports the hypothesis that non-oracle acquisition functions generate more diverse batches by having a greater source of randomness through the prediction of the model. By adding randomness through the stratified sampling the influence of the scores is reduced but diversity is increased. This shows for Movie Reviews that diversity is not an issue since trading off the effect of the scores towards more diverse batches hurts performance.

On MultiRC the picture is similar yet different. Stratified sampling increases diversity massively and puts it close but below the token performance of RANDOM. In parallel, the performance increases for both approaches and gets closer to RANDOM. This is particularly the case for the token F1 but in combination with HXL also for the macro F1 of the auxiliary head. Since the performance stays below RANDOM, one cannot conclude if stratified sampling reduces the influence of the utility scores and is therefore getting closer to RANDOM. An alliterative explanation is that HXL with stratified sampling results in higher scores than with top-k sampling due to useful yet diverse instances. If the

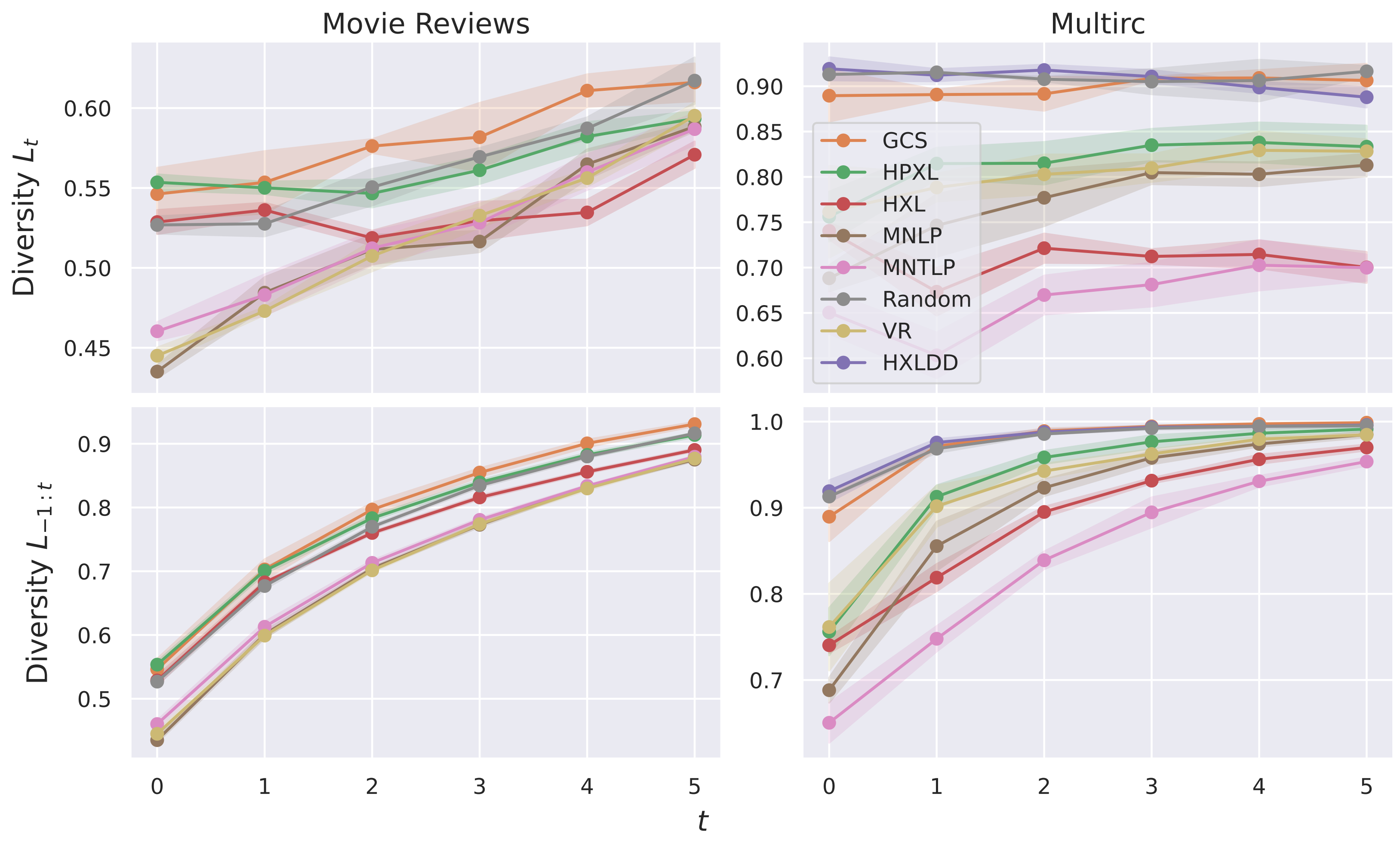


Figure 3.17: Diversity of selected instances.

performance had risen above random, one could have stated that the utility scores are helpful but the selected batches produced by top-k sampling lack diversity.

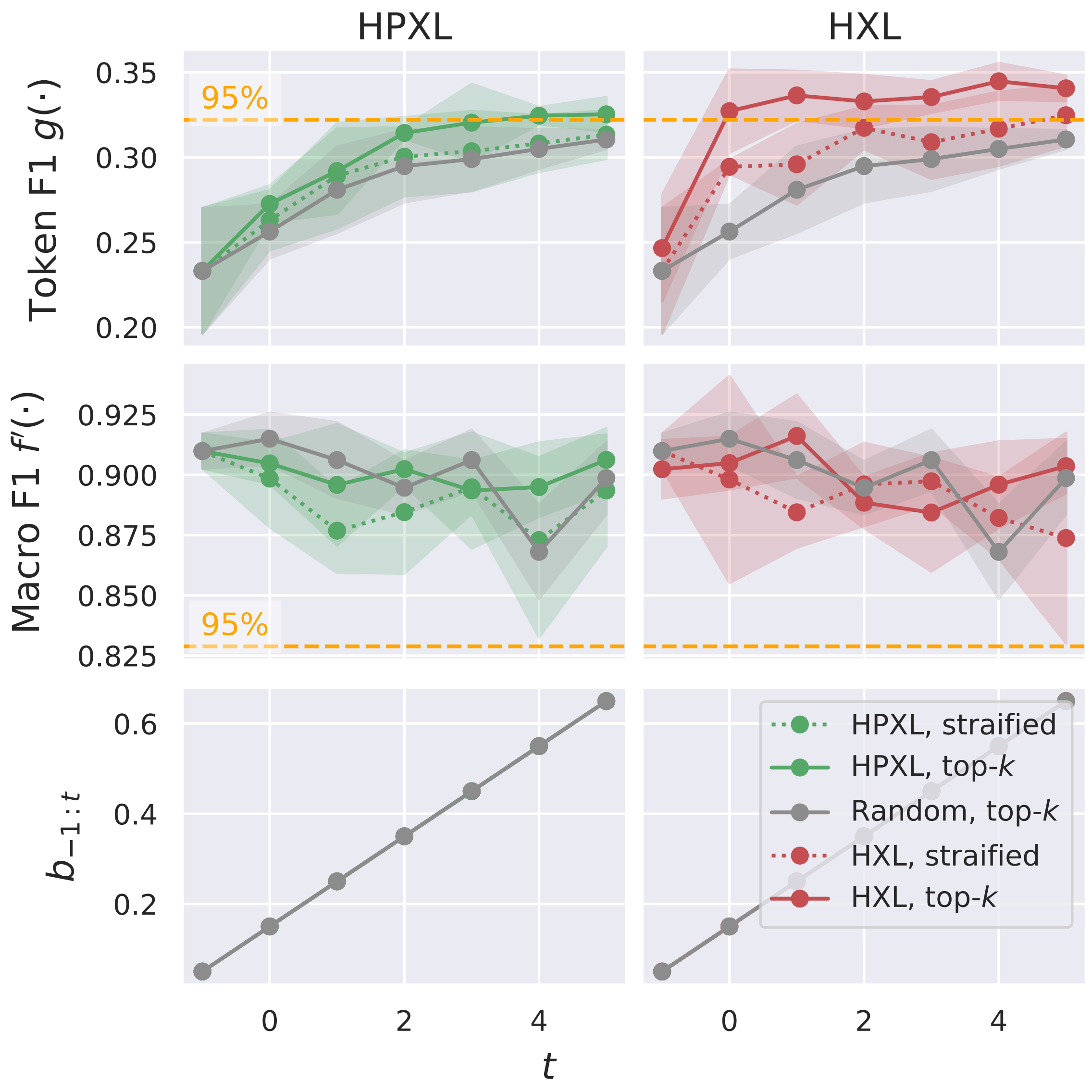


Figure 3.18: Comparison of top-k and stratified batch construction on Movie Reviews

3.4.4.7 Batch Size

To investigate the hypothesis that the larger batch size of the selected instances b_t in case of MultiRC contributes to the diversity issue and hinders performance we re-run HXL and HPXL on MultiRC and change the warm-start fraction b_{-1} and the batch size b_t to roughly match the absolute number of instances selected as in case Movie Reviews. Table 3.7 lists the hyperparameters used in former experiments and the adapted ones.

Table 3.7: Standard and adapted b_ts for MultiRC to match Movie Reviews.

Dataset	b_{-1}		b_t		$b_{-1:5}$	
	rel.	abs.	rel.	abs.	rel.	abs.
Movie Reviews	0.050	80	0.100	160	0.65	1040
MultiRC (standard)	0.020	480	0.040	961	0.26	6247
MultiRC (adapted)	0.003	73	0.007	189	0.26	1082

Results Figure 3.21 shows the outcome of the experiment in terms of performance and diversity. We provide the results of the former experiment (standard settings) up to $t = 1$. The diversity is only reported for the active learning iterations $t >= 0$, since the instances are randomly selected in the warm-start iteration ($t = -1$) and therefore do not reflect the behavior of the active learning strategies. Note that (only here) we report the size of the labeled dataset instead of the active learning iteration in contrast to figures earlier on in this work.

One can observe that HPXL with batch size $b_t = 0.007$ results in comparable token performance to Random and to the run using the standard setting. This means that the batch size does not have a significant influence for HPXL, for the tested settings. HXL using $b_t = 0.007$, however, performs worse than Random with the same step size and the performance gap is bigger than for HXL and Random with $b_t = 0.04$. Additionally, HXL using $b_t = 0.007$ results in worse result performance than when using $b_t = 0.04$. A plausible explanation could be that the top scores are not very reliable, since larger, selected batches contain more candidates. This is mitigating the influence of wrongly scored instances.

The diversity of the labeled dataset $L_{-1:t}$ in the $b_t = 0.007$ setting approaches the diversity of the runs using $b_t = 0.04$ for all methods. Interestingly, this is not the case for batch diversity. Batch diversity is much lower for all methods in the small batch setting than in the large base setting. One has to bear in mind that this does not reflect the diversity introduced per instance in the selected batch but the overall representatives of the batch regarding the unlabeled dataset. Therefore, the scores of smaller batches are naturally smaller. A possible extension to this measure is normalizing the scores by the batch size. In this case, the diversity introduced by each instance is bigger when using smaller batches.

Generally, we can conclude that using smaller batch sizes does not increase the diversity of the labeled dataset $L_{-1:t}$ and therefore does not provide considerable impact. This is also apparent from the performance results.

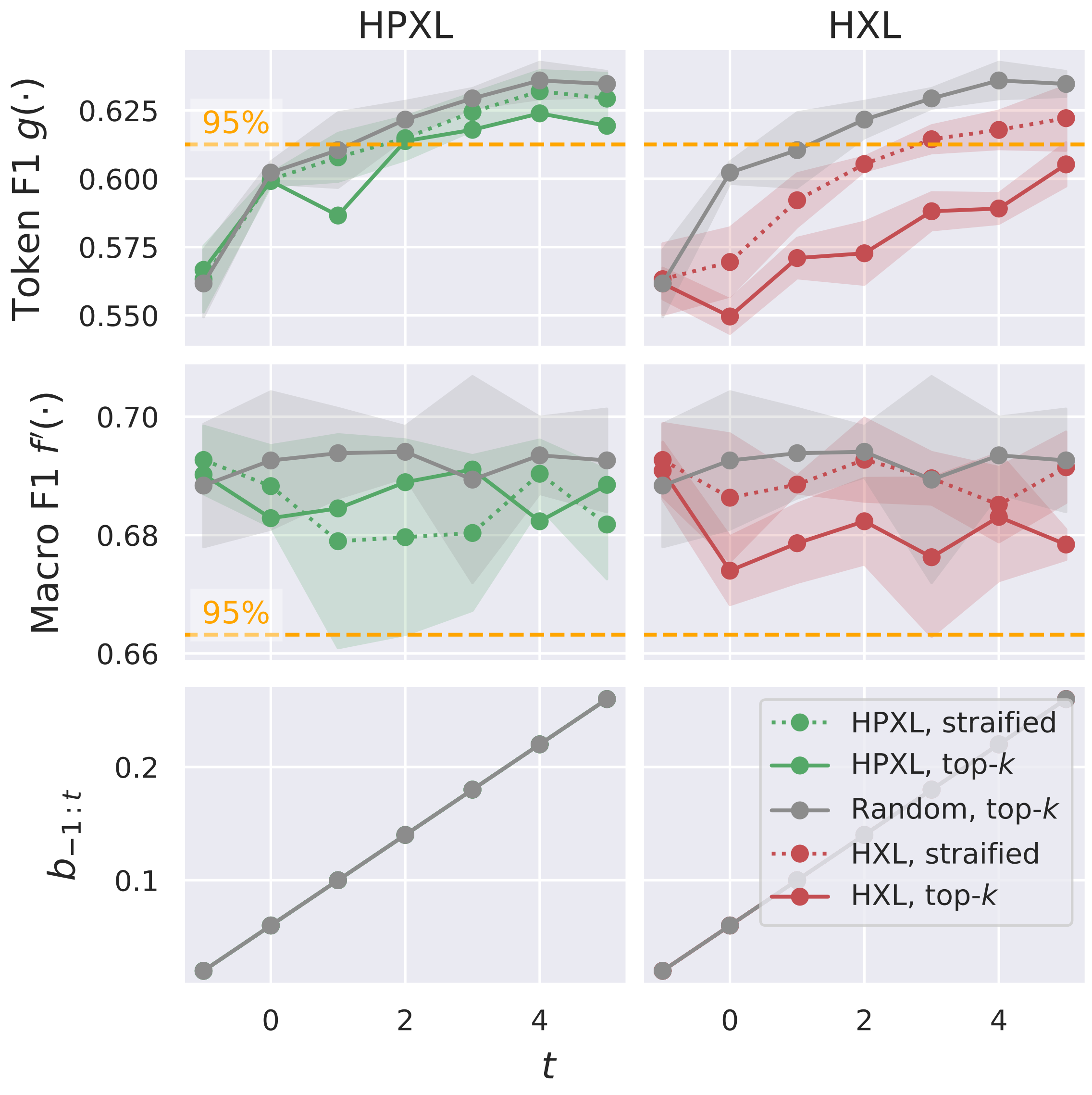


Figure 3.19: Comparison of top-k and stratified batch construction on MultiRC

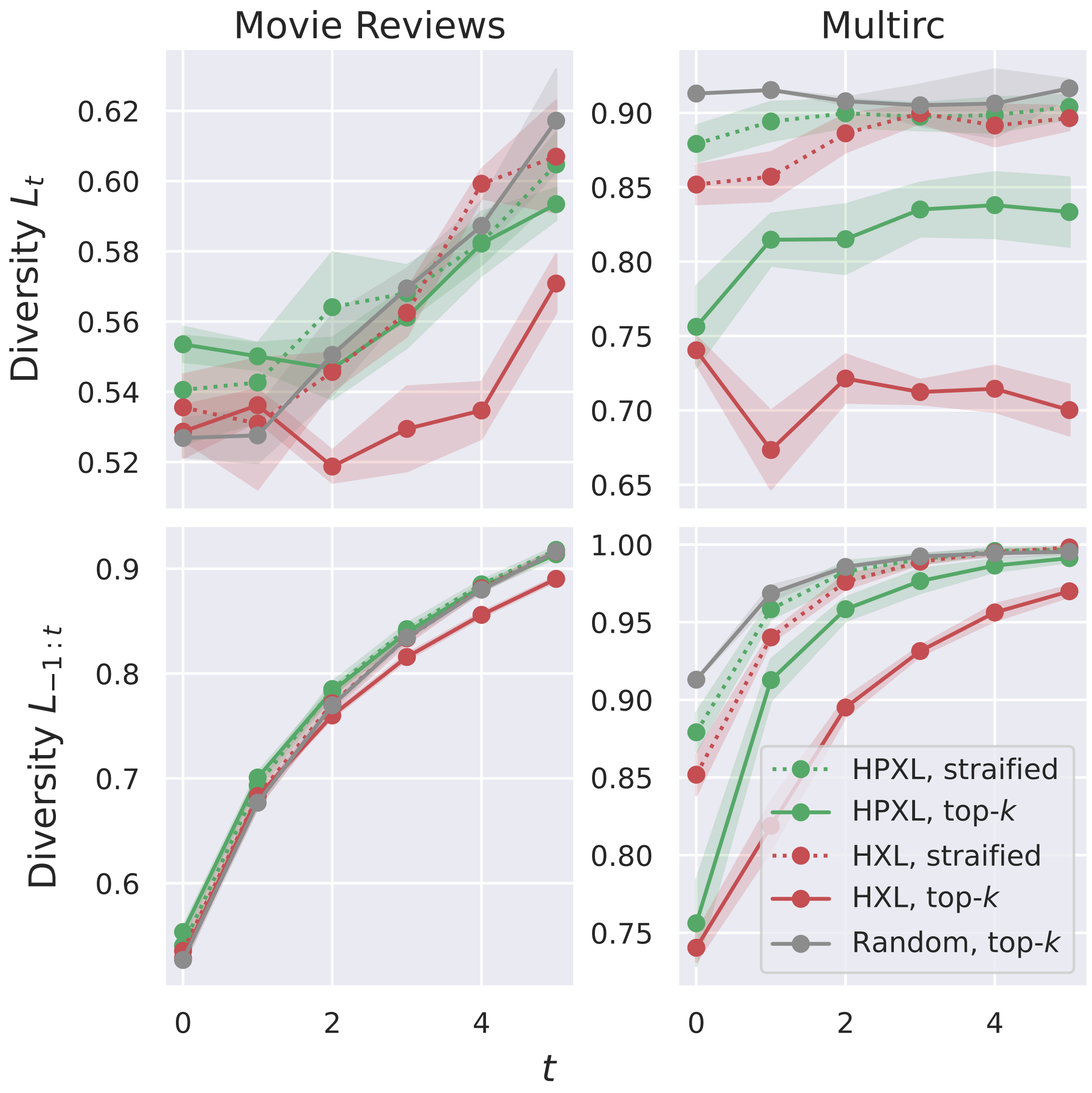
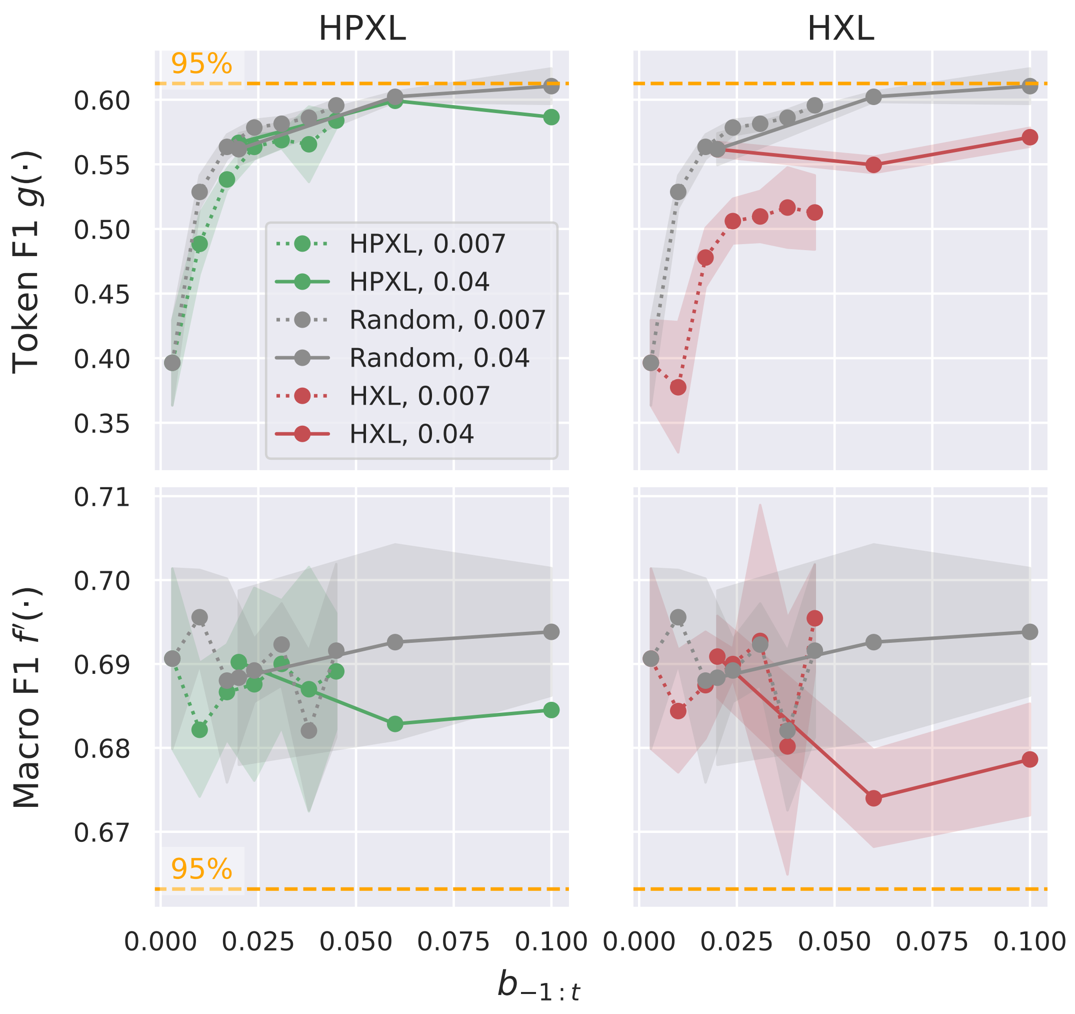
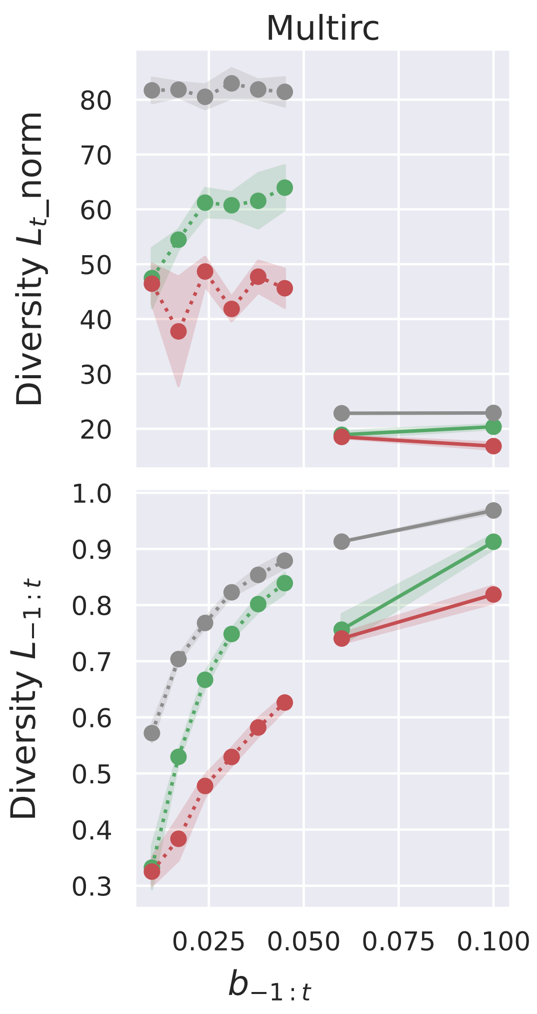


Figure 3.20: Diversity comparison of batch construction methods (top-k and stratified)



(a) Performance



(b) Diversity

Figure 3.21: Results for HPXL and HXL for setting with $b_t = 0.007$ (adapted) and $b_t = 0.04$ (standard).

3.4.4.8 Auxiliary Head Uncertainty as an Indicator for Hardness of Instances

We end our experiments with a slightly unrelated experiment. One potential failure case, already mentioned before, with the least certain methods is that they only work well if the predicted class is the actual class. This is due to the fact that if the model is highly certain about a wrong prediction, it will not select the instance to be labeled. Under the assumption that there are sentences that are harder to understand by the model, e.g. because they contain rare words, one can argue that the hardness of the auxiliary task $f'(\cdot)$ is correlated with the hardness of the rationale generation head $g(\cdot)$. Since in our scenario we assumed that the true labels $\mathbf{y}^*$ for the auxiliary task are given, we can use the uncertainty predicted by the auxiliary head for the true class as an acquisition function:

$$\mathbf{x}^* = \text{argmax}_{\mathbf{x}}\, 1 - p(\mathbf{y}^*|\mathbf{x}) \tag{3.31}$$

We call this *auxiliary least true certain* (AuxLTC) [4]. In this experiment, we pre-train the model only using task-level annotation $\mathbf{y}^*$ and then selects all instances in one batch. This gives some indication if this is a useful direction to investigate further.

Results Figure 3.22 shows the performance results. On Movie Reviews, the token F1 is worse than Random, whereas on MultiRC it is comparable. The impression that the auxiliary head uncertainty on the true class does not correlate much is confirmed by the ranking comparisons of the other methods, presented in Figure 3.23 and Figure 3.24. We can make three interesting observations: firstly for both datasets, the scores are almost all very close to zero or zero, where the majority is at 0. Looking at the raw numbers, there are scores in between but just very few. A score of 0 means that the auxiliary head predicts the ground truth with almost 100% certainty and 1 means it predicts the instances wrong with very high probability. The second interesting fact is that the acquisition functions are not based on the explanation loss and therefore weight the tokens equally. This results in a slight negative correlation of the ranks for Movie Reviews (ρ around -0.25). For the loss-based and all methods on MultiRC, there is no correlation ($\rho \approx 0$). Lastly, the horizontal stripes of the rank plots on MultiRC show that many instances get exactly the same score leading to the same ranks. This means that the distributions around 0 and 1 are very narrow leading to numerical collisions.

Overall, one can state that the auxiliary head is not a suitable indicator of the uncertainty of the explanation head if already trained on a fully labeled dataset.

[4] In our implementation this is erroneously called auxiliary least true uncertain

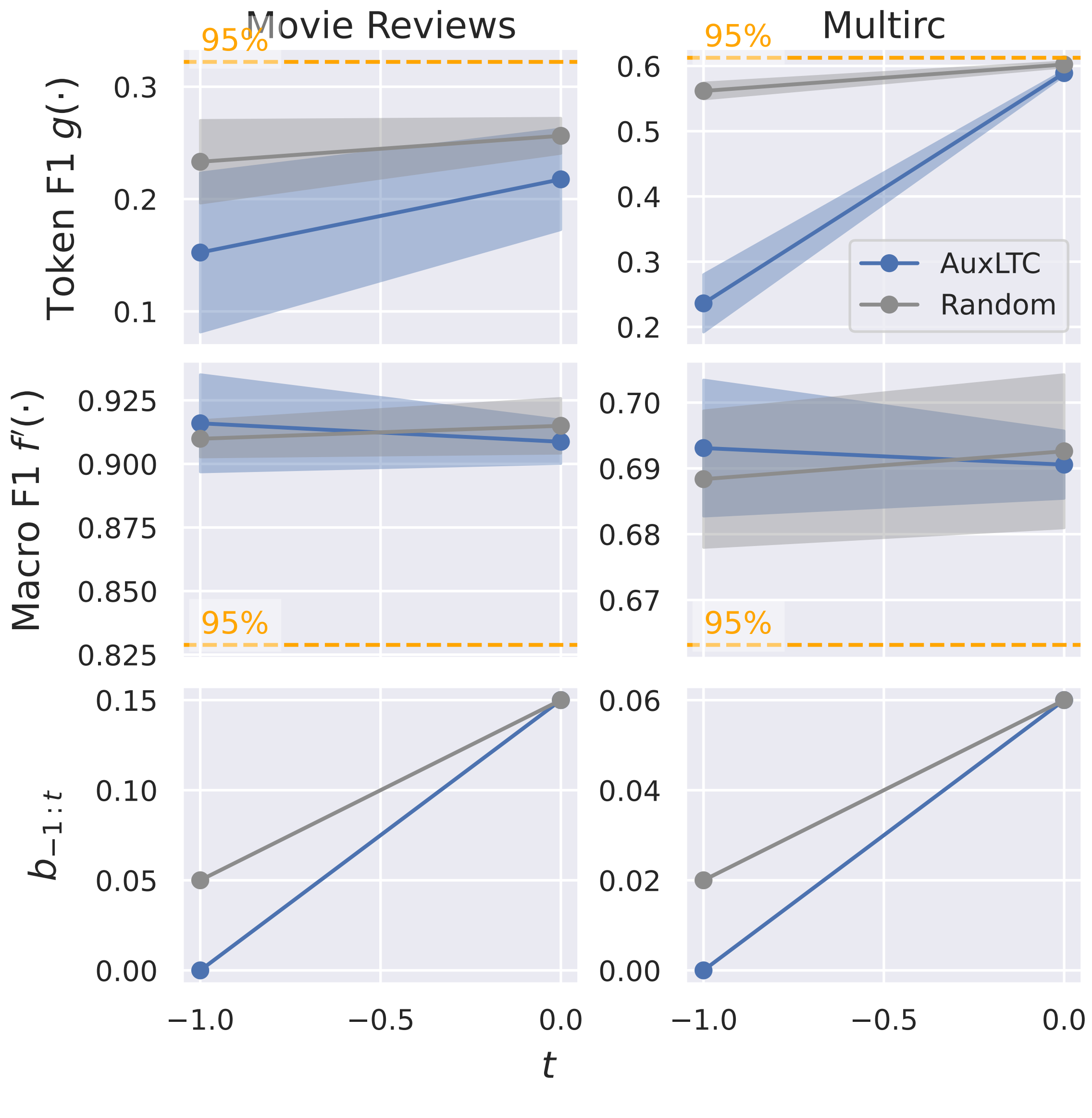


Figure 3.22: AuxLTC on Movie Reviews and MultiRC

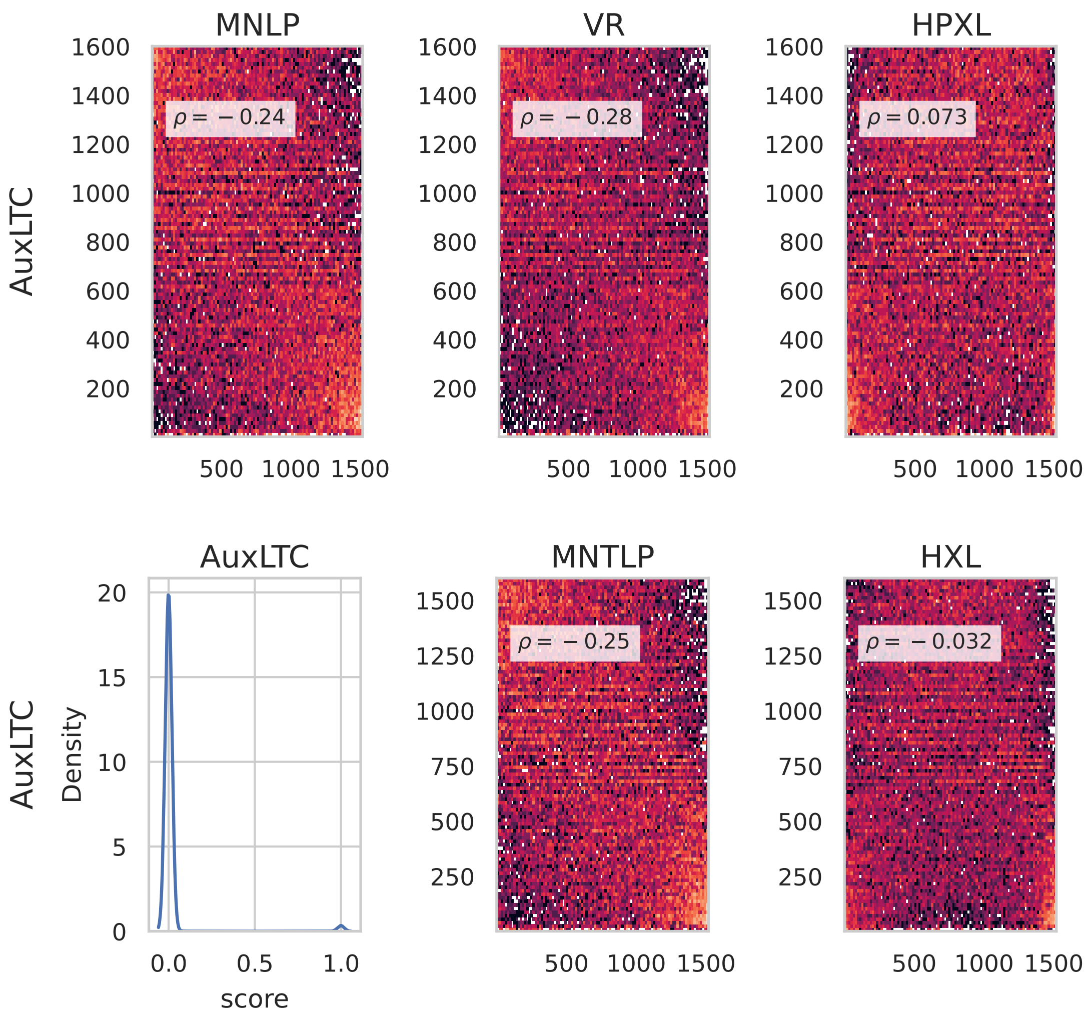


Figure 3.23: Pair-wise rank comparisons of AuxLTC on Movie Reviews

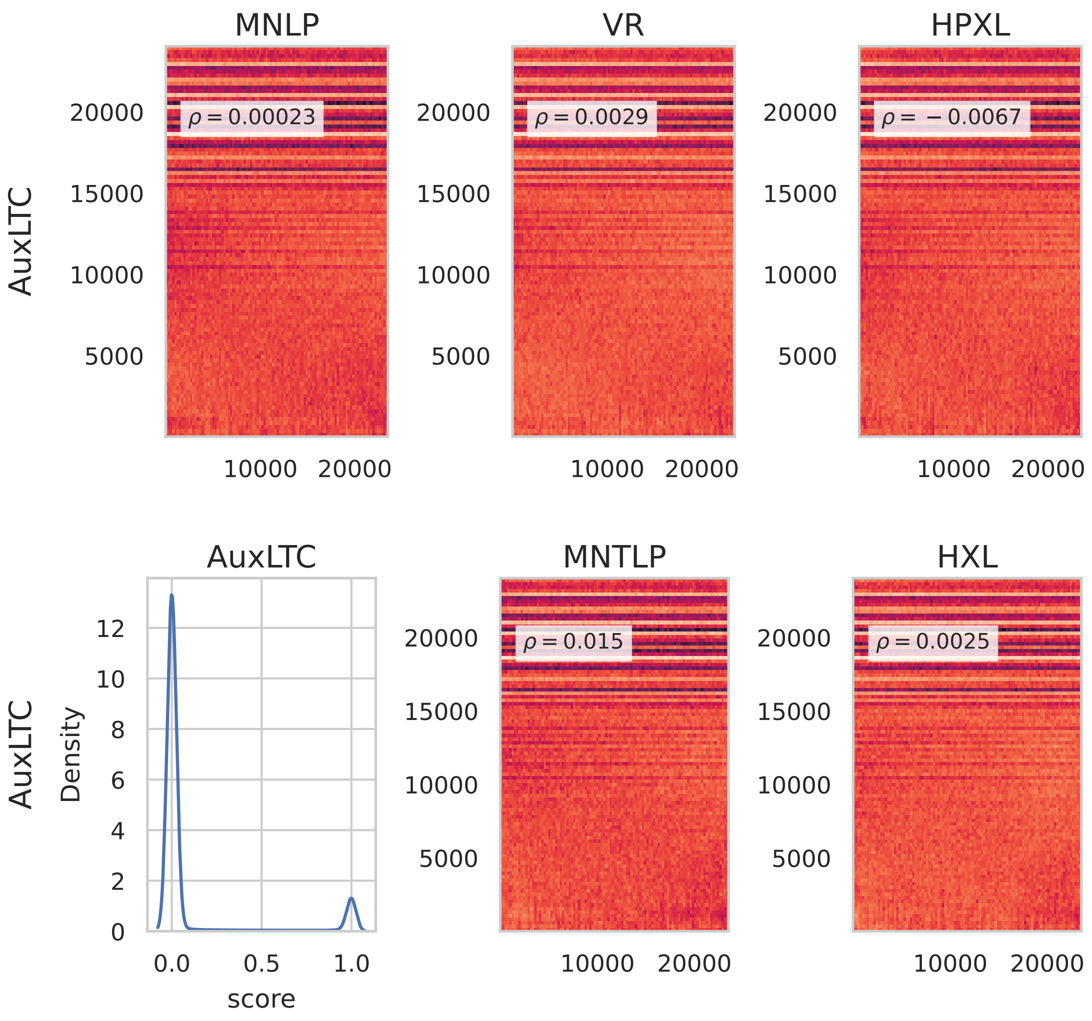


Figure 3.24: Pair-wise rank comparisons of AuxLTC on MultiRC

Chapter 4

Closure

4.1 Conclusion

In this work, we investigated:

(i) the degree to which the performance of ExPred, as a representative of pipeline explain-then-predict models, depends on the amount of rationale annotations available during training (**RQ1**).

(ii) if weak supervision, here using the example of pre-training on weak labeled produced by Lei et al. (u), can reduce the number of strongly rationale-labeled instances needed during training (**RQ2**).

(iii) if we can improve through labeling specifically useful instances using active learning (**RQ3**).

RQ1 We show that even with only a fraction of instances annotated with rationales from the original dataset, ExPred still achieve good performance (within 95% of the performance when using 100% annotation). On Movie Reviews, between 60% and 70% of the rationale annotations are needed, which is equivalent to around 960 and 1100 instances, respectively. On MultiRC, ExPred achieves 95% of the performance at 100% supervision with 10% to 20% of that annotate data, i.e. 2400 to 4800 instances, respectively. (For more details see section 3.2 "Preliminary Analysis: The Baseline and its Need for Rational Supervision".)

RQ2 The results on weak supervision are mixed: Using Lei et al. (u) as weak labeler and the simple weak supervision scheme of pre-training on the weak and fine-tuning on the strong data ($\text{NN}_{weak \rightarrow strong}$) provides only on Movie Reviews a considerable benefit, reducing the fraction of instance needed to achieve 95% performance from 70% to 5%. On MultiRC, the weakly labeled instances do not help but can hurt the performance. This is most likely the case due to the low performance of the weak labeler. (Further details can be found in section 3.3 "Learning from Weakness: Semi-Supervision using Weak Labels".)

RQ3 As emphasized in [LLW19], selecting a suitable active learning method without testing it is an unsolved problem and poses a considerable challenge to the practical application of active learning. We experience the same: all examined off-the-shelf methods (MNLP, VR using MCD, and GCS) did not consistently outperform the RANDOM baseline. On Movie Reviews, VR achieves slightly better results than RANDOM, but the performance does not justify the additional training cost. In the case of MultiRC, the performance even drops below RANDOM when using any of the three methods. We hypothesize that for MultiRC the bad performance is an issue regarding the diversity of the selected batches. (For more detailed description refer to section 3.4 "Active Learning")

In our subsequent investigation of the active learning methods, we learn in subsubsection 3.4.4.1 that for both datasets the utility for the training of the instances differs and therefore a successful active learning method can exist.

For Movie Reviews, we can successfully design active learning approaches (HXL and MNTLP) that outperform RANDOM, under the assumption that the ground truth labels are available before asking the oracle for the labels. This approach is not practical but generates the insight that the uncertainty estimates provided by the classification layer do not reflect the uncertainty regarding the unknown label. This is also true for the uncertainty estimates generated by Bayesian Inference (MCD, VR). For MultiRC, even with perfect information, these approaches fail. (See subsubsection 3.4.4.1.)

Moreover, we are able to design a practical, non-oracle approach that successfully outperforms the RANDOM baseline on Movie Reviews by weighting the uncertainty of the tokens in a sequence non-uniformly as done in the formulation of the training loss. On MultiRC, this method does not improve over RANDOM either. (See subsubsection 3.4.4.4.)

Our analysis on representatives and diversity of selected batches and the resulting datasets, in subsubsection 3.4.4.5 and subsubsection 3.4.4.2, indicate that on MultiRC no method produces a more diverse selection than random sampling. On Movie Reviews only GCS and HPXL do. In conjunction with the performance results, we observe that for Movie Reviews higher diversity is in some cases correlated with better performance but not always. Since on MultiRC none of the investigated methods performs better than RANDOM, we can not draw a clear conclusion if the diversity itself leads to the higher token performance. The increased token performance is either caused by more diversity or because the more diverse methods rely less strongly on the utility scores and are therefore closer to the performance of random selection.

Hence, the diversity issue could also be due to the large batch sizes used on MultiRC. We investigate in subsubsection 3.4.4.7 this issue and find out that at least using the same absolute batch sizes as on Movie Reviews does not lead to considerable improvements. This could be different when using very small batch sizes but would make the application of active learning non-feasible.

The comparison of the ranks produced by the different approaches suggests what also the performance results show: which instances provide more utility to the training is less ambiguous on Movie Reviews than on MultiRC and therefore easier to measure. In general, all methods agree mostly on which instances provide almost no useful information but produce conflicting results on the top ranks. (See subsubsection 3.4.4.3 for more details.)

The unsuccessful attempt to use the auxiliary head as an indicator for the instances' utility shows that the hardness of the instances for the task-level predictions is not correlated to the hardness of the explanation generation. (See subsubsection 3.4.4.8.)

To summarize, we show that applying active learning for annotating rationales performs reasonably on the sentiment analysis task Movie Reviews when using an acquisition function that weighs the token non-uniformly. On MultiRC, none of the tested methods and variations improves over RANDOM. Potential issues are unreliable uncertainty estimates and non-representative selected batches.

4.2 Limitations and Future Work

The presented results only apply to EXPRED in combination with MultiRC and Movie Reviews. In order to investigate if the results generalize to other explain-then-predict models and datasets, one has to extend the experiments, e.g. to other datasets from the ERASER benchmark and models like FRESH. In general, adding a statistical test to determine if the observed results are statistically significant or not could increase the explanatory power of the work. This, however, is out of focus of this work because we aim to provide the first step of exploration.

Our first exploratory steps in this works show for the weak supervision approaches, it is a potentially promising line of research to test more sophisticated schemes like FWL and other, potentially multiple sources of weak labels, such as post-hoc methods like attention scores or input gradients. These sources could then be combined using SNORKEL.

Regarding active learning, there are several interesting directions left for investigation. However, the main concerns are creating better utility scores and constructing more representative batches. To achieve the former, one should test if MCD-methods improve when increasing the number of forward passes, i.e. increasing the size of the simulated ensemble, and explore other utility functions like BALD. Investigating the latter includes finding better sequence representations to measure diversity on and exploring active learning schemes that combine diversity and uncertainty like APLS or BADGE. All these methods have to be adapted in order to reflect the insight that a non-uniform weighting of the tokens' uncertainty estimates produces better results.

Acknowledgments

During the work on my master thesis, I received a lot of support, both on a professional and on a personal level.

First of all, I would like to thank my supervision and examiner, Prof. Avishek Anand, for giving me the opportunity to work on this topic of active research and gain insights into the world of academia. Special thanks also to Zijian Zhang, who provided the implementation of ExPred, supported me on extending its functionality, and together with Koustav Rudra and Prof. Avishek Anand gave me very useful feedback in our weekly meetings. Again, thank you. Many thanks to Prof. Wolfgang Nejdl for being my second examiner and to the other members of the workgroup for technical and personal support.

Especially sincere thanks to my parent who gave the opportunity to study computer science in the first place, always supported me, and never doubted my success.

And a very warm thank you to all my friends and fellow students who accompanied and supported me on this journey: Sarah, Imke, Lennart, Jan, Noah, Max, Jonas, Leo, Jooke, Malte and Rasmus. Thank you for all the impulses when I got stuck, the very welcome distractions, and your emotional support.

Bibliography

[Ash+19] Jordan T. Ash et al. "Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds". en. In: Sept. 2019. URL: https://openreview.net/forum?id=ryghZJBKPS (visited on 03/24/2021).

[BAT19] Jasmijn Bastings, Wilker Aziz, and Ivan Titov. "Interpretable Neural Predictions with Differentiable Binary Variables". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2963–2977. DOI: 10.18653/v1/P19-1284. URL: https://www.aclweb.org/anthology/P19-1284 (visited on 03/03/2021).

[BCB16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv:1409.0473 [cs, stat]* (May 2016). arXiv: 1409.0473. URL: http://arxiv.org/abs/1409.0473 (visited on 07/15/2021).

[Bel+18] William H. Beluch et al. "The Power of Ensembles for Active Learning in Image Classification". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. ISSN: 2575-7075. June 2018, pp. 9368–9377. DOI: 10.1109/CVPR.2018.00976.

[Cam+18] Oana-Maria Camburu et al. "e-SNLI: Natural Language Inference with Natural Language Explanations". en. In: *Advances in Neural Information Processing Systems* 31 (2018). URL: https://proceedings.neurips.cc/paper/2018/hash/4c7a167bb329bd92580a99ce422d6fa6-Abstract.html (visited on 07/16/2021).

[Cho+14] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *arXiv:1406.1078 [cs, stat]* (Sept. 2, 2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078 (visited on 09/08/2021).

[Cho+21] Hyunjin Choi et al. "Evaluation of BERT and ALBERT Sentence Embedding Performance on Downstream NLP Tasks". In: *arXiv:2101.10642 [cs]* (Jan. 26, 2021). arXiv: 2101.10642. URL: http://arxiv.org/abs/2101.10642 (visited on 09/01/2021).

[Cla+19] Christopher Clark et al. “BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 2924–2936. DOI: 10.18653/v1/N19-1300. URL: https://aclanthology.org/N19-1300 (visited on 07/16/2021).

[Deh+18] Mostafa Dehghani et al. “Fidelity-Weighted Learning”. en. In: Feb. 2018. URL: https://openreview.net/forum?id=B1X0mzZCW (visited on 03/16/2021).

[Dev+19] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv:1810.04805 [cs]* (May 2019). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805 (visited on 05/04/2020).

[DeY+20] Jay DeYoung et al. “ERASER: A Benchmark to Evaluate Rationalized NLP Models”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 4443–4458. DOI: 10.18653/v1/2020.acl-main.408. URL: https://www.aclweb.org/anthology/2020.acl-main.408 (visited on 03/03/2021).

[Ein+20] Liat Ein-Dor et al. “Active Learning for BERT: An Empirical Study”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 7949–7962. DOI: 10.18653/v1/2020.emnlp-main.638. URL: https://www.aclweb.org/anthology/2020.emnlp-main.638 (visited on 03/03/2021).

[GB10] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. en. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. ISSN: 1938-7228. JMLR Workshop and Conference Proceedings, Mar. 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html (visited on 08/09/2021).

[GF17] Bryce Goodman and Seth Flaxman. “European Union regulations on algorithmic decision-making and a "right to explanation"”. In: *AI Magazine* 38.3 (Oct. 2017). arXiv: 1606.08813, pp. 50–57. ISSN: 2371-9621, 0738-4602. DOI: 10.1609/aimag.v38i3.2741. URL: http://arxiv.org/abs/1606.08813 (visited on 07/16/2021).

[GIG17] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. “Deep Bayesian Active Learning with Image Data”. In: *arXiv:1703.02910 [cs, stat]* (Mar. 2017). arXiv: 1703.02910. URL: http://arxiv.org/abs/1703.02910 (visited on 03/24/2021).

[GS19] Daniel Gissin and Shai Shalev-Shwartz. "Discriminative Active Learning". en. In: *arXiv:1907.06347 [cs, stat]* (July 2019). arXiv: 1907.06347. URL: `http://arxiv.org/abs/1907.06347` (visited on 03/24/2021).

[He+16] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ISSN: 1063-6919. June 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[Hig+16] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". en. In: (Nov. 2016). URL: `https://openreview.net/forum?id=Sy2fzU9gl` (visited on 07/15/2021).

[Hin+12] Geoffrey E. Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv:1207.0580 [cs]* (July 2012). arXiv: 1207.0580. URL: `http://arxiv.org/abs/1207.0580` (visited on 08/09/2021).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. URL: `https://doi.org/10.1162/neco.1997.9.8.1735` (visited on 08/06/2021).

[Hua+16] Jiaji Huang et al. "Active Learning for Speech Recognition: the Power of Gradients". en. In: *arXiv:1612.03226 [cs, stat]* (Dec. 2016). arXiv: 1612.03226. URL: `http://arxiv.org/abs/1612.03226` (visited on 03/24/2021).

[Jai+20] Sarthak Jain et al. "Learning to Faithfully Rationalize by Construction". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 4459–4473. DOI: `10.18653/v1/2020.acl-main.409`. URL: `https://www.aclweb.org/anthology/2020.acl-main.409` (visited on 03/03/2021).

[JW19] Sarthak Jain and Byron C. Wallace. "Attention is not Explanation". In: *NAACL-HLT*. 2019. DOI: `10.18653/v1/N19-1357`.

[Kha+18] Daniel Khashabi et al. "Looking Beyond the Surface: A Challenge Set for Reading Comprehension over Multiple Sentences". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 252–262. DOI: `10.18653/v1/N18-1023`. URL: `https://aclanthology.org/N18-1023` (visited on 07/16/2021).

[KKK16] Been Kim, Rajiv Khanna, and Oluwasanmi O. Koyejo. "Examples are not enough, learn to criticize! Criticism for Interpretability". en. In: *Advances in Neural Information Processing Systems* 29 (2016). URL: `https://papers.nips.cc/paper/2016/hash/5680522b8e2bb01943234bce7bf84534-Abstract.html` (visited on 05/10/2021).

[LBJ16] Tao Lei, Regina Barzilay, and Tommi Jaakkola. "Rationalizing Neural Predictions". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 107–117. DOI: 10.18653/v1/D16-1011. URL: https://www.aclweb.org/anthology/D16-1011 (visited on 03/03/2021).

[Leh+19] Eric Lehman et al. "Inferring Which Medical Treatments Work from Reports of Clinical Trials". 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 3705–3717. DOI: 10.18653/v1/N19-1371. URL: https://www.aclweb.org/anthology/N19-1371 (visited on 11/18/2020).

[LLW19] David Lowell, Zachary C. Lipton, and Byron C. Wallace. "Practical Obstacles to Deploying Active Learning". In: *arXiv:1807.04801 [cs, stat]* (Nov. 1, 2019). arXiv: 1807.04801. URL: http://arxiv.org/abs/1807.04801 (visited on 05/31/2021).

[LMJ17] Jiwei Li, Will Monroe, and Dan Jurafsky. "Understanding Neural Networks through Representation Erasure". In: *arXiv:1612.08220 [cs]* (Jan. 2017). arXiv: 1612.08220. URL: http://arxiv.org/abs/1612.08220 (visited on 07/15/2021).

[Ma+19] Xiaofei Ma et al. "Universal Text Representation from BERT: An Empirical Study". In: *arXiv:1910.07973 [cs]* (Oct. 23, 2019). arXiv: 1910.07973. URL: http://arxiv.org/abs/1910.07973 (visited on 09/01/2021).

[Mar+21] Diego Marcos et al. "Contextual Semantic Interpretability". en. In: *Contextual Semantic Interpretability*. Ed. by Hiroshi Ishikawa et al. Vol. 12625. Computer Vision – ACCV 2020. ISSN: 1611-3349 Num Pages: 18 Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 351–368. ISBN: 978-3-030-69538-5. DOI: 10.1007/978-3-030-69538-5_22. URL: http://link.springer.com/10.1007/978-3-030-69538-5_22 (visited on 05/05/2021).

[Mil18] Tim Miller. "Explanation in Artificial Intelligence: Insights from the Social Sciences". In: *arXiv:1706.07269 [cs]* (Aug. 2018). arXiv: 1706.07269. URL: http://arxiv.org/abs/1706.07269 (visited on 05/04/2021).

[Mol20] Christoph Molnar. *Interpretable machine learning ; a guide for making black box models explainable*. English. OCLC: 1233321581. 2020. ISBN: 978-0-244-76852-2.

[MPL19] Tom McCoy, Ellie Pavlick, and Tal Linzen. "Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3428–3448. DOI: 10.18653/v1/P19-1334. URL: https://www.aclweb.org/anthology/P19-1334 (visited on 05/09/2021).

[Nar+20] Sharan Narang et al. "WT5?! Training Text-to-Text Models to Explain their Predictions". In: *arXiv:2004.14546 [cs]* (Apr. 2020). arXiv: 2004.14546. URL: http://arxiv.org/abs/2004.14546 (visited on 07/15/2021).

[NB19] Nikita Nangia and Samuel R. Bowman. "Human vs. Muppet: A Conservative Estimate of Human Performance on the GLUE Benchmark". In: *arXiv:1905.10425 [cs]* (June 2019). arXiv: 1905.10425. URL: http://arxiv.org/abs/1905.10425 (visited on 05/06/2021).

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162 (visited on 08/06/2021).

[Raj+19] Nazneen Fatema Rajani et al. "Explain Yourself! Leveraging Language Models for Commonsense Reasoning". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4932–4942. DOI: 10.18653/v1/P19-1487. URL: https://aclanthology.org/P19-1487 (visited on 07/16/2021).

[Rat+17a] Alex Ratner et al. *Weak Supervision: The New Programming Paradigm for Machine Learning · Stanford DAWN*. en. June 2017. URL: https://dawn.cs.stanford.edu/2017/07/16/weak-supervision/ (visited on 04/16/2021).

[Rat+17b] Alexander Ratner et al. "Snorkel: rapid training data creation with weak supervision". In: *Proceedings of the VLDB Endowment* 11.3 (Nov. 2017), pp. 269–282. ISSN: 2150-8097. DOI: 10.14778/3157794.3157797. URL: https://doi.org/10.14778/3157794.3157797 (visited on 07/23/2021).

[Ros20] Corby Rosset. *Turing-NLG: A 17-billion-parameter language model by Microsoft*. en-US. Feb. 2020. URL: https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/ (visited on 07/23/2021).

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1135–1144. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778 (visited on 05/05/2021).

[RSG18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Anchors: High-Precision Model-Agnostic Explanations”. en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). Number: 1. ISSN: 2374-3468. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11491 (visited on 07/15/2021).

[Rus+15] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. en. In: *International Journal of Computer Vision* 115.3 (Dec. 2015), pp. 211–252. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-015-0816-y. URL: http://link.springer.com/10.1007/s11263-015-0816-y (visited on 05/06/2021).

[Sak20] Tetsuya Sakai. “On Fuhr’s Guideline for IR Evaluation”. en. In: *ACM SIGIR Forum* 54.1 (2020), p. 8.

[San+20] Victor Sanh et al. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv:1910.01108 [cs]* (Feb. 2020). arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108 (visited on 08/09/2021).

[SC08] Burr Settles and Mark Craven. “An analysis of active learning strategies for sequence labeling tasks”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP ’08. USA: Association for Computational Linguistics, Oct. 2008, pp. 1070–1079. (Visited on 05/27/2021).

[Set12] Burr Settles. *Active Learning*. eng. Synthesis lectures on artificial intelligence and machine learning 18. OCLC: 809876977. San Rafael, Calif.: Morgan & Claypool, 2012. ISBN: 978-1-60845-725-0.

[Sha+14] Ali Sharif Razavian et al. “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”. In: 2014, pp. 806–813. URL: https://www.cv-foundation.org/openaccess/content_cvpr_workshops_2014/W15/html/Razavian_CNN_Features_Off-the-Shelf_2014_CVPR_paper.html (visited on 08/09/2021).

[She+17] Yanyao Shen et al. “Deep Active Learning for Named Entity Recognition”. In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 252–256. DOI: 10.18653/v1/W17-2630. URL: https://aclanthology.org/W17-2630 (visited on 08/25/2021).

[She+21] Artem Shelmanov et al. “Active Learning for Sequence Tagging with Deep Pre-trained Models and Bayesian Uncertainty Estimates”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 1698–1712. URL: https://www.aclweb.org/anthology/2021.eacl-main.145 (visited on 05/27/2021).

[Smi+17] Daniel Smilkov et al. “SmoothGrad: removing noise by adding noise”. In: *arXiv:1706.03825 [cs, stat]* (June 2017). arXiv: 1706.03825. URL: http://arxiv.org/abs/1706.03825 (visited on 07/15/2021).

[SR19] Raphael Schumann and Ines Rehbein. “Active Learning via Membership Query Synthesis for Semi-Supervised Sentence Classification”. In: *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. CoNLL 2019. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 472–481. DOI: 10.18653/v1/K19-1044. URL: https://aclanthology.org/K19-1044 (visited on 07/28/2021).

[SS18] Ozan Sener and Silvio Savarese. “Active Learning for Convolutional Neural Networks: A Core-Set Approach”. en. In: Feb. 2018. URL: https://openreview.net/forum?id=H1aIuk-RW (visited on 06/02/2021).

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks”. In: *arXiv:1703.01365 [cs]* (June 2017). arXiv: 1703.01365. URL: http://arxiv.org/abs/1703.01365 (visited on 07/15/2021).

[Thi29] L. Thiessé. *Œuvres complètes de Voltaire*. Œuvres complètes de Voltaire Bd. 48. Baudouin, 1829. URL: https://books.google.de/books?id=zl5AzgEACAAJ.

[Tho+18] James Thorne et al. “FEVER: a Large-scale Dataset for Fact Extraction and VERification”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 809–819. DOI: 10.18653/v1/N18-1074. URL: https://aclanthology.org/N18-1074 (visited on 07/16/2021).

[Vas+17] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[Wal+21] Eric Wallace et al. “Universal Adversarial Triggers for Attacking and Analyzing NLP”. In: *arXiv:1908.07125 [cs]* (Jan. 2021). arXiv: 1908.07125. URL: http://arxiv.org/abs/1908.07125 (visited on 07/15/2021).

[WGS20] Eric Wallace, Matt Gardner, and Sameer Singh. “Interpreting Predictions of NLP Models”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*. Online: Association for Computational Linguistics, Nov. 2020, pp. 20–23. DOI: 10.18653/v1/2020.emnlp-tutorials.3. URL: https://www.aclweb.org/anthology/2020.emnlp-tutorials.3 (visited on 05/09/2021).

[Wil92] Ronald J. Williams. “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. In: *Machine Language* 8.3-4 (May 1992), pp. 229–256. ISSN: 0885-6125. DOI: 10.1007/BF00992696. URL: https://doi.org/10.1007/BF00992696 (visited on 07/22/2021).

[Wol+20] Thomas Wolf et al. “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[Yeh+18] Chih-Kuan Yeh et al. “Representer point selection for explaining deep neural networks”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pp. 9311–9321. (Visited on 07/15/2021).

[YLB20] Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. “Cold-start Active Learning through Self-supervised Language Modeling”. en. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, pp. 7935–7948. DOI: 10.18653/v1/2020.emnlp-main.637. URL: https://www.aclweb.org/anthology/2020.emnlp-main.637 (visited on 03/25/2021).

[ZE08] Omar Zaidan and Jason Eisner. “Modeling Annotators: A Generative Approach to Learning from Annotator Rationales”. In: *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii: Association for Computational Linguistics, Oct. 2008, pp. 31–40. URL: https://aclanthology.org/D08-1004 (visited on 07/16/2021).

[ZMW16] Ye Zhang, Iain Marshall, and Byron C. Wallace. “Rationale-Augmented Convolutional Neural Networks for Text Classification”. In: *arXiv:1605.04469 [cs]* (Sept. 2016). arXiv: 1605.04469. URL: http://arxiv.org/abs/1605.04469 (visited on 07/15/2021).

[ZRA21] Zijian Zhang, Koustav Rudra, and Avishek Anand. “Explain and Predict, and then Predict Again”. In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. WSDM ’21. New York, NY, USA: Association for Computing Machinery, Mar. 2021, pp. 418–426. ISBN: 978-1-4503-8297-7. DOI: 10.1145/3437963.3441758. URL: https://doi.org/10.1145/3437963.3441758 (visited on 07/23/2021).

Part I

Appendix I

4.1 Active Learning

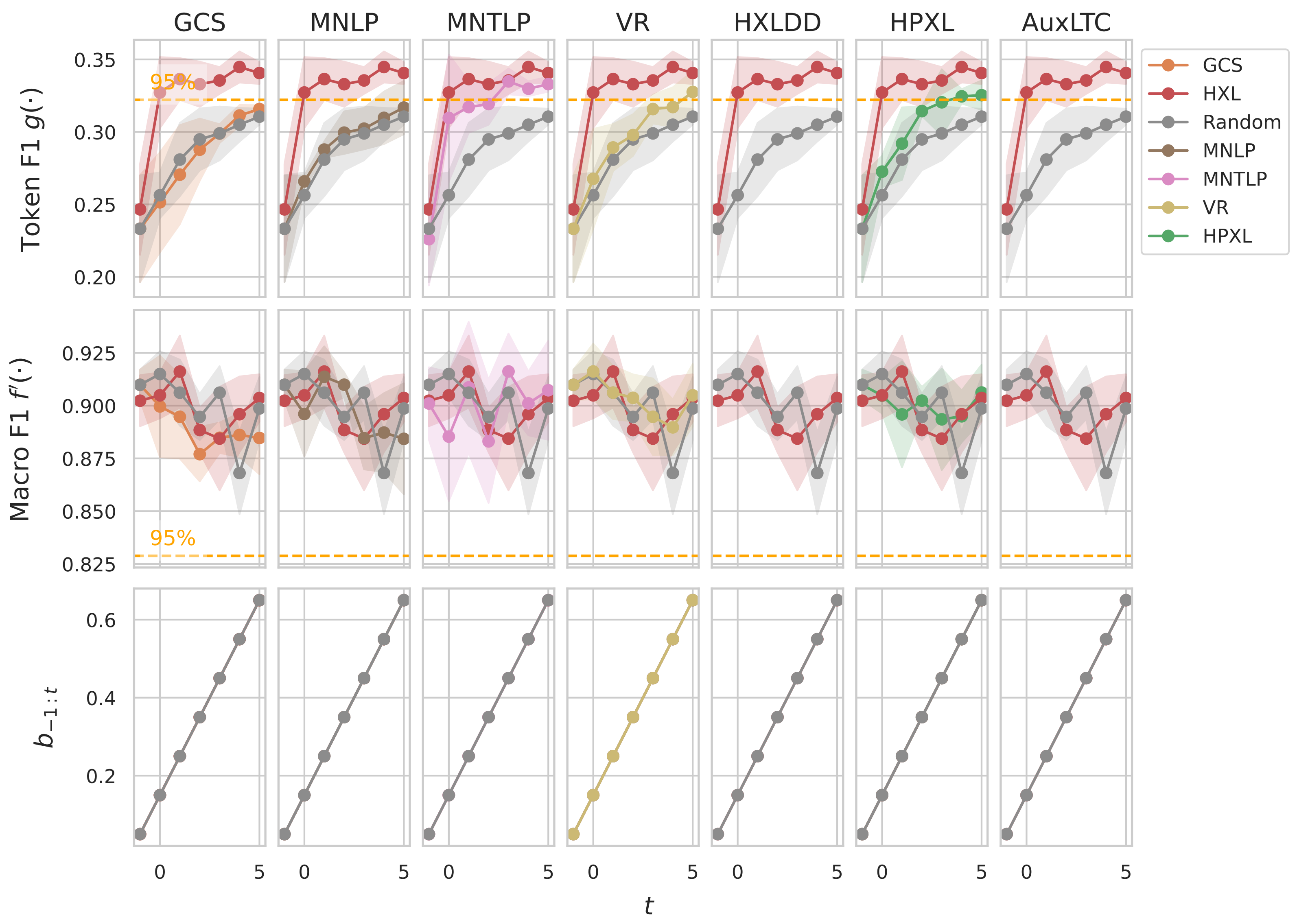


Figure 4.1: Performance results for all active learning methods on Movie Reviews using the same setting as described in subsection 3.4.2

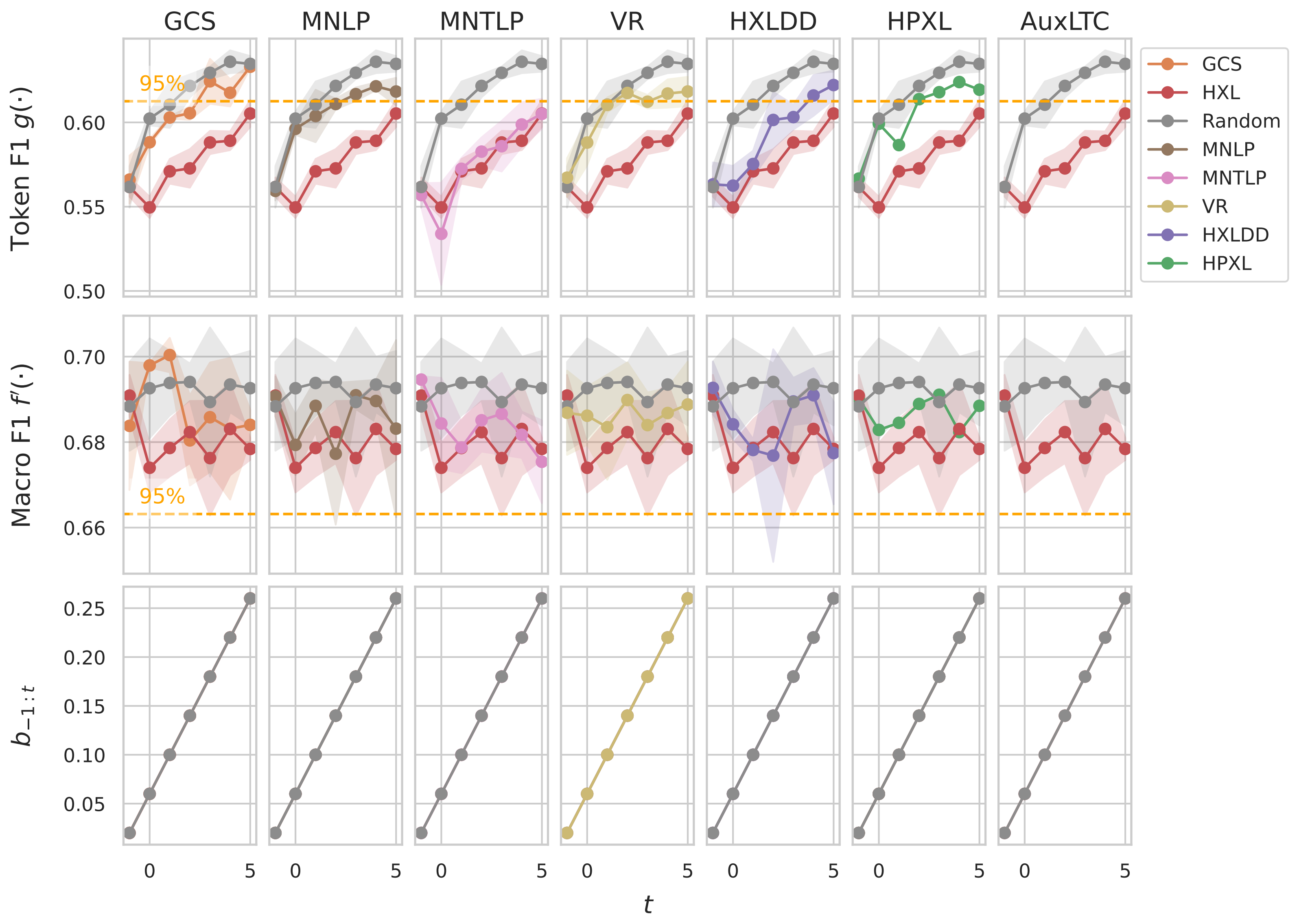


Figure 4.2: Performance results for all active learning methods on MultiRC using the same setting as described in subsection 3.4.2