Interpretability in Neural Information Retrieval

Lyu, L.

DOI
10.4233/uuid:fbce75ab-4dca-432e-9388-475993c60105

Publication date
2025

Document Version
Final published version

Citation (APA)
Lyu, L. (2025). *Interpretability in Neural Information Retrieval*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:fbce75ab-4dca-432e-9388-475993c60105

Interpretability in Neural Information Retrieval

Interpretability in Neural Information Retrieval

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on Monday, 24th of February 2025 at 10:00 o'clock

by

Lijun LYU

Master of Science in Internet Technologies and Information Systems,
Leibniz University Hannover, Germany,
born in Sichuan, China.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. ir. G.J.P.M Houben,	Delft University of Technology, promotor
Dr. A. Anand,	Delft University of Technology, promotor
Independent members:	
Prof. dr. A. van Deursen,	Delft University of Technology
Prof. dr. S. Verberne	Leiden University
Prof. dr. N. Ferro,	University of Padua, Italy
Prof. dr. M. Hagen,	Friedrich-Schiller-Universität Jena, Germany
Prof. dr. A.E. Zaidman,	Delft University of Technology, reserve member

SIKS Dissertation Series No. 2025-12
The work in the thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems. This research has been supported by German Research Foundation (DFG), under the Project IREM with grant No. AN 996/1-1.

Keywords: interpretable-machine-learning, information retrieval

Cover: Agathe Balayn and David Maxwell

Style: TU Delft House Style, with modifications by Moritz Beller
`https://github.com/Inventitech/phd-thesis-template`

Printed by:

ISBN 978-94-6518-007-6

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

What I cannot create, I do not understand.

Richard P. Feynman

Contents

Acknowledgments

This has been a long and tough journey, yet full of joy, excitement and growth. Reflecting on the past four years, not only did I manage to complete this thesis–a collection of work I am deeply proud of, but also I enjoyed every moment of the pursuit, whether sweet or bitter. The time I spent in Hannover, Delft, and Edinburgh has left me with memories I will always cherish. Remarkably, I almost did not sacrifice any sleep or weekend! Have I grown into who I aspired to be? Not yet, but I am undoubtedly a step closer. For this, I am forever grateful to this PhD journey and my younger self who dared to embrace this challenge. Along the way, I have been fortunate to meet many wonderful people who shared parts of this path with me, bringing support, laughter, and friendship. To all of them, I owe my heartfelt thanks.

To my supervisor Avishek, I still don't know why you believed in me even when I didn't believe in myself. You once told me that this shouldn't be about publication, but rather a process of building confidence. I believe beyond earning the degree, this has been my biggest growth, as today I can easily argue with you. Thanks to you for being encouraging, patient, and always taking my opinion and interest into first consideration. You never 'supervised' me as a professor, instead, you have been 'coworking' as a colleague or a friend. For that, you have my deep respect.

To my colleagues in the group–Max, Jonas, Mandeep, Yumeng, Zijian and Venky–thanks for so many insightful discussions. And to my Hannover pals–Shanshan, Huyen, Raneen, Simon, Nicholas and many more–thank you for the great company in Mensa (the sweet Mensa turned out to be what I missed the most from Hannover).

I could not be more grateful for WIS members, who welcomed me with warm hearts from my very first day in Delft. My deepest gratitude goes to Geert-Jan for agreeing to be my promotor under such special circumstances. Sole and Jie, you supported me in many ways, just like my second advisor. To all of you–Ujwal, Christos, George (1 and 2), Kyriakos, Andra, Ziyu, Lorenzo, Garrett, Gaole, Sara, Robin, Shreyan, Esra, Anne, and Nirmal–we share the grumbling over Echo craps, but more joy of coffee chats from our free machine to the fancy library, and countless pubs in lovely Delft of course! To my office buddies Aditya and Danning, we had many cynical discussions, but those were just fun. A special thanks to Daphne and Nadia, for keeping all of us on track with your hard work and thoughtfulness.

In particular, I would like to thank my brilliant friends Agathe and David, who also designed the cover of this thesis. You are the ones I can always turn to when I need support, advice and comfort. Of course, we also share so many laughs and moments on random days, on our trips in the south of France, in Glasgow, in St. Andrews. The same gratitude goes to my dear paranymphs, Alisa and Tianqi. Your friendship means a lot, and I appreciate the nice arts you introduced to me: NDT dances, Friends exhibition, movies, concerts, etc. Please keep increasing the aesthetics of my life.

One of the biggest benefits of PhD is we get to travel and meet many similar minds. The pandemic made it hard, but what a great surprise to bump into Klaus five years later at a random conference! And many others I won't list the names one by one, you made the boring conferences much more interesting. I also had the wildest time at Dagstuhl: endless drinking and board gaming after midnight; hiking slash castle exploration in the dark with Anja and Julius. Look forward to you being my tour guide in China!

I am grateful for many friends outside of work. Rico and Alex, thanks for so much delicious food, and UNLIMITED resources of entertainment, I enjoyed our movie nights and conversations at the dinner table, involving three languages makes it even more fun. Kaiyi and Akong, thank you for being there with me over the past ten years, even though we are far apart, we've always felt close. And to my college girls, now shining in their own fields, especially Liyao, you have never stopped inspiring me.

Ultimately, my gratitude to my family and Jurek is beyond words. Thank you for everything.

Lijun
Delft, Jan 2025

Summary

Neural information retrieval (IR) has transitioned from using classical human-defined relevance rules to leveraging complex neural models for retrieval tasks. While benefiting from advances in machine learning (ML), neural IR also inherits several drawbacks, including the opacity of the model's decision-making process. This thesis aims to tackle this issue and enhance the transparency of neural IR models. Particularly, our work focuses on understanding which input features neural ranking models rely on to generate a specific ranking list. Our work draws inspiration from interpretable ML. However, we also recognize the unique aspects of IR tasks, which guide our development of methods specifically designed to interpret IR models.

We begin with interpreting black-box text ranking models. Without access to the internal parameters, we employ simple surrogates to approximate the complex black-box model's prediction. Ideally, the surrogate model can offer a high-fidelity explanation if it approximates the complex model's output closely. Therefore, we come up with optimization techniques to directly maximize the approximation of surrogate models, so that they can closely simulate the complex model and generate similar (identical in an ideal scenario) rank lists.

Secondly, building intrinsically explainable models has been highly advocated over interpreting trained models due to several advantages. Following this approach, we explore the applicability of popular outcomes from interpretable ML to IR. Our findings indicate that we can leverage interpretable ML to minimize feature redundancy, making predictions in IR more transparent by utilizing a limited number of features.

However, despite their seemly great performance, we start to question if the previously explored interpretable models are really faithful. Specifically, we study the selector-predictor paradigm and demonstrate that the predictor incorporates additional information when making predictions, even though it appears that only the selected features are used as input. That is, the information about unselected features or the labels are leaked to the predictor, making the explanations unfaithful, because in such scenario, the prediction does not (entirely) come from the explanation. As a result, we propose formal definitions of leakage and introduce our faithful approach with theoretical guarantees. Initially developed for interpretable ML, our approach is also applicable to IR.

Therefore, this thesis contributes to interpretable neural IR, and ultimately interpretable ML. We develop our approaches on public benchmark datasets, and all of our source code is publicly available. With these two domains becoming increasingly intertwined, we hope our findings and insights in this thesis help construct transparent and effective systems for both two domains.

Samenvatting

Neurale informatie retrieval (IR) heeft de overgang gemaakt van het gebruik van klassieke, door mensen gedefinieerde relevantieregels naar het gebruik van complexe neurale modellen voor retrieval taken. Hoewel neurale IR profiteert van de vooruitgang in machine learning (ML), heeft het ook een aantal nadelen, waaronder de ondoorzichtigheid van het besluitvormingsproces van het model. Dit proefschrift heeft als doel dit probleem aan te pakken en de transparantie van neurale IR-modellen te verbeteren. In het bijzonder richt ons werk zich op het begrijpen op welke inputkenmerken neurale rangschikkingsmodellen zich baseren om een specifieke ranglijst te genereren. Ons werk is geïnspireerd op interpreteerbare ML. We erkennen echter ook de unieke aspecten van IR-taken, waardoor we methoden ontwikkelen die specifiek zijn ontworpen om IR-modellen te interpreteren.

We beginnen met het interpreteren van black-box tekstrangschikkingsmodellen. Zonder toegang tot de interne parameters gebruiken we eenvoudige surrogaten om de voorspelling van het complexe black-box model te benaderen. Idealiter kan het surrogaatmodel een waarheidsgetrouwe verklaring bieden als het de uitvoer van het complexe model nauwkeurig benadert. Daarom komen we met optimalisatietechnieken om de benadering van surrogaatmodellen direct te maximaliseren, zodat ze het complexe model nauwkeurig kunnen simuleren en vergelijkbare (identieke in een ideaal scenario) ranglijsten kunnen genereren.

Ten tweede wordt het bouwen van intrinsiek verklaarbare modellen sterk bepleit boven het interpreteren van getrainde modellen vanwege verschillende voordelen. Volgens deze benadering onderzoeken we de toepasbaarheid van populaire uitkomsten van interpreteerbare ML op IR Onze bevindingen geven aan dat we interpretable ML kunnen gebruiken om feature redundantie te minimaliseren, waardoor voorspellingen in IR transparanter worden door gebruik te maken van een beperkt aantal features.

Ondanks hun ogenschijnlijk goede prestaties beginnen we ons echter af te vragen of de eerder onderzochte interpreteerbare modellen wel echt getrouw zijn. We bestuderen het selector-predictor paradigma en tonen aan dat de predictor extra informatie meeneemt bij het doen van voorspellingen, ook al lijkt het alsof alleen de geselecteerde kenmerken als invoer worden gebruikt. Dat wil zeggen, de informatie over niet-geselecteerde kenmerken of de labels lekken op de een of andere manier naar de voorspeller, waardoor de uitleg ontrouw wordt, omdat in zo'n scenario de voorspelling niet (volledig) uit de uitleg komt. Daarom stellen we formele definities van lekken voor en introduceren we onze getrouwe benadering met theoretische garanties. Onze benadering is in eerste instantie ontwikkeld voor interpreteerbare ML, maar is ook toepasbaar op IR.

Daarom draagt dit proefschrift bij aan interpreteerbare neurale IR, en uiteindelijk aan interpreteerbare ML. We ontwikkelen onze benaderingen op openbare benchmark datasets en al onze broncode is openbaar beschikbaar. Nu deze twee domeinen steeds meer met elkaar verweven raken, hopen we dat onze bevindingen en inzichten in dit proefschrift helpen bij het bouwen van transparante en effectieve systemen voor beide domeinen.

1 Introduction

Information retrieval (IR) describes the process of finding relevant information that satisfies the user's search need from a usually large collection of corpus. It is one of the most user-centric systems ranging from general web search, to domain-specific and enterprise search etc. Take web search as an example: the user encodes her information need into a **query**–a usually short textual term that can be *under-specified* or *ambiguous*. Then, the search engine takes the query and applies some *ranking model* to compute the **relevance** scale of each item (e.g., document) within the collection to the query. Eventually, a list of ranked top-k relevant documents will be returned to the user, who determines the ground-truth relevance of a document. A desirable ranking model should have a high coherence with the user in terms of relevance judgment. Therefore, understanding the model's decision-making logic is essential to ensure it serves users' search needs. In the era of generative AI, where information retrieval (IR) serves as the initial step for numerous knowledge-intensive tasks [98], the need for transparent and reliable IR systems has become increasingly urgent.

1.1 Complex Ranking Models

Classical ranking models, such as BM25 [4] measure the relevance of a document to the issued query by the frequency of the query terms appearing in the document (i.e., *term frequency*). This type of static model is transparent and easily understandable for system developers and users; and has been well applied until the recent advances in neural models and natural language processing (NLP) [31, 126]. Modern ranking models employ neural components, such as transformer architecture [126] and BERT pre-training to learn the relevance judgments implicitly from the user-annotated labels and have shown noticeable performance boosts. Popular options include the **cross-encoder** design, where the query and document are combined into a single input, and the model predicts the relevance score as the output. Another approach is the **dual-encoder** design, which uses two separate models to encode the query and document individually, with the relevance score determined by measuring the similarity (e.g., semantic similarity) between the two representations. However, these learned neural ranking models have also inherited the

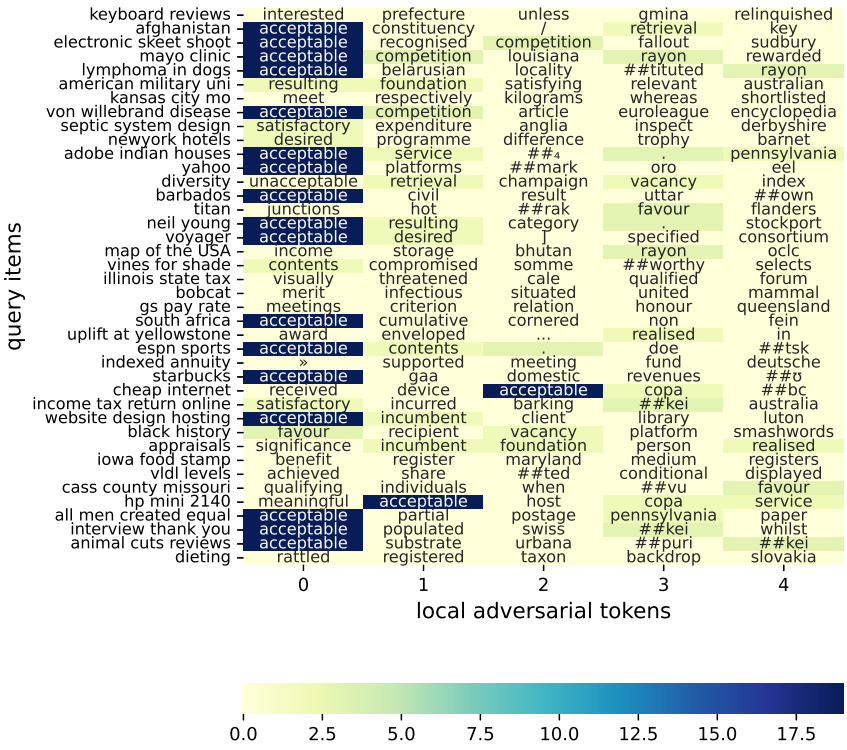


Figure 1.1: The five adversarial tokens added to the beginning of the highest-ranked document for 40 queries from ClueWeb09, selected by local ranking attack method to demote the document. Specific tokens frequently recur across queries. The frequency is denoted by the color.

disadvantage of opaqueness, a well-recognized and extensively studied issue in general machine learning (ML) [145].

With the ranking model being a black box, it is unclear how the decision of a single item or rank has been made. Data bias, such as gender and ethnicity bias, might potentially be encoded in the model and harm the reliability of the system. Short-cuts or hidden patterns might be the essential factor for model decision, without being noticed or aligned to the *right* reason [62]. Furthermore, neural models are known to be fragile towards imperceptible perturbations [130]. In previous work [133], we show that a small change (less than five words) made in a target document can promote or demote the rank of the document by a large margin. For instance, Figure 1.1 presents frequent words that can be added to shift the document rank by a BERT-based cross-encoder from the bottom-10 to the top-10 position within a 100-depth list in the ClueWeb09 dataset [25]. This highlights the brittle yet intriguing nature of complex neural models. Accordingly, interpreting model decisions in various applications, including language and computer vision (CV), have proven useful in identifying adversarial snippets, thereby enhancing the system's reliability and robustness. [40, 49, 74, 91].

1.2 Interpretability in Machine Learning

Before diving into interpretability in information retrieval (IR), we begin with a brief overview of interpretability in machine learning (ML), which mainly solves classification tasks for images and texts. This is crucial as advancements in interpretable ML have profoundly shaped the landscape of interpretable IR.

Interpretable ML aims to improve the interpretability or explainability[1] of complex ML models. Current efforts generally fall into two main categories: (1) Developing methods to interpret a complex trained model, resulting in what is termed as an **explanation**. These methods are typically referred to as **post-hoc methods**; (2) Designing models that are inherently interpretable to humans rather than interpreting a black-box model. These models provide explanations alongside predictions, driven by challenges in evaluating post-hoc methods and additional computation costs (usually including many runs of forward calls) for generating explanations. Such models are often referred to as **self-explainable**, **intrinsically interpretable**, or **interpretable-by-design** models, which are used interchangeably in this thesis.

Post-hoc methods typically interpret a particular model prediction by looking at the input instance. The resulting explanations can be (causal) rules (if X, then Y) [41, 107], or more commonly, attribute scores (or heatmaps) corresponding to the contribution of individual input features (e.g., pixels or tokens). Different methods compute the scores differently, such as using gradient-related scores (like integrated-gradient [105]) when model parameters are accessible, or employing permutation techniques (like Shapley values [82]) or simple surrogate approximation (like LIME [106]) otherwise.

On the other hand, current self-explainable models primarily focus on the explicit use of input features while treating the neural architectures as a black box. Specifically, the model identifies (either specified or learned) the important features and uses only those for prediction. These used features are considered **faithful** explanations [9, 24, 147] as they are directly derived from the model. Meanwhile. when a large input is reduced to a small set of crucial features[2], it becomes easier to discern which features are essential and which can be discarded. Unlike post-hoc heatmaps with continuous importance scores, this type of explanation can be seen as concrete binary masks. In principle, hard masks should offer clearer insights than heatmaps because the masked-out features are entirely excluded from the model's decision-making process, leaving a handful of features for user interpretation. However, the more features that are masked out—resulting in higher feature **sparsity**—the more likely the model's performance may suffer. Conversely, better explanations are achieved if the model maintains performance with a smaller set of features. Therefore, a model's interpretability degree is often evaluated using the performance-sparsity curve. Popular self-explainable models are constructed on the **selector-predictor** framework [24, 71, 141], where the selector conducts feature selection and the predictor performs downstream prediction with selected features. Usually, both modules are learned simultaneously. We point to the survey [45] for more detailed discussion.

[1] The terms *explainable* and *interpretable* are now used interchangeably in current literature.
[2] Note this differs from feature compression.

1

1.3 Interpretability in Information Retrieval

Prior to the emergence of neural IR, two types of ranking models were conventionally considered **intrinsically transparent**: the first type includes static retrieval models based on classical human-defined relevance factors (e.g., *term matching*). Models with closed form notations like BM25 [4] or RM3 [51] fall into this type; the second type comprises models based on inherently interpretable architectures like decision trees[3], Support Vector Machines (SVMs), etc., where the decision-making process or the contribution of features leading to the decision is understandable by users (developers). In IR domains, these two types of models are usually associated with particular tasks. For instance, the retrieval models are often used in **text ranking**, namely ranking documents[4] for a given query. Whereas the learnable interpretable models are typically studied in popular **LETOR Learning-to-Rank** (we denote as **LTR**) benchmarks [102]. This is a specific IR task where the query-document pair is represented by a vector of pre-computed numerical statistics (e.g., *term frequency*, *incoming URL frequency*, etc.). Nowadays, both types of models have been replaced by their complex neural counterparts for performance gains. As previously mentioned, the opaque nature of these neural components and the automatic learning process (using deep learning) create challenges in understanding which input elements make it relevant. This need for enhanced interpretability parallels the recent performance improvements seen with neural models in general machine learning (ML). Therefore in this thesis, we tackle the issues related to neural IR, or models with neural components. We categorize our contributions in a manner similar to those in interpretable ML, with each contribution focused on a specific IR task.

1.3.1 Post-hoc Interpretations

When a search query is submitted to a ranking model, the model presents a list of items (documents) ranked by their assigned relevance scores to the user. This multi-decision aggregation process contrasts with single-prediction tasks like classification. As a result, the scope of interpretation tasks expands into the following three folds:

- pointwise – why does the model predict doc_i as relevant or not relevant?
- pairwise – why does the model rank doc_i higher than doc_j ($doc_i \succ doc_j$)?
- listwise – why does the model rank $doc_1 \succ doc_2 \cdots \succ doc_n$?

The majority of works (particularly those adapted from interpretable ML) [55, 115] can answer the first two questions, while the last one concerns only the ranking scenario and thus requires additional designs than merely applying existing methods (e.g., LIME [106]) from interpretable ML.

Regardless of the pointwise, pairwise or listwise ranking decision, it can be explained from multiple aspects. That is, the resulting explanations are of different formats. For instance, it can be tokens (features if not for textual input) in the query or document highlighted by **attribute scores**. Those scores indicate the significance of each individual token to the ranking model for classifying doc_i as (ir)relevant, or doc_i more relevant than

[3]Note that large ensemble or deep trees are considered hard to interpret.
[4]Broadly, this can include images, audio, videos, etc., but we focus solely on document retrieval in this thesis.

Query: can you do yoga from a chair

Rank	Score	Document Text
1	9.37	10 Yoga Poses You Can Do in a Chair | Chair pose yoga, Chair yoga, Yoga poses ...

Query: can you do yoga from a chair

Rank	Score	Document Text
2	9.34	Chair Hip Opening And Strength Flow Yoga | Yoga Sequences, Benefits ...

Figure 1.2: Post-hoc explanation: attribute scores. The dark orange color highlights the important words in both query and document for classifying the relevance of the document.

Query: can you do yoga from a chair + {poses, guide, home, how, hip, sequence, learn}

Rank	Score	Document Text
1	9.37	10 Yoga Poses You Can Do in a Chair | Chair pose yoga, Chair yoga, Yoga poses ...
2	9.34	Chair Hip Opening And Strength Flow Yoga | Yoga Sequences, Benefits ...
3	9.31	Chair Seated Twists Yoga | Yoga Sequences, Benefits, Variations, and Sanskrit ...
4	9.31	5 Best Yoga Ball Chair for Homes and Offices (2021 Buying Guide) - Learn ...
5	9.29	Chair Yoga for Seniors: 8 Chair Yoga Poses Seniors Can Do Easily At Home ...

Figure 1.3: Post-hoc explanation: quey expansion. The expansion terms (green) are chosen such that a simple term-based ranker best approximates the rank list produced by a black-box model.

doc_j. Figure 1.2 showcases explaining the model prediction on one query-document pair with attribute scores. Token highlights can indeed offer some insights, but it is restricted from explaining listwise decisions. Moreover, previous works [46] show that token highlights seem ineffective in improving user understanding or even counter-intuitive, simply because they only answer *which tokens make the document relevant*, but not *what are the criteria for relevance judgment*. Another type of explanation is **generated** natural sentences, appreciated for their straightforward and understandable nature [144]. It requires human-crafted explanations to train a model for generating terms or sentences for a given rank list. The quality of the explanation highly relies on the model capability and training explanation datasets. In addition, generalizing to other ranking tasks with shifted domains can be problematic. Furthermore, the generated explanations are not guaranteed to align with the model prediction, and it is hard to verify because the model's reasoning may not always align with the user's either. This is particularly the case when the search queries are ambiguous and the retrieved documents from a large corpus can contain any or all of the meanings of a query. Therefore, to understand how a ranking model generates a particular ranking decision (from pointwise to listwise), it is important to first identify how the model understands the query. Namely, explaining ranking results is posed as query interpretation. Towards this, existing works [80, 116] tend to expand the short query terms by additional words. This technique is known as **query expansion** in the IR domain, where a model (e.g., RM3) applies simple heuristics to obtain a set of words to enrich the query, as indicated in Figure 1.3, so that the ranking performance can be hopefully improved. In return, the expanded query can serve as a more clarified explanation for the original query, and obtaining such expansion remains challenging and thus is the major contribution to query understanding. Note that this thesis does not cover all types of explanations, and for more notions and clarifications we point to our survey [5].

One of the major difficulties for post-hoc explanations is **evaluation**. As mentioned before, the discrepancy between the human and the model makes the manually annotated

explanations untruthful. In most cases, ground-truth datasets are unavailable. Instead, a logical hypothesis is that if the explanation is indeed the complete and necessary reason for the model's prediction, it should be able to produce the same results as the model. Consequently, a widely used evaluation metric measures how well the explanations can replicate the model's results. This so-called **fidelity** measure calculates the similarity between the predictions from the model and those from the explanation. For ranking outputs, the *correlation* factor is typically used as a similarity metric.

Limitations of current literature

Most current explanation methods in IR are merely borrowed from interpretable ML, which may not align well with the decision-aggregation output in IR. Consequently, the resulting explanations may lack clarity in understandable relevance factors.

1.3.2 Interpretability by Design

It is appealing to have an intrinsically interpretable model without much compromise in performance, in comparison to post-hoc interpretations. As mentioned in Section 1.2, current solutions mainly offer transparency in used features, rather than entirely transparent decision paths. This idea is particularly practical for the Learning-to-Rank (LTR) task because it works on structured datasets with numerical feature values, which can be kept or eliminated by applying some concrete binary mask.

Such process of identifying important input features has been previously studied and noted as **feature selection** in LTR. The goal is to select only features that matter—either align with human judgments or result in the best performance. In the former scenario, usually simple heuristics, such as Fisher or Laplace [47] scores, or if available, human annotations are applied; while the latter tends to discover features that maximize the performance of a particular model, and thus has become increasingly favored especially in the modern neural era.

Specifically, feature selection is addressed as a learning problem, as iterating each combination of features is NP-hard. It can be learned along with the model optimization via regularization terms such as L1 or L2 regularization [123]. This is particularly effective for LTR task because each input feature can be eliminated by tuning its weight to zero. However, most existing methods are constrained to classical ML models like decision trees or regression models, and the application of feature selection on neural LTR models is still unexplored.

Limitations of current literature

Most existing feature selection methods in IR are only designed for traditional LTR models, and thus do not satisfy the needs of neural LTR models.

1.4 Goals and Research Questions

In this thesis we lay out our contributions to interpretability for information retrieval in post-hoc and intrinsic diagrams. We start by interpreting black-box neural models for text ranking. The goal is to generate easy-to-understand (to developers) explanations

that can cover all ranking-specific aspects ranging from pointwise to listwise. Parallelly, we explore the possibilities of designing self-explainable ranking models. This is under-explored in IR, however, a popular research topic in ML. Therefore, our goal is firstly to investigate whether those research outcomes from ML are directly applicable to IR. Following that, we take a close look at those widely applied methods in ML, and carefully examine if their self-generated explanations are faithful to their predictions. We highlight the significance of this investigation because faithfulness is one of the most critical and fundamental requirements of explanations. Towards this, we aim at proposing our novel self-explainable model that is proven to be faithful. Even though this solution falls in the ML domain, we argue it is generally applicable to ranking targets as well. As a result, we aim to achieve the aforementioned research goals to improve the interpretability of neural IR and formulate the following research questions which will be studied in each chapter:

RQ1 How do we explain ranking-specific decisions from black-box text ranking models?

RQ2 Is interpretable ML applicable for building self-explainable ranking models?

RQ3 Are self-explainable models faithful and how to design theoretically guaranteed faithful models?

1.5 Contribution

This thesis comprises three main chapters, each focusing on one of the above research questions and altogether contributing to the field of interpretable IR and furthermore general machine learning. In terms of interpretation paradigm, we focus on post-hoc explanations in Chapter 2, and interpretable-by-design models in Chapter 3 and Chapter 4. In terms of specific retrieval tasks, we target on text ranking in Chapter 2, and LTR benchmark in Chapter 3. Note that Chapter 4 mainly solves tabular and image classification, but it is easily applicable to the LTR benchmark, due to the same data format. We summarize the main contributions of the thesis as follows:

- ✔ In Chapter 2, we address **RQ1** and interpret trained complex ranking models by query expansion, which shows one or multiple understandings of the black-box model to a single query. This shows the challenge in terms of manual evaluation due to the machine-human discrepancy.
- ✔ Based on query expansion, we employ multiple classical IR models (e.g., BM25) as the surrogates to explain complex black-box rankers for **RQ1**. This helps explain the ranking models by not just query specification, but also exact relevance factors.
- ✔ We directly optimize the fidelity via extending preference coverage maximization solved by linear programming in Chapter 2. Our solution called Multiplex results in potentially optimal explanations, addressing the research gap in high-fidelity listwise explanations.
- ✔ In Chapter 3, we introduce intrinsically interpretable ranking models via input feature selection to answer **RQ2**. We investigate all (to our best knowledge) available options proposed in the general ML field but remain unexplored in the IR domain.

1

We summarize an overview of interpretable ML and bridge the research gap between ML and IR. Practically, we also construct a benchmark conducting feature selection for neural LTR models.

✔ Chapter 3 answers **RQ2** from different diverse aspects closely related to ranking tasks, including interpretability, efficiency, robustness, and so on.

✔ In Chapter 4, inspired by the previous work in Chapter 3, we study **RQ3** and experimentally prove that existing methods in interpretable ML, despite being widely applied, are not faithful due to the occurrence of *leakage* during training.

✔ Chapter 4 introduces the first in literature the formal *leakage* concept to solve **RQ3**, including feature leakage and label leakage. Furthermore, we formally define the necessary conditions of leakage-free guarantee as a theoretical ground for constructing faithful models.

✔ We propose SUWR in Chapter 4 as a promising solution to **RQ3**, the first local feature selection model that is guaranteed to have no leakage, and provides step-wise highlights to enhance the interpretability of model decisions.

1.6 Thesis Origins

The three major chapters of this thesis have origins in three individual papers published during my Ph.D. All papers are collaboration results between me (with major contributions) and my co-authors. Here we list the original papers and the publication venues.

- **Chapter 2** 🗎 *Lijun Lyu and Avishek Anand. 2023. Listwise explanations for ranking models using multiple explainers. In ECIR 2023. Springer, 653–668* [83].
- **Chapter 3** 🗎 *Lijun Lyu, Nirmal Roy, Harrie Oosterhuis, Avishek Anand. 2024. Is Interpretable Machine Learning Effective at Feature Selection for Neural Learning-to-Rank? In ECIR 2024. Springer, 384-402* [85].
- **Chapter 4** 🗎 *Harrie Oosterhuis*, Lijun Lyu*, and Avishek Anand. 2024. Local Feature Selection without Label or Feature Leakage for Interpretable Machine Learning Predictions. * equal contribution. Accepted at ICML 2024..*

2

Interpreting Ranking Models

*In this chapter, we start with **RQ1**–How do we explain ranking-specific decisions from black-box text ranking models? Towards this, we propose a novel approach towards better interpretability of a trained text-based ranking model in a post-hoc manner. Popular approaches for post-hoc interpretability of text ranking models are based on locally approximating the model behavior using a simple ranker. Since rankings have multiple relevance factors and are aggregations of predictions, existing approaches that use a single ranker might not be sufficient to approximate a complex model, resulting in low fidelity. In this chapter, we overcome this problem by considering multiple simple rankers to better approximate the entire ranking list from a black-box ranking model. We pose the problem of local approximation as a Generalized Preference Coverage (GPC) problem that incorporates multiple simple rankers towards the listwise explanation of ranking models. Our method Multiplex uses a linear programming approach to judiciously extract the explanation terms, so that to explain the entire ranking list. We conduct extensive experiments on a variety of ranking models and report fidelity improvements of 37% – 54% over existing competitors. We finally compare explanations in terms of multiple relevance factors and topic aspects to better understand the logic of ranking decisions, showcasing our explainers' practical utility.*

This chapter is based on the following paper:

Lijun Lyu and Avishek Anand. 2023. Listwise explanations for ranking models using multiple explainers. In ECIR 2023. Springer, 653–668 [83].

2.1 Introduction

Recent approaches for ranking text documents have focused heavily on neural models [58, 89, 92]. Neural rankers learn the complex and often non-linear relationships between the query and document that are difficult to encode using closed-form analytical ranking functions like BM25 [4]. However, the superior ranking performance of such models comes at the expense of reduced interpretability, thus increasing the risk of encoding spurious correlations and undesirable biases [116, 133]. In parallel to developing better rankers, there has been an increased focus on interpreting neural ranking models [36, 114–116] that specifically aim at explaining the rationale behind the ranking decisions.

This chapter aims to propose post-hoc approaches to interpret neural text rankers. Post-hoc methods explain *already-trained* models and do not compromise on the accuracy of the learned model, hence making them popular choices for interpreting machine learning models. One prevalent strategy in post-hoc interpretability is to *locally approximate* a trained model with a *simple and interpretable proxy or a surrogate model.* The degree of approximation is called *fidelity* and the objective is to maximize the fidelity between the proxy model and the underlying black-box model. Post-hoc methods for rankings entail using simple rankers to locally approximate (on a per-query basis) complex rankers such that the simple ranker has a high rank correlation (or high fidelity) with the complex ranking. Adapting this general post-hoc framework to ranking models has two specific challenges – *how do we aggregate multiple decisions inherent in a single ranking?* And *how do we explain ranking decisions with different inherent relevance factors?*

- **Rankings as aggregations of decisions.** Text ranking models output a ranked list of documents for a given query. Unlike other learning tasks (e.g. regression and classification) that deal with a single decision, the ranking task can be viewed as an *aggregation of multiple pointwise or pairwise decisions* [2]. Any interpretability approach or explainer should therefore explain the reasoning behind the ranking list, or multiple preference pair predictions. Therefore existing explanation techniques such as feature-attribution methods [112, 113, 120] that explain a single decision (pointwise) cannot be seamlessly used for rankings. Instead, a *listwise explanation* method that intends to cover all individual decisions in the entire ranking list is needed for rankings.

- **Different explanations for different relevance factors.** Secondly, it is well-known that when ranking text, multiple relevance factors (also called ranking heuristics or axioms) determine the relevance of a document to a query, e.g., *lexical matching*, *semantic similarity*, *term proximity* etc. Unlike traditional models that explicitly encode each of these relevance factors, neural rankers automatically learn them from data. The next challenge in explaining rankings is ascertaining the relevance factor that best explains a given decision. Informally, there might not exist a single relevance factor that explains or satisfies all preferences $d_i \succ d_j$ in a given ranking. Therefore trying to approximate a ranking with a single relevance factor might result in low fidelity. A notable example is the listwise explanation approach [116] that considers covering multiple ranking decisions, but uses a single explainer which captures only one relevance factor (i.e., term matching), resulting in low-fidelity explanations due to the mismatch of exact terms.

Explainers	Explanation Terms
Term Matching	charlotte, north, sales, 2008
Position Aware	basketball, north, states, learn
Semantic Similarity	felidae, carnivorous, boko, extinction, deserts, iucn
MULTIPLEX	felidae, carnivorous, boko, extinction, deserts, gvwr, north

Figure 2.1: Explaining the query bobcat with multiple relevance factors – (i) "charlotte-bobcat basketball club"; (ii) "learn to hunt bobcat"; (iii) "animal bobcat" and (iv) "bobcat mechanical retailer". MULTIPLEX carefully chooses from multiple relevance factors to explain a ranking. See Figure 2.6 for more examples.

Due to the aforementioned challenges, we then ask **RQ1**: How do we explain ranking-specific decisions from black-box text ranking models? To address this question in this chapter, we define an explanation to be a combination of the underlying *relevance factors* along with the actual *machine intent*. We **firstly** consider multiple simple rankers or explainers (formally defined in Section 2.3.1), which rely on different *well-known* and *human understandable* (to system designers, or IR practitioners) relevance heuristics. **Secondly**, we explain the *machine intent* in terms of *expansion terms* (in addition to the query terms) such that the simple ranker explains a complex black-box model by inducing a similar ranking list. Thus a combination of *simple rankers* that represents a relevance factor, along with its *expanded query terms* (also called explanation terms) is the listwise explanation of the reasoning behind the ranking.

Approach wise, we carefully select a small set of explanation terms sourced from the documents of the ranked list to maximize the explanation's approximation ability (i.e. fidelity). Specifically, we define the GENERALIZED PREFERENCE COVERAGE (GPC) framework, on which we optimize the preference coverage using approximated integer linear programming. Our method MULTIPLEX is shown to be able to improve the fidelity, and more interestingly combine terms from multiple explainers, implicitly covering multiple topics for an ambiguous query. Figure 2.1 shows an example of explanation terms extracted by each single explainer and MULTIPLEX can cover terms of multiple aspects. Note the aspects of terms are specified by manual observation.

We conduct extensive experiments using datasets from the TREC test collections – TREC-DL and Clueweb09 with three neural rankers to evaluate MULTIPLEX. We report fidelity improvements of 37% – 54% over existing competitors. We also present anecdotal examples that showcase the practical utility of MULTIPLEX in understanding neural rankers. The datasets and source code are publicly available [1].

2.2 Related Work

2.2.1 Feature attribution for ranking models

The earliest works of interpreting ranking models were simple extensions to existing pointwise explanation techniques – explain a single instance given a trained ML model for general machine learning tasks in vision and language. Singh and Anand [115], Verma and Ganguly [127] adapted the popular surrogate-based LIME [106] to generate terms as the

[1] https://github.com/GarfieldLyu/RankingExplanation

explanation for a trained black-box ranker. On the other hand, Fernando et al. [36] applied a game-theory feature attribution method [82] to interpret the relevance score of a document given a query. Alternatively, other prevalent gradient-based feature attribution methods [112, 113, 120] can be adapted in the same way to attribute the relevance prediction to the textual input elements. All these methods provide pointwise explanations (why is doc_i relevant?) or pairwise explanations (why is doc_i ranked higher than doc_j?). We instead focus on listwise explanations or explaining the entire ranked list.

2.2.2 Listwise explanations for ranking models

There is limited work on listwise explanations, i.e., explaining the entire ranking list. LiEGe [142] tackles the task as text generation. Specifically, LiEGe employs a Transformer style model to generate terms for each document in a ranked list, and the explanation contains all generated terms. However, this method presupposes documents with labeled explanation terms, which is unrealistic in most application scenarios. Additionally, the explanation generator is not human-understandable, hindering understanding of the explanation generation process. In contrast, GreedyLM [116] uses a simple ranker to replicate the ranking list of a complex black-box model by expanding the query terms. The *simple ranker* and *expanded query terms* constitute the explanation for the complex model. We follow the same philosophy that the *explanation terms* along with the explanation generation process should be human interpretable. However, a limitation of [116] is that it assumes that a single relevance factor (modeled by a simple surrogate ranker) is adequate to explain an entire ranking. We challenge this assumption in this work and use multiple simple explainers instead.

2.2.3 Axioms as explanations

Another line of work uses IR axioms (or ranking heuristics) to ground the decisions of complex models. Axioms are well-understood, interpretable, and deterministic sets of rules that lay down the fundamental relevance factors of documents given a query. Recent works [19, 105] diagnosed a group of ad-hoc neural rankers with a set of axioms and found out that neural models only to a limited extent adhere to the IR axioms. Similarly, Völske et al. [129] also found it hard to characterize BERT models in terms of IR axioms. The hypothesis is axiomatic approaches are limited to using just the query terms, resulting in low fidelity. In this work, we consider a much larger vocabulary of explanation terms to optimize the fidelity of our explanations.

In parallel, there are other works dealing with explaining learning-to-rank (LTR) [114, 118], probing contextual ranking models [117, 131], and intrinsic methods for extractive explanations [48, 73, 147]. We point the readers to a recent survey [5] in explainable information retrieval for a more detailed overview. In this work, we operate on text rankers and generate term-based explanations in a post-hoc manner.

2.3 Background and Preliminaries

We start with the notion of a ranker Φ that takes as input a keyword query $\mathcal{Q}$ to output an ordering π over a set of documents $\pi = (d_1 \succ d_2 \succ \ldots \succ d_n)$ based on the relevance of the documents to the query, i.e., $\Phi(\mathcal{Q}) \longrightarrow \pi$. We aim to interpret Φ in a model-agnostic manner,

using simple proxy rankers (called explainers Ψ). Note that the output of a ranker can be viewed as a set of preferences over the documents, or w.l.o.g $\pi = \{(d_i \succ d_j)\}$. Therefore explaining a ranking π is akin to explaining all or most of the preference pair decisions in π. An example of a single decision is whether the preference pair $(d_i \succ d_j)$ is `true`/`false`.

2.3.1 Explainers for Ranking

The explainer Ψ mimicking a black-box ranking model is essentially a simple ranker operating based on human-understandable closed form formulae (i.e. ranking heuristics). A popular example of such interpretable rankers is BM25 [4] model, which ranks documents for a given query by measuring the *term-matching* frequency of query terms in each document. Apart from term matching, there are also other factors or heuristics that might affect the relevance judgment such as the *term position*. Specifically, in news articles, the title and the introductory paragraphs are regarded to be more important. A ranking model should then weigh the term matching that occurred in the earlier paragraphs more than the rest. Additionally, *semantic similarity* is known to be crucial to address the exact mismatch problem. This is particularly true in neural models with embedding vectors as input. However, the semantic meaning of a term is less interpretable as it can vary if the context changes due to different training procedures or datasets. In this regard, we draw the line of choosing the commonly-used context-free embeddings (i.e. GloVe [97]) as human-understandable input representation, instead of other contextualized embeddings (i.e., generated by BERT language model).

This set of simple ranking heuristics can be large given different granularities [19, 105]. In this work we start from **three** explainers to encode the above three ranking heuristics. Note that our framework allows a flexible amount of explainers, and thus more heuristics can be added if necessary. In summary, the explainers rank a document (d) based on its relevance to a query (q) by:

- **Term Matching** or Ψ_{lm}: $\Psi_{lm}(q,d) = \frac{1}{|d|}\sum_{t\in q} \texttt{tf}(t,d)$, where $\texttt{tf}(t,d)$ denotes the term frequency of t in d.

- **Position Aware** or Ψ_{pa}: a position-aware term-matching model [37], $\Psi_{pa}(q,d) = \sum_{t\in d}\frac{1}{|d|}\sum_{p\in d}\texttt{tf}(t,p)^{\frac{1}{p}}$, where p denotes the p_{th} paragraph in d.

- **Semantic Similarity** or Ψ_{emb}: $\Psi_{emb}(q,d) = \frac{1}{|q|\times|d|}\sum_{t\in q, w\in d}\texttt{cosine}(t,w)$, where t and w are represented by the pre-trained GloVe embedding vectors [97].

2.3.2 Explanations to a Ranking Model

The output of an interpretability procedure is an explanation, which should be *simple*, *human-understandable*, and *faithful* to the behavior of Φ. For the ranking task, the explanation can be decomposed into **two parts**: (1) a simple ranker whose decision-making process is fully transparent; (2) the machine intent of Φ in terms of an expanded query. The quality or fidelity (in XAI parlance) of the explanation can be evaluated by comparing

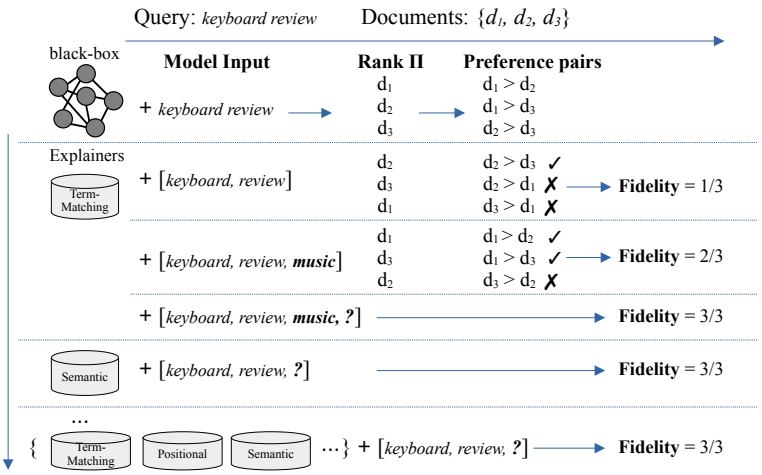


Figure 2.2: Explaining black-box model with simple rankers and query terms.

the ranked lists induced by Φ and Ψ by standard rank-correlation metrics, e.g., Kendall's tau or just counting concordant preference pairs.

Take Figure 2.2 as an example of interpreting the ranking induced by a black-box model. The simple Term Matching explainer with the input terms ("keyboard" and "review") can be regarded as an explanation, with a fidelity of 1/3, as only one out of three preference pairs agrees with the original ranking. It is common that the query term is under-specified, and thus the simple ranker fails to extract the exact query intent. One solution is to use *query expansions* – a commonly used concept in IR (e.g., RM3 [51]) to improve ranking performance. For instance, when adding "music" to the query, the explainer is aware of the musical preference of the black-box ranker and improves the explanation fidelity to 2/3. The questions we ask are: (1) *which terms can be added to the query to maximize fidelity?*, and if more than one explainer is applied, (2) *how can we combine multiple simple explainers to cover as many pairs as possible?*

Fidelity Variants. Note that rankings can be misleading because they do not show the magnitude of the relevance difference. Sometimes the relevance scores of a preference pair can be very close, and explaining such pair is challenging even to humans. Therefore, to avoid uncertainty due to small score differences, we obtain a set of *important* preference pairs after excluding the similar pairs whose prediction difference is below some threshold. As Figure 2.2 shows, suppose the black-box ranker predicts similar scores for d_2 and d_3, then $d_2 \succ d_3$ is not considered for evaluation. As a result, the Term Matching explainer, along with the input terms ("keyboard", "review" and "music"), can faithfully cover all pairs and get 100% fidelity. Given different choices of selecting to-be-explained preference pairs, we introduce different variants of fidelity, which will be further discussed in Section 2.5.3.

2.3.3 Problem Statement

We solve the explaining task as directly optimizing the fidelity, under the constraints of pre-defined explainers and the associated terms. Formally, given a query $\mathcal{Q}$, a complex ranking model Φ and a set of simple ranking models $\{\Psi\}$, we aim to select a small set of terms $\mathbb{E} \in \mathcal{V}$ (where $\mathcal{V}$ is the vocabulary), to explain most of the preference pairs $\{d_i \succ d_j\}$ from the original ranking π.

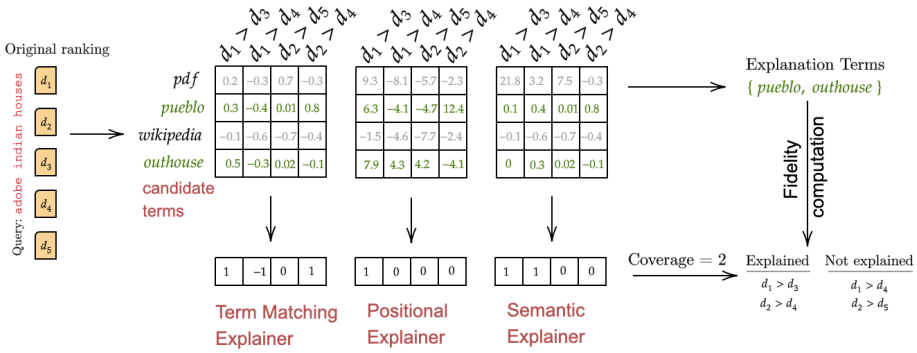


Figure 2.3: Approach overview of MULTIPLEX using multiple explainers.

2.4 Generalized Preference Coverage

As mentioned earlier, choosing explanation terms to maximize fidelity can be formulated as a coverage problem of the preference pairs. We briefly describe the preference coverage (PC) framework as introduced in [116], using a single explainer as a precursor to introducing the generalized PC problem.

2.4.1 The Preference Coverage Framework

Similar to [116], the PC framework operates on a preference matrix constructed with a single Ψ. First, a set of n potentially important candidate terms $\mathcal{X}(\mathcal{X} \subseteq \mathcal{V}, |\mathcal{X}| = n)$ are extracted from the list of documents using simple statistics (e.g., *tf-idf*). Then, m preference pairs are sampled from π to create the preference matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$. Each cell in $\mathbf{M}$ represents the utility or degree of Ψ in explaining the preference $d_{\pi(i)} \succ d_{\pi(j)}$ with t as input, by computing a preference score $f_{ij}^t = \Psi(t, d_{\pi(i)}) - \Psi(t, d_{\pi(j)})$. A positive f score means with t, the Ψ can explain or cover this pair, otherwise cannot. Each t can now be viewed as an m-dimensional vector $\mathbf{f}$, where each element represents how well it explains a specific pair. The PC framework using a single Ψ aims to choose a subset of rows $\mathbb{E} \subseteq \mathcal{X}$ (equivalent to selecting terms) from $\mathbf{M}$ so as to maximize the number of non-zero values in the aggregated vector. Since choosing or not choosing the row/term is a boolean decision,

we can formulate the PC objective as an Integer Linear Program (ILP):

$$maximize \sum_{i=1}^{m} \big(\texttt{sign}(\mathbf{x}^\top \mathbf{M})\big)_i, \quad \text{s.t. } \mathbf{x} = [x_1, \cdots, x_n];\ x_i \in \{0,1\} \tag{PC}$$

$\mathbf{x}$ is a selection vector with boolean values where $x_i = 1$ indicates selecting term $\mathcal{X}_i$, and $x_i = 0$ otherwise. The `sign` is an element-wise operation. Namely, $\mathbb{E} = \{i | x_i == 1\}$. This equation however is NP-hard and not solvable by the prevalent convex programming solvers (e.g., supported by CVXPY [33]) due to the non-differentiable `sign` function. Next, we present an improved formulation of the PC problem followed by a generalization to accommodate multiple explainers called the Generalized Preference Coverage problem.

2.4.2 Optimizing PC for Multiple Explainers

Compared to PC, our proposal should be (i) practically solvable, (ii) ensuring sparse output $\mathbf{x}$ so that the explanation is human-understandable, and (iii) flexible to combine multiple explainers or $\mathbf{M}$.

Correspondingly, the first change we introduce is using `tanh` to approximate the non-convex `sign` operator. Secondly, we add a ℓ_1-regularization $\|\mathbf{x}\|$ to enforce sparsity constraints on the number of terms to be selected. A straightforward way to combine all explainers is to sum up their scores, i.e., $\Psi_{multi}(t,d) = \sum \Psi(t,d)$. However, different explainers can have different output ranges and exhibit high variance. For instance, the term-matching score usually lies in $[0,1]$, whereas the position-aware score typically operates in a much larger range. Normalization these scores in the optimization procedure is central to flexibly adding multiple explainers. We therefore formulate the Generalized Preference Coverage problem that intends to optimize multiple matrices simultaneously as:

$$minimize \left(-\sum_{i=1}^{m} \big(\texttt{tanh}(\mathbf{v})\big)_i + \|\mathbf{x}\| \right) \tag{GPC}$$

$$\text{s.t. } \mathbf{v} = \sum_{j=1}^{p} \texttt{tanh}(\mathbf{x}^\top \mathbf{M}_j), \quad 0 \le x_i \le 1, \quad a \le \sum_{i=1}^{m} x_i \le b$$

Like in PC, GPC also maximizes the number of positive elements in the aggregated vector $\mathbf{v}$, computed by summing up multiple vectors transposed from multiple $\mathbf{M}$. $\mathbf{M}_j$ denotes the matrix constructed by the j^{th} explainer from the total p explainers. Note that `tanh` is also element-wise. The sparsity constraint is ensured by a and b, namely the lower/upper bound of the term-selection budget. The current formulation can now be solved by the latest proposed solver GENO [68] that handles constraints with the *augmented lagrangian algorithm*.

Picking the i^{th} term will choose all i^{th} row vectors simultaneously. Before summing them up, each vector element is already transformed to the same range by `tanh` activation. This accounts for the variable range problem. Figure 2.3 briefly shows the coverage computing when selecting "pueblo" and "outhouse" during optimization.

2.5 Experimental Setup

2.5.1 Datasets and Ranking Models

We choose two datasets: (1) **Clueweb09** collection (category B), for all ranking models, we use 120/40/40 splits for train/dev/test, and the explanation experiments are conducted on the test queries. (2) 40 randomly selected queries from **Trec-DL** 2019 passage ranking test set, and the ranking models are trained on the MS MARCO passage ranking dataset. We focus on the following three neural ranking models:

- **DRMM** [44] computes the term-document similarity histograms beforehand and then jointly learns a matching and a term gate layer from the query and matching histograms. We take the implementation from MatchZoo[2].
- **BERT** [31] takes the query and document separated by [SEP] as input and computes the pooled ([CLS]) representation, on which a feed-forward layer predicts the final relevance score. Both DRMM and BERT models are trained to optimize the margin between the scores of a relevant/non-relevant input pair.
- **DPR** [58] encodes the query and document by two separate BERT models. The relevance is simply measured by the cosine similarity of the two pooled representations. We use the pretrained checkpoints directly without fine-tuning.

2.5.2 Baseline and Competitors

We compare our approach named MULTIPLEX with the following methods:

- **QUERY-TERMS** serves as the baseline by feeding only the query terms to our explainers. By comparing this baseline, we argue that only the original query is insufficient to discover the underlying ranking logic.
- **DEEPLIFT** [112] is a popular white-box feature attribution method. To adapt it to ranking, we first compute the importance of a word in a document using Captum [3], then we take the average across all documents and extract important terms as a listwise explanation for a query. Note that we omit this baseline for DRMM since its input is a histogram, thus the importance cannot be attributed to the word level.
- **GREEDY-LM** [116] uses a term-matching explainer to approximate neural rankers. It optimizes the preference coverage greedily. Our approach shares a similar pipeline of generating candidate terms and preference matrices. By comparing this baseline, we show the improvements of combining multiple explainers and approximated linear programming optimization.

2.5.3 Metrics

Since multiple explainers are applied, a preference pair from the original ranking is counted as explained as long as a single explainer can explain it. This evaluation does not apply to GREEDY-LM as it generates explanation terms based on a single explainer. For both

[2] https://github.com/NTMC-Community/MatchZoo
[3] https://github.com/pytorch/captum

Table 2.1: Fidelity ($\mathcal{F}$) results. The best results are in bold.

		Clueweb09			Trec-DL		
Model	**Method**	$\mathcal{F}_{global}$	$\mathcal{F}_{diff}$	$\mathcal{F}_{sampled}$	$\mathcal{F}_{global}$	$\mathcal{F}_{diff}$	$\mathcal{F}_{sampled}$
Bert	query-terms	0.81	0.88	0.76	0.81	0.82	0.63
	deeplift [112]	0.77	0.81	0.67	0.70	0.75	0.62
	greedy-lm [116]	0.63	0.77	0.69	0.59	0.69	0.84
	Multiplex	**0.88**	**0.97**	**0.93**	**0.86**	**0.93**	**0.97**
DPR	query-terms	0.81	0.86	0.71	0.82	0.84	0.64
	deeplift [112]	0.68	0.71	0.57	0.60	0.63	0.58
	greedy-lm [116]	0.61	0.68	**0.88**	0.63	0.70	0.75
	Multiplex	0.87	**0.93**	0.87	**0.87**	**0.92**	**0.96**
DRMM	query-terms	0.82	0.85	0.72	0.80	0.81	0.59
	deeplift [112]	-	-	-	-	-	-
	greedy-lm [116]	0.57	0.60	0.72	0.53	0.54	0.34
	Multiplex	**0.88**	**0.92**	**0.84**	**0.85**	**0.88**	**0.95**

greedy-lm and Multiplex, we fix 200 candidate terms and 500 sampled pairs for preference matrix construction. We also fix a maximum of 10 explanation terms for all methods except query-terms. For both datasets, we consider a ranking depth (k) of 100.

Similar to [105], we measure fidelity by computing the fraction of the maintained preference pairs by the explainers given the explanation terms. In other words, the fidelity measures the coverage over the *feasible* preference pairs. As mentioned in Section 2.3, depending on the choice of feasible preference pairs, we consider the following three variants of fidelity:

- **Fidelity-global** ($\mathcal{F}_{global}$) includes all $\binom{k}{2}$ pairs induced by a k-length ranking list.
- **Fidelity-sampled** ($\mathcal{F}_{sampled}$) considers the sampled pairs from the matrix construction.
- **Fidelity-diff** ($\mathcal{F}_{diff}$) discards all pairs whose relevance score difference $< g$. The magnitude of g is chosen based on the relevance score distribution of a particular model. For BERT we set $g = 2$ as the prediction margin appears to be larger than the rest two models, for which $g = 0.05$.

2.6 Evaluation Results

To show the effectiveness of our approach, we first present the quality of our approach in terms of fidelity on all datasets and models compared to other competitors in Table 2.1. Then we show the improvements of adding multiple explainers by an ablation study presented in Figure 2.4. Finally, we discuss how our explanations can be used to explain a specific preference pair, as well as other potential use cases.

2.6.1 Effectiveness of Explanations

In terms of fidelity (cf. Table 2.1), our method consistently outperforms other competitors. Besides, for all methods the global fidelity ($\mathcal{F}_{\text{global}}$) scores are always lower than $\mathcal{F}_{\text{diff}}$ where close, hence potentially noisy pairs are all excluded. This shows that all methods and prominently MULTIPLEX can better explain document pairs with larger differences in relevance scores.

Ranking heuristics vs. Query expansion. Though both factors constitute the explanation of ranking, which one is more crucial? Take QUERY-TERMS and GREEDY-LM as a comparison, note that QUERY-TERMS includes the given query terms but three ranking heuristics, while GREEDY-LM on the contrary only relies on one term-matching but richer query information. Their fidelity results show QUERY-TERMS outperforms GREEDY-LM by a large margin, strongly suggesting that ranking heuristics particularly semantic similarity, are more effective in explaining neural models.

The importance of explanation aggregation. Applying simple aggregation strategies (i.e. *average*) on the prevalent pointwise feature attribution methods is shown to be less effective by the results of DEEPLIFT. Compared to QUERY-TERMS, the extra expanded query terms extracted by DEEPLIFT seem unhelpful in enhancing fidelity but introducing noise. On the other hand, methods directly optimizing fidelity (i.e. GREEDY-LM and MULTIPLEX) explicitly include the aggregation in the optimization loop. The $\mathcal{F}_{\text{sampled}}$ results of DEEPLIFT and GREEDY-LM further confirm the importance of aggregation.

The Benefits of our optimization solution. We also experimented with every single Ψ to extract explanation terms with our approximated ILP objective shown in Figure 2.4. Comparing the fidelity results of term-matching (orange bar) with the $\mathcal{F}_{\text{diff}}$ of GREEDY-LM (using the same explainer) in Table 2.1, we show the superiority of our optimizing strategy over the greedy-algorithm.

The Benefits of Combining Explainers. As Figure 2.4 indicates, semantic explainer overall generates the most faithful explanations than the rest. However, combining all explainers can further improve the preference coverage and in turn increase the fidelity results. When one explainer fails to explain a pair, it is still possible to be covered by other explainers. Moreover, we also notice that combining multiple explainers in optimization can generate explanation terms exhibiting multiple topic aspects, especially for short and ambiguous queries. More examples are presented in Figure 2.1 and Figure 2.6.

2.6.2 Utility of Explanations

Explaining document preference. We now show how to explain a single preference pair using MULTIPLEX, i.e., why does a model prefer d_i over d_j? We start by constructing preference scores for each candidate term as described in Section 2.4.1. Next, we select the important terms with significant scores. Figure 2.5 illustrates the explanation terms of two opposing decisions by BERT and DPR respectively, for `keyboard review`.

Discovering model preference and spurious correlations. We believe that explanation terms encode relevance factors that rank relevant documents over others. Based on this assumption, we create a perturbed document by adding explanation terms to a potentially *non-relevant document* (e.g. at the lowest rank). We then feed this modified

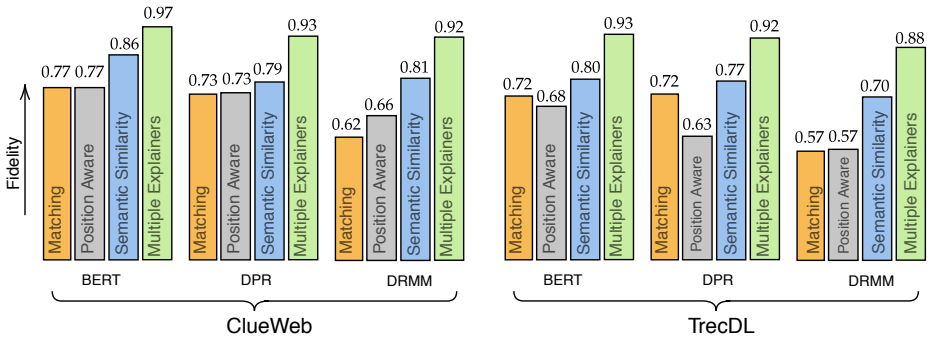
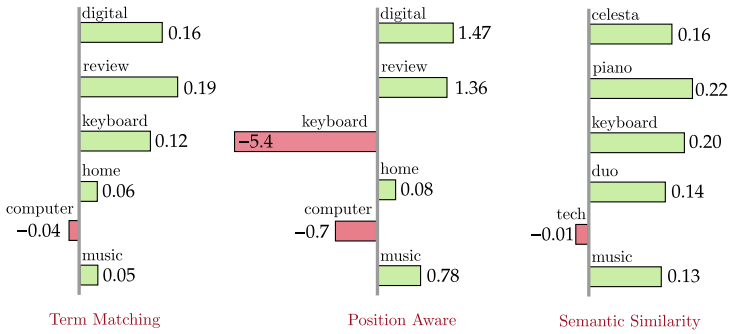


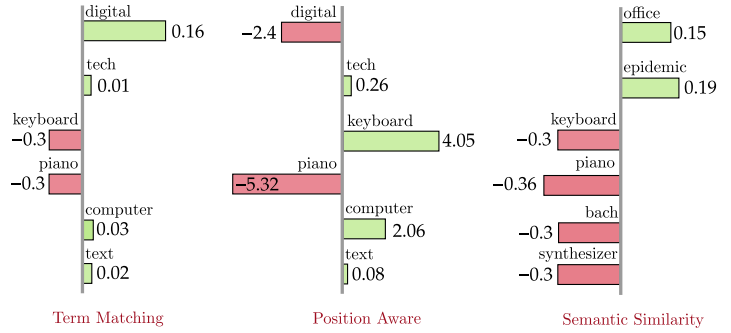
Figure 2.4: *Fidelity-diff* results of each single and combined explainer using our method.

document to the black-box model and measure the rank improvement. Unsurprisingly, the terms extracted by MULTIPLEX result in the maximum rank increase (cf. Figure 2.7), meaning our method can better identify the black-box model's preference. Moreover, we manually selected some ambiguous queries, and our initial observation of their explanation terms suggests the ranking model shows some topic preference when ranking the documents, while the explanation terms representing the preferred topics are also shown dominant quantitively. Thus, it helps understand the model's topic preference more easily by analyzing the explanations instead of going through hundreds of documents.

Another possible usage is model debugging, or finding spurious correlations in models or datasets, by analyzing explanation terms. One simple example is "Wikipedia" which appears as an explanation term for many different queries. This is not surprising as the Wikipedia entity pages are usually labeled as relevant. We leave a more systematic exploration of making use of ranking explanations to future work.



BERT: music-related terms are *pos*



DPR: music-related terms are *neg*.

Figure 2.5: Query: keyboard review. Document pair: clueweb09-en0008-49-09140 (musical keyboard) *vs.* clueweb09-en0010-56-37788 (technical keyboard). BERT prefers the former whereas DPR prefers the latter, resulting in opposite explanations.

Query	Explainer	Explanation	$\mathcal{F}_{\text{diff}}$
adobe indian houses	Term Matching	pdf, adobe, style, house, first, also	0.85
	Position Aware	pdf, adobe, style, texas, wikipedia, 2009	0.81
	Semantic Similarity	pueblo, amarillo, castroville, outhouse, abourezk, alcove	**0.95**
	Multiplex	pueblo, amarillo, castroville, outhouse, abourezk, pdf	0.91
espn sports	Term Matching	espn, abc, network, company, award, entertainment,	0.86
	Position Aware	espn, sportscenter, abc, company, news, espn.com	**0.99**
	Semantic Similarity	espn, sportscenter, abc, walt, disney, entertainment,	0.93
	Multiplex	espn, sportscenter, abc, walt, disney, news, espn.com	**0.99**
hp mini 2140	Text Matching	hp, mini, 2140, 2133	**0.94**
	Position Aware	hp, mini, 2140, 2133	0.90
	Semantic Similarity	hp, touchpad, overview, hdd,	0.71
	Multiplex	hp, mini, 2140, 2133, touchpad, overview	0.91

Figure 2.6: Anecdotal examples show that each explainer selects terms from a different aspect. The color highlights denote the explanation terms in *Multiple* are combined from different explainers. For ambiguous query "adobe Indian houses", *Term Matching* and *Position Aware* focus on popular but 'shallow' terms indicating "adobe company". For certain query "hp mini 2140", the *semantic similarity* suffers from OOV. *Position Aware* can capture the non-frequent yet important terms based on their position, e.g., the official site for the query "ESPN sports".

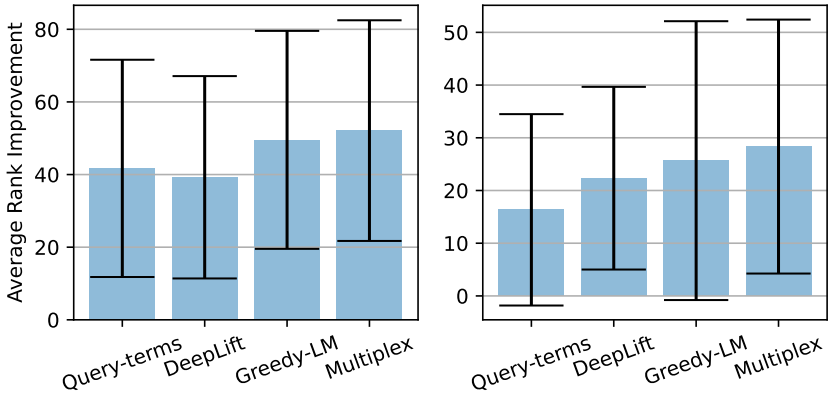


Figure 2.7: Average rank improvements. Left: on all test queries; Right: on hand-picked ambiguous queries. Note that for each query the document size ≤ 100.

2.7 Conclusion and Outlook

RQ1: How do we explain ranking-specific decisions from black-box text ranking models? To conclude, this chapter answers **RQ1** by proposing a post-hoc model-agnostic framework to explain text ranking models using multiple explainers. Our method MULTIPLEX systematically combines multiple explainers to capture different relevance factors encoded in the ranking decisions. The extensive experiments show that our method can generate high-fidelity explanations for over-parameterized models like BERT, delivering up to 54% fidelity improvements. Our method explains a ranking by a set of terms attributed to a union of multiple explainers. It is interesting to examine which explainer (or ranking heuristic) contributes to which extent using which particular terms for future work. We also plan to extend our framework to account for n-grams and to make our explanation generation procedure efficient enough to be used during query processing. Moreover, it is well known that validating explanations is challenging, especially in the absence of ground-truth data. We measure fidelity in this work, however, the fidelity might not reflect the real underline logic of a complex model. Therefore, incorporating human perspectives into the evaluation and meanwhile, balancing the cost of annotating numerous decisions in a ranking are also worth exploring in future work.

3

Interpretable Ranking Models – by Design

*In this chapter, we continue with **RQ2** to explore if interpretable ML is applicable for building self-explainable ranking models. Neural ranking models have become increasingly popular for real-world search and recommendation systems in recent years. Unlike their tree-based counterparts, neural models are much less interpretable. That is, it is very difficult to understand their inner workings and answer questions like how do they make their ranking decisions? or what document features do they find important? This is particularly disadvantageous since interpretability is highly important for real-world systems. In this chapter, we explore feature selection for neural learning-to-rank (LTR). In particular, we investigate six widely-used methods from the field of interpretable machine learning (ML) and introduce our own modification, to select the input features that are most important to the ranking behavior. To understand whether these methods are useful for practitioners, we further study whether they contribute to efficiency enhancement. Our experimental results reveal a large feature redundancy in several LTR benchmarks: the local selection method TabNet can achieve optimal ranking performance with less than 10 features; the global methods, particularly our G-L2x, require slightly more selected features, but exhibit higher potential in improving efficiency. We hope that our analysis of these feature selection methods will bring the fields of interpretable ML and LTR closer together.*

This chapter is based on the following paper:

Lijun Lyu, Nirmal Roy, Harrie Oosterhuis, Avishek Anand. 2024. Is Interpretable Machine Learning Effective at Feature Selection for Neural Learning-to-Rank? In ECIR 2024. Springer, 384-402 [85].

3.1 Introduction

Learning-to-rank (LTR) is at the core of many information retrieval (IR) and recommendation tasks [79]. The defining characteristic of LTR, and what differentiates it from other machine learning (ML) areas, is that LTR methods aim to predict the optimal ordering of items. This means that LTR methods are not trying to estimate the exact relevance of an item, but instead predict relative relevance differences, i.e., whether it is more or less relevant than other items. Traditionally, the most widely adopted and prevalent LTR methods were based on Gradient Boosted Decision Trees (GBDT) [18, 61, 134]. However, in recent years, neural LTR methods have become increasingly popular [39, 95, 99]. Recently, [103] have shown that neural models can provide ranking performance that is comparable, and sometimes better, than that of state-of-the-art GBDT LTR models on established LTR benchmark datasets [21, 30, 101]. It thus seems likely that the prevalence of neural LTR models will only continue to grow in the foreseeable future.

Besides the quality of the results that ranking systems return, there is an increasing interest in building trustworthy systems through interpretability, e.g., by understanding which features contribute the most to ranking results. Additionally, the speed at which results are provided is also highly important [6, 8, 10]. Users expect ranking systems to be highly responsive and previous work indicates that even half-second increases in latency can contribute to a negative user experience [10]. A large part of ranking latency stems from the retrieval and computation of input features for the ranking model. Consequently, *feature selection* for ranking systems has been an important topic in the LTR field [42, 43, 94, 100, 104, 119, 139]. These methods reduce the number of features used, thereby helping users understand and greatly reduce latency and infrastructure costs, while maintaining ranking quality as much as possible. In line with the history of the LTR field, existing work on feature selection has predominantly focused on GBDT and support-vector-machine (SVM) ranking models [39, 56], but has overlooked neural ranking models. To the best of our knowledge, only two existing works have looked at feature selection for neural LTR [100, 104]. This scarcity is in stark contrast with the importance of feature selection and the increasing prevalence of neural models in LTR.

Outside of the LTR field, feature selection for neural models has received much more attention, for the sake of efficiency [72, 73], and also to better understand the model behaviours [7, 148]. Those methods mainly come from the *interpretable* ML field [34, 90], where the idea is that the so-called *concrete* feature selection can give insights into what input information a ML model uses to make decisions. This tactic has already been successfully applied to natural language processing [147], computer vision [9], and tabular data [7, 141]. Accordingly, there is a potential for these methods to also enable *embedded feature selection* for neural LTR models, where the selection and prediction are optimized simultaneously. However, the effectiveness of these interpretable ML methods for LTR tasks is currently unexplored, and thus, it remains unclear whether their application can translate into useful insights for LTR practitioners.

The goal of this work is to answer **RQ2**: Is interpretable ML applicable for building self-explainable ranking models? We address this by investigating whether six prevalent feature selection methods – each representing one of the main branches of the interpretable ML field – can be applied effectively to neural LTR. In addition, we also propose a novel method with minor modifications. Our aim is to bridge the gap between the two fields by

translating the important concepts of the interpretable ML field to the LTR setting, and by demonstrating how interpretable ML methods can be adapted for the LTR task. Moreover, our experiments consider whether these methods can bring efficiency into the practical application by reducing irrelevant input features for neural ranking models.

Our results reveal a large feature redundancy in LTR benchmark datasets, but this redundancy can be understood differently for interpretability and for efficiency: For understanding the model, feature selection can vary per document and less than 10 features are required to approximate optimal ranking behavior. In contrast, for practical efficiency purposes, the selection should be static, and then 30% of features are needed. We conclude that – when adapted for the LTR task – not all, but a few interpretable ML methods lead to effective and practical feature selection for neural LTR.

To the best of our knowledge, this is the first work that extensively studies embedded feature selection for neural LTR. We hope our contributions bring more attention to the potential of interpretable ML for IR field. To stimulate future work and enable reproducibility, we have made our implementation publicly available [1] (MIT license).

3.2 Related Work

3.2.1 Learning-to-Rank (LTR)

Traditional LTR algorithms mainly rely on ML models, such as SVMs and decision trees to learn the correlation between numerical input features and human-annotated relevance labels [22, 38, 57, 64, 77, 134, 137, 143]. Neural approaches [16, 17, 20, 108, 122, 135] have also been proposed, but did not show significant improvements over traditional non-neural models. Inspired by the transformer architecture [126], recent works have also adapted self-attention [95, 99, 103] and produced the neural LTR methods that outperform LambdaMART [134], albeit with a relatively small difference. It shows that neural rankers can provide competitive performance, consequently, the interest and effort towards neural models for LTR are expected to increase considerably in the near future.

Efficiency is crucial in real-world systems since users expect them to be highly responsive [6, 8, 10]. Aside from model execution, the latency of ranking system is largely due to feature construction, as it happens *on-the-fly* for incoming queries. Thus, efficiency is often reached by reducing (expensive) features. Previous works [39, 132] apply a cascading setup to reduce the usage of expensive features. Another growing trend in LTR is to design *interpretable models*. Existing methods rely on specific architecture design, such as general additive model (GAM) [150] or a feature-interaction constrained and depth-reduced tree model [81].

3.2.2 Feature Selection for LTR

Feature selection can achieve both efficiency and interpretability [7, 72, 73, 148]. By selecting a subset of input features, the input complexity of models is reduced while maintaining competitive performance. This helps with (1) efficiency as it avoids unnecessary construction of features [72, 73], and (2) interpretability as fewer input features are involved in prediction [7, 148].

[1] https://github.com/GarfieldLyu/NeuralFeatureSelectionLTR

Existing feature selection methods in LTR are classified commonly as *filter*, *wrapper* and *embedded* methods [42, 43, 100]. Filter and wrapper methods are applied to given static ranking models which are not updated in any way; filter methods are model-agnostic [42] while wrapper methods are designed for a particular type of model [43]. In this work we will focus on the third category, embedded methods, where feature selection is performed simultaneously with model optimization. Most embedded methods are limited to particular model designs such as SVMs [64, 65, 67] or decision trees [81, 94, 139]. To the best of our knowledge, only two methods are designed for neural LTR [100, 104]: one applies group regularization methods [104] to reduce both input and other model parameters; the other [100] uses the gradients of a static ranking model to infer feature importance, and thus it belongs to the *filter* category. We do not investigate these two methods further, as the focus of this work is on *embedded* input feature selection methods.

3.2.3 Interpretable Machine Learning

The earliest work in interpretable ML attempted to explain a trained model in *post-hoc* manner, mainly relying on input perturbations [82, 106], gradients [113, 120] and so on [112]. In parallel, more recent works advocated intrinsically interpretable models, that are categorized as *interpretable-by-design* methods [1, 3, 110]. For neural networks, explaining the decision path is challenging due to the large set of parameters. Therefore, the more prevalent choice for intrinsic interpretable neural models is to shift the transparency to the input features. Namely, the final prediction comes from a subset selection of input elements, e.g., words or pixels and the rest irrelevant features are masked out [24, 73, 147]. Importantly, this selection decision can be learned jointly with the predictive accuracy of a model. Thereby, we limit our research focus in intrinsic interpretable ML models.

Due to the discrete nature of selection, many approaches such as L2X [24], *Concrete AutoEncoders* (CAE) [9], *Instance-wise Feature grouping* (IFG) [88] apply Gumbel-Softmax sampling [52] to enable backpropagation through the feature selection process. Alternatively, regularization is also a commonly-used feature selection approach in traditional ML algorithms [64, 123], and is applicable to neural models, i.e., with INVASE [141] or LassoNet [72]. Moreover, TabNet [7] applies both regularization and the sparsemax activation function [87] to realize sparse selection. These approaches have been successfully applied in language, vision and tabular domains, and suggested that the resulting feature selections substantially improved the user understanding of models and datasets [60, 109].

Despite their success in other domains, we find that the above-mentioned feature selection methods for neural models (L2X, CAE, IFG, INVASE, LassoNet and TabNet) have not been studied in the LTR setting. In response, we hope to bridge this gap between the interpretable ML and the LTR field by adapting and applying these methods to neural ranking models.

3.3 Background

3.3.1 Learning-to-Rank (LTR)

The LTR task can be formulated as optimizing a scoring function f that given item features x predicts an item score $f(x) \in \mathbb{R}$, so that ordering items according to their scores corresponds to the optimal ranking [79]. Generally, there are relevance labels y available for

Table 3.1: Properties of feature selection methods from the interpretable ML field as discussed in Section 3.3.

Method	Global	Local	Sampling	Regularization	Fixed-Budget	Composable
L2x [24]		☒	☒		☒	☒
INVASE [141]		☒	☒	☒		☒
CAE [9]	☒		☒		☒	☒
IFG [88]		☒	☒			☒
LASSONET [72]	☒			☒		☒
TABNET [7]		☒		☒		
G-L2x (ours)	☒		☒		☒	☒

each item, often these are labels provided by experts where $y \in \{0, 1, 2, 3, 4\}$ [21, 101, 102]. Given a training set $\mathscr{D}_q = \{(x_i, y_i)\}_{i=1}^{N_q}$ for a single query q, optimization is done by minimizing a LTR loss, for instance, the *softmax cross entropy loss* [15, 20]:

$$\mathscr{L}(f \mid \mathscr{D}_q) = -\frac{1}{|\mathscr{D}_q|} \sum_{(x,y)\in\mathscr{D}_q} \sum_{i=1}^{N_q} y_i \log \sigma(x_i \mid f, \mathscr{D}_q), \tag{3.1}$$

where σ is the softmax activation function:

$$\sigma(x \mid f, \mathscr{D}_q) = \frac{\exp(f(x))}{\sum_{x'\in\mathscr{D}_q} \exp(f(x'))}. \tag{3.2}$$

The resulting f is then commonly evaluated with a ranking metric, for instance, the widely-used normalized discounted cumulative gain metric (NDCG) [53].

3.3.2 Properties of Feature Selection Methods

As discussed before, feature selection is used in the interpretable ML field to better understand which input features ML models use to make their predictions. Furthermore, feature selection is also important to LTR for increasing the efficiency of ranking systems. However, selecting a subset of input features without compromising the model performance is an NP-hard problem, since the number of possible subsets grows exponentially with the number of available features [43, 100]. As a solution, the interpretable ML field has proposed several methods that approach feature selection as an optimization problem. We will now lay out several important properties that can be used to categorize these methods, which will be elaborated in next section.

Global vs. local. Global methods select a single subset of features for the entire dataset, whereas local methods can vary their selection over different items.

Composable vs. non-composable. Non-composable methods are designed for a specific model architecture, and therefore, they can only perform feature selection for those models. Conversely, composable methods are not constrained to specific architectures, and thus, they work for any (differentiable) model.

Fixed-budget vs. budget-agnostic. Fixed-budget methods work with a pre-defined selection budget, i.e., what number of features should be selected in total, or a cost per feature

and a maximum total cost for the selection. Their counterparts are budget-agnostic methods that do not use an explicit budget, consequently, one has to carefully fine-tune their hyper-parameters to achieve a desired performance-sparsity trade-off.

Sampling-based vs. regularization-based. As their names imply, sampling-based methods optimize a sampling procedure to perform the feature selection, whereas regularization-based methods use an added regularization loss to stimulate sparsity in feature selection. While these groups apply very different approaches, whether one is significantly more useful for LTR purposes than the other remains unknown.

3.4 Feature Selection from Interpretable ML for LTR

In this section, we present a brief technical overview of our selection of six interpretable ML methods and their adaption to neural LTR models, and propose our G-L2x method based on a minor modification. While any ranking loss can be chosen, we use a listwise softmax cross entropy (Eq. 3.1) with all methods, for the sake of simplicity. Therefore, the training of each query is conducted after generating the output of all documents associated with the query. Table 3.1 highlights the properties of all methods and Figure 3.1 provides a visual overview to accompany this section.

3.4.1 Sampling-based Feature Selection

Sampling-based approaches use a two-stage architecture consisting of a *selector* that generates a sparse selection over the input features; and a *ranker* that only takes selected features as its input, in the form of a masked vector $\hat{x}$.

The training of a ranker follows conventional LTR, i.e., Eq 3.1 with x_i replaced by $\hat{x_i}$. But the optimization of a selector (ζ) is not as straightforward; Usually, ζ constructs a probability distribution $\mathbf{p} = [p_1, p_2, \cdots p_d]$, indicating a selection probability per feature. However, the ranker uses a concrete selection $m \in \{0,1\}^d$ from the probability distribution, and this concrete operation does not allow optimization of the selector via backpropagation. The common solution is to generate a differentiable approximation $\tilde{m}$, by *concrete relaxation* or the Gumbel-Softmax trick [52]. Namely, the selection of p_i can be approximated with the differentiable c_i as:

$$c_i = \frac{exp\{(\log p_i + g_i)/\tau\}}{\sum_{j=1}^{d} exp\{(\log p_j + g_i)/\tau\}}, \tag{3.3}$$

where g is the Gumbel noise and $\tau \in \mathbb{R}^{>0}$ is the temperature parameter. Now, the selector ζ can be optimized with stochastic gradient descent by using $\tilde{m}$. The following four sampling-based methods apply this overall procedure, but differ in how they generate $\mathbf{p}$ and $\tilde{m}$.

Learning to explain (L2x). L2x [24] is a local selection method since its neural selector generates a probability distribution $\mathbf{p}$ for each individual input instance. To generate $\tilde{m}$, L2x repeats the sampling procedure k times (Eq. 3.3), and subsequently, uses the maximum c_i out of the k repeats for the i_{th} element in $\tilde{m}$. The intention behind this maximization step is to make the top-k important features more likely to have high probability scores (ideally close to 1).

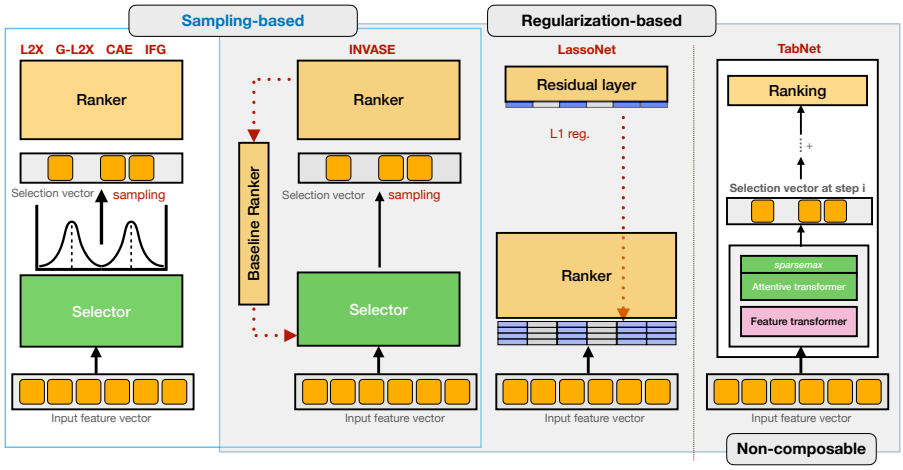


Figure 3.1: Methods overview, as described in Section 3.4.

Global learning to explain (G-L2x, ours). As a counterpart, we propose a global method G-L2xbased on L2x. Our change is straightforward, where L2x generates a different distribution $\mathbf{p}$ for each item, we apply the same $\mathbf{p}$ to all items. In other words, G-L2x includes a global selector layer ζ ($\zeta \in \mathbb{R}^d$) to simulate $\mathbf{p}$, and sampling is conducted in the same way as L2x on the selector weights. Thereby, G-L2x will select the same features for all items in the dataset.

Concrete autoencoder (CAE). CAE [9] is a global method where the selector is the encoder part of an auto-encoder model [63]. Specifically, the selector compresses the input into a smaller representation $\hat{x}$, by linearly combining selected features, i.e. $x^\top \tilde{m}$, where $\tilde{m} \in \mathbb{R}^{k \times d}$ can be viewed as approximated k-hot concrete selection, sampled from the selector weights ($\zeta \in \mathbb{R}^{k \times d}$). Therefore, CAE might result in repetitive selection, and the input dimension to the predictor is reduced to k.

Instance-wise Feature Grouping (IFG). IFG [88] applies a similar approach as L2x, but clusters features into groups and then selects k feature groups for prediction. IFG first assigns a group for each feature via Gumbel-sampling, and then makes a feature selection by Gumbel-sampling k out of the resulting groups. This grouping decision is also guided by how rich the selected features are to recover the original input. Therefore, apart from the ranking objective, IFG jointly optimizes an additional input restoring objective as well (similar to auto-encoders [63]). IFG is agnostic to the number of selected features and the group sizes, it can produce oversized groups and very large selections.

3.4.2 Regularization-based Feature Selection

Instead of the budget-explicit feature selection, regularization-based methods induce sparsity through implicit constraints enforced by regularization terms in the training objective.

We propose modifications to three existing methods to make them applicable to the LTR setting.

Invase. We consider Invase [141] to be a hybrid approach involving both sampling and regularization. Built on the same structure as L2x, the selector of Invase generates a *boolean/hard mask* m (instead of the approximation $\tilde{m}$) via Bernoulli sampling to train the predictor. Since this disables backpropagation, Invase uses a customized loss function that does not need the gradients from the predictor to train the selector. The idea is to apply another individual baseline predictor model that takes the full-feature input, simultaneously with the predictor that takes the masked input. The loss difference between the two predictors is used as a scale to train the selector. Meanwhile, the L1 regularization is applied to the selector output to enforce selection sparsity. Ultimately, Invase will push the selector to output a small set of selections which leads to the most similar predictions as using all features.

LassoNet. As the name suggests, LassoNet [72] adapts a traditional Lasso (L1 regularization) [123] on the first layer of a neural model to eliminate unnecessary features. The challenge with neural models is, all weights corresponding to a particular feature entry in the layer have to be zero in order to mask out the feature. Towards this, LassoNet adds a residual layer with one weight per input feature to the original model to perform as the traditional Lasso. Then, after every optimization step, LassoNet develops a proximal optimization algorithm to adjust the weights of the first layer, so that all absolute elements of each row are smaller than the respective weight of residual layer corresponding to a specific feature. Thereby, LassoNet performs global selection and the sparsity scale is adjusted by the L1 regularization on the residual layer weights.

TabNet. Unlike the previous methods, TabNet [7] is non-composable and tied to a specific tree-style neural architecture. It imitates a step-wise selection process before it outputs the final prediction based only on the selected features. Each step has the same neural component/block but with its own parameters, thus the model complexity and selection budget grow linearly with the number of steps. At each step, the full input is transformed by a feature transformer block first, and then an attentive transformer block conducts feature selection by sparsemax activation [87], as the weights in the resulting distribution corresponding to unselected features are zeros. The final prediction is aggregated from all steps to simulate ensemble models. A final mask m is a union of selections from all steps, and the entropy of the selection probabilities is used as the sparsity regularization.

3.5 Experimental Setup

Since feature selection can be applied in various manners and situations, we structure our experiments around three scenarios:

- *Scenario 1: Simultaneously train and select.* Both the ranking model and the feature selection are learned once and jointly. The methods are evaluated by the performance-sparsity trade-off. It is the standard setup for evaluating embedded feature selection methods in the interpretable ML field [88, 104].
- *Scenario 2: Train then select with an enforced budget.* Practitioners generally set hard limits to the computational costs a system may incur and the efficiency of the system

	#features	#queries			#docs		
		training	validation	test	training	validation	test
MQ2008	46	471	157	156	9630	2707	2874
Web30k	136	18919	6306	6306	2270k	747k	753k
Yahoo	699	19944	2994	6983	473k	71k	165k

Table 3.2: The statistics of three benchmark datasets. The MQ2008 [102] and Web30k [101] are from Miscrosoft LETOR and Yahoo is from YAHOO LTR challenge Set1 [21]. We chose Fold1 split for training, validation and test for all datasets.

can be greatly enhanced if it only requires a much smaller amount of features to reach competitive performance. Following the previous scenario, we evaluate the trained model with test instances where only a fixed amount of features (which the method deems important and selects frequently during training) are presented and the rest are masked out. The resulting ranking performance and the costs of computing the required features indicate how practical the method is in efficiency improvements.

- *Scenario 3: Train, select then re-train.* In real-world settings, ranking systems are updated iteratively, where features will be added or discarded and models re-optimized frequently. To simulate this sequential scenario, we use standard LTR to re-train a DNN model on varying budgets of features produced in Scenario 1. Similarly as in scenario 2, this is also a global selection setup and we select features based on their selection frequency for budget-agnostic methods. This scenario enables a fairer comparison with the non-composable TABNET since re-training is done with the same DNN architecture for all methods.

Datasets and preprocessing. We choose three public benchmark datasets: MQ2008 (46 features) [102], Web30k (136 features) [101] and Yahoo (699 features) [21], to cover varying numbers of available features. More statistics are presented in Table 3.2 We apply a $\log_{1p}$ transformation to the features of Web30k, as suggested in [103]. Yahoo contains cost labels for each feature, for Web30k we use cost estimates suggested by previous work [39].[2] All reported results are evaluated on the held-out test set partitions of the datasets.

Models. We use a standard feed-forward neural network with batch normalization, three fully-connected layers and tanh activation as the ranking model, denoted as DNN. According to the findings in [103], this simple model performs closely to the most effective transformer-based models, but requires much less resources to run. The selector models of L2x and INVASE have the same architecture, and as the only exception, TABNET is applied with its own unalterable model (see Section 3.4).

Implementation. Our experimental implementation is done in *PyTorch Lightning* [35]. For TABNET and LASSONET existing implementations were used.[3] We created our own implementations for the rest of the methods.

[2]MQ2008 is omitted from cost analysis since no associated cost information is available.
[3]`https://github.com/dreamquark-ai/tabnet`; `https://github.com/lasso-net/lassonet`

Table 3.3: Results of ranking performance and feature sparsity for methods applied in Scenario 1. For comparison, we also include GBDT [14, 61] and DNN baselines without feature selection as upper bound. #F denotes the number of selected features. Reported results are averaged over 5 random seeds (*std* in parentheses). Bold font indicates the highest performing selection method; the ⋆ and underlines denote scores that are *not* significantly outperformed by GBDT and the bold-score method, respectively ($p > 0.05$, paired t-tests using Bonferonni's correction).

Listwise loss	MQ2008 NDCG@k (%) @1	@10	#F	Web30k NDCG@k (%) @1	@10	#F	Yahoo NDCG@k (%) @1	@10	#F
Without feature selection.									
GBDT	69.3 (2.5)	80.8 (1.7)	46	50.4 (0.1)	52 (0.1)	136	72.2 (0.1)	79.2 (0.1)	699
DNN	66.2⋆ (2)	80.2⋆ (0.6)	46	46.1 (0.6)	47.7 (0.2)	136	69.4 (0.3)	76.9 (0.1)	699
Fixed-budget feature selection using the DNN ranking model.									
CAE	63.0 (1.1)	78.7 (0.5)	4	32.9 (2.9)	36.6 (2.2)	13	59.2 (0.2)	69.5 (0.1)	6
G-L2X	63.8⋆ (1.3)	79.1 (0.4)	4	41.1 (0.9)	44.4 (0.3)	13	65.4 (0.1)	74.0 (0.0)	6
L2X	63.0 (2.1)	78.7 (0.7)	4	34.5 (2.4)	39.7 (1.9)	13	61.9 (1.1)	73.2 (0.3)	6
Budget-agnostic feature selection using the DNN ranking model.									
INVASE	62.5 (2.2)	77.5 (2.1)	5 (2)	15.1 (0.0)	22.1 (0.0)	0	38.7 (0.0)	57.8 (0.0)	0
IFG	**66.4**⋆ (0.9)	**80.4**⋆ (0.5)	20 (2)	32.5 (5.3)	37.5 (5.3)	72 (30)	69.6 (0.2)	77.1 (0.2)	58 (3)
LASSONET	64.7⋆ (2.2)	79.3 (1.2)	6 (3)	39.4 (0.8)	42.1 (0.3)	8 (2)	63.1 (2.3)	72.4 (1.5)	12 (4)
Budget-agnostic feature selection using a method-specific ranking model.									
TABNET	64.7⋆ (2.7)	78.2 (1.2)	7 (3)	**47.0** (0.4)	**49.2** (0.1)	8 (1)	**70.2** (0.4)	**77.7** (0.1)	6 (1)

Table 3.4: Scenario 1: ranking quality and sparsity for MQ2008, using pointwise and pairwise training objectives. Pointwise models trained using regression loss and pairwise models using Lambdarank loss for GBDT. INVASE for pairwise training is not directly applicable and is omitted.

Method	Pointwise loss, NDCG@k @1	@5	@10	#F	Pairwise loss, NDCG@k @1	@5	@10	#F
GBDT	62.2⋆ (2.7)	72.3 (1.3)	78.2⋆ (1.0)	46	69.7 (0.5)	77.9 (0.5)	81.4 (0.5)	46
DNN	66.5 (0.5)	76.5 (0.4)	80.4 (0.3)	46	62.4⋆ (2.2)	71.0 (0.9)	76.1 (1.0)	46
CAE	63.9⋆ (1.6)	72.9 (0.8)	77.6 (0.5)	4	**62.0** (0.3)	**71.5** (0.1)	**76.9** (0.2)	4
G-L2X	61.8⋆ (1.2)	73.0 (0.4)	77.7 (0.4)	4	57.2 (1.6)	66.1 (2.5)	72.5 (2)	4
L2X	62.3⋆ (0.7)	73.4⋆ (1.1)	77.9⋆ (0.7)	4	61.4 (2.2)	71.1 (1.3)	76.6 (0.8)	4
INVASE	61.2⋆ (3.1)	71.2⋆ (3.9)	76.4⋆ (3.3)	9 (3)	-	-	-	-
IFG	**65.4**⋆ (0.9)	**74.5**⋆ (0.4)	**79.7**⋆ (0.4)	10 (5)	61.8 (2.9)	71.7 (1.5)	76.9 (1.3)	10 (4)
LASSONET	64.2 (1.8)	74.0 (1.3)	78.8 (1.0)	7 (11)	59.1⋆ (4.4)	67.6 (3.4)	73.5 (2.6)	2 (1)
TABNET	62.3⋆ (5)	72.1 (3.8)	77.6 (2.7)	6 (3)	59.0⋆ (4.3)	66.8 (3.8)	73.3 (2.9)	16 (5)

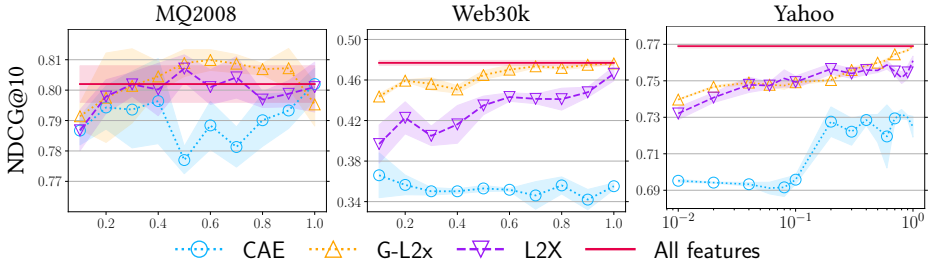


Figure 3.2: Results of three fixed-budget methods applied to scenario 1. The x-axis indicates the pre-specified percentile of selected features (k). The shaded area shows the standard deviation over 5 random seeds.

3.6 Results

We report the findings in the next three subsections, where each corresponds to one of the scenarios described in Section 3.5. Additionally, we also analyze the selection similarity, robustness to feature perturbations and sensitivity to random factors of several methods.

3.6.1 Simultaneous Optimization and Selection

We begin by investigating the effectiveness of the feature selection methods when applied to Scenario 1, where feature selection and model optimization are performed simultaneously. The results for this scenario are displayed in Table 3.3, Table 3.4 and Figure 3.2.

Table 3.3 shows the ranking performance and the respective feature sparsity of all feature selection methods and two baselines without any selection as the upper-bound reference. For fixed-budget methods, the budgets were set to 10% of the total number of features for MQ2008 and Web30k, and 1% for Yahoo (the results with varying budgets are displayed in Figure 3.2). Since the sparsity of budget-agnostic methods is more difficult to control, we performed an extensive grid search and used the hyper-parameters that produced the highest ranking performance with a comparable feature sparsity as the other methods.

The results in Table 3.3 show that not all feature selection methods are equally effective, and their performance can vary greatly over datasets. For instance, on MQ2008 all methods perform closely to the baselines, with only a fraction of the features. However, this is not the case for bigger datasets like Web30k and Yahoo. In particular, Invase selects no features at all due to big uncertainty in selection (for this reason, we omit Invase from all further comparisons). On the other hand, Ifg performs poorly in inducing sparsity, mainly because of its input reconstruction objective, whereas the ranking performance is not substantially better than the rest of methods. Additionally, Cae does not seem effective either, and furthermore, increasing the selected features does not always result in better ranking performance (cf. Figure 3.2). This is most likely because Cae samples with replacement, and thus the same features can be selected repeatedly.

In contrast, the other two sampling-based methods L2x and G-L2x are designed to avoid the repetitive selection issue. Overall, the global selection G-L2x outperforms the local counterpart L2x, possibly because global selection generalizes better to unseen data. Another global method LassoNet is also inferior to G-L2x, mainly due to the difficulties in sparsity weight tuning and manually adjusting weights in the input layer.

Overall, TabNet shows the best performance-sparsity balance across all datasets, and even outperforms the Dnn baseline. Although, the comparison between TabNet and Dnn is not completely fair, as they optimize different neural architectures. It does reveal large feature redundancies in these datasets: TabNet uses <10% of features on Web30k and 1% on Yahoo, yet still beats the Dnn baseline with all features.

Lastly, We also show the adaptability of all methods to both pointwise and pairwise objectives, using MQ2008 as an example in Table 3.4. Note that the hyperparameter setups might affect the performance of both Gbdt and neural models, to save us from heavy hyperparameter tuning, we use the same model parameters as the listwise training. Similarly, the majority of results are very close to their corresponding Dnn upper bounds, with way fewer selected features in comparison to the results on MQ2008 using listwise objective in Table 3.3.

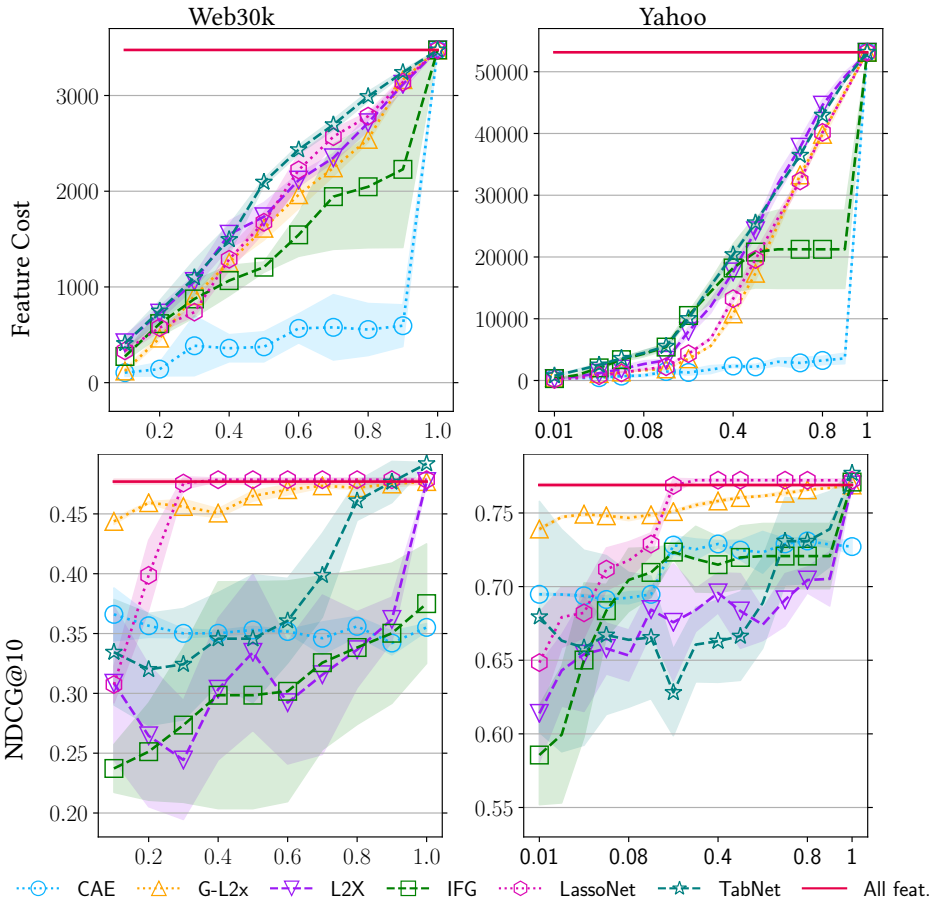


Figure 3.3: Scenario 2. Feature cost (left two) and ranking performance (right two) under incomplete input. The x-axis indicates how many percentages of features are present in the input, to test the trained ranking model. Note this differs from specifying k during training for fixed-budget methods in scenario 1.

To summarize, we find that the local method TabNet is the most effective at balancing ranking performance and sparsity. Slightly inferior but competitive enough is the global method G-L2x, which reached > 95% of baseline performance with only 1% features on Yahoo and > 93% with 10% on Web30k.

3.6.2 Feature Selection for Trained Ranking Models

Next, we evaluate the methods in Scenario 2, where only a specified budget (i.e., a given number of features) of features are present in the test input. Figure 3.3 displays both the ranking performance and the total feature cost for varying degrees of sparsity. The costs represent the time it requires to retrieve the selected feature sets, and allow us to estimate the actual efficiency improvements they provide.

Unlike previous scenario, all local methods including TabNet, are no longer able to

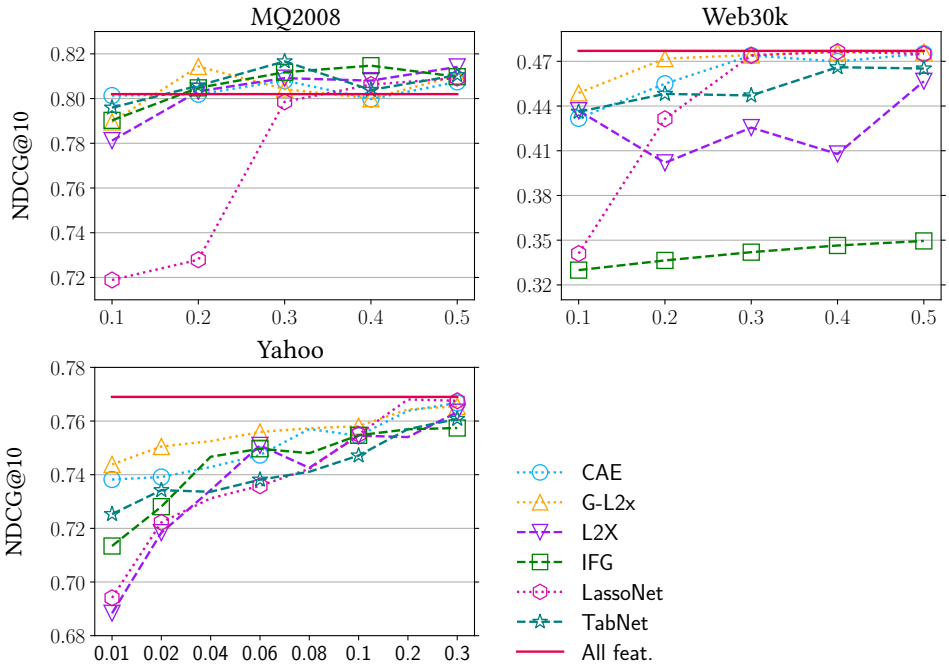


Figure 3.4: Results of Scenario 3: performance of Dnn trained with top% (x-axis) important features identified by each method.

maintain superior performance. This is because for local methods, the selection is made conditioned on full input information, and an incomplete input could affect the selection and thus disrupt its prediction performance.

In contrast, global methods are immune to input changes. Therefore, Cae is still not performing well as it did in Scenario 1; G-L2x and LassoNet provide the best overall performance under small costs. LassoNet maintains baseline performance with less than 40% of features on both datasets, while G-L2x outperforms LassoNet when selected features are less than 30%. Meanwhile, it also shows LassoNet tends to select more costly features than G-L2x.

To conclude, we find that global methods G-L2x and LassoNet perform the best in Scenario 2, where upcoming query inputs are masked under enforced feature budgets. Particularly, G-L2x is superior in both ranking and computing cost when the feature budget is small. This translates to substantial efficiency improvements in practical terms, as ranking performance is maintained by selected features only.

3.6.3 Re-training after Feature Selection

In this section, we address Scenario 3 and evaluate the performance of a Dnn ranking model that is re-trained on the features selected from different methods, as displayed in Figure 3.4.

We see that with only a small number of features (e.g., 30%), the performance of global

methods is as good as using all features, on all three datasets. In particular, most methods outperform the full-feature baseline with only 20% of features on MQ2008. However, it seems clear that G-L2x is the most reliable method in this scenario, as it is the only method that reaches optimal performance with 20% of features on all three datasets.

Overall, local methods have slightly lower performance than global methods. Thus, as expected, it appears that some performance is lost when local feature selections are turned into global selections. However, the difference is limited, suggesting that local methods do tend to select features that are influential in the entire dataset. IFG again performs poorly on Web30k, in line with the results in Table 3.3. Lastly, when comparing Figure 3.4 with the results of Scenario 1 in Figure 3.2, we see that all methods reach higher performance in Scenario 3, indicating that the extra re-training does bring a considerable boost in performance.

In conclusion, all methods except IFG are quite effective across all datasets in Scenario 3, in that, they are able to select 50% of features and reach near-optimal performance. Among them, the global method G-L2x appears to be the best, as it maintains near-optimal ranking performance with as few as 20% of features.

3.6.4 Agreement, Sensitivity and Robustness

Finally, to gain a more thorough understanding of the feature selection methods, we also investigate some of their other characteristics; in particular, selection similarity, method sensitivity and robustness.

Selection Similarity. To understand to what extend different methods agree on what features are most important, we consider the selection similarity between each pair of methods using Kendall's τ correlation, as illustrated by Figure 3.5. It appears that the selections of CAE have the highest correlation with other methods, while IFG has the highest dissimilarity on Web30k. Due to the poor ranking performance of IFG (cf. Table 3.3), we think that IFG mostly disagrees with the other methods because it fails to identify the most-predictive features. Additionally, Figure 3.6 also presents the five most-frequently selected features. Unsurprisingly, the majority of methods deem BM25 relevance-features as most important.

Sensitivity and Robustness. Generally, we wish to avoid fragile or unstable methods that generate drastically different selections, due to random factors or noise in its input data. Accordingly, we conduct two extra experiments to evaluate sensitivity and robustness:

1. A five-fold repetition of simultaneous training and selecting, under identical circumstances, but with different random seeds.

2. We randomly drop features, i.e., replacing their values with 0, and then measure how these perturbations affect the (input-dependent) selection decisions of local methods.

Figure 3.7 illustrates the sensitivity to the random seed of each method, while most methods appear quite stable, TABNET is clearly the most sensitive. Possibly, this is because for TABNET, a different random seed can heavily affect its sparsemax activations and thus change the majority of its selection decisions.

Figure 3.8 depicts the robustness of all local methods to feature perturbations. Again, TabNet is the least robust compared to the other approaches. Ifg appears to be more robust, but this may simply be a result of its large selection size (cf. Table 3.3).

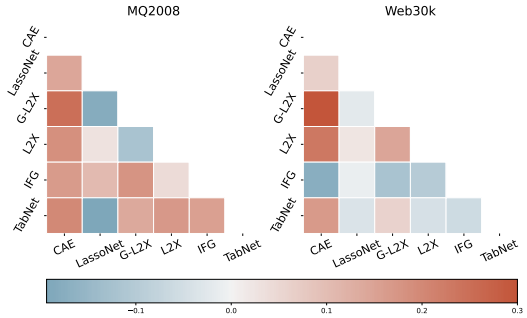


Figure 3.5: Correlation between the selections of different methods according to Kendall's τ on MQ2008 (left) and Web30k (right).

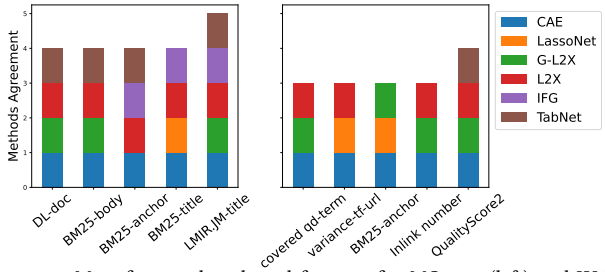


Figure 3.6: Most-frequently selected features for MQ2008 (left) and Web30k (right). BM25-related features are highly favoured in both datasets.

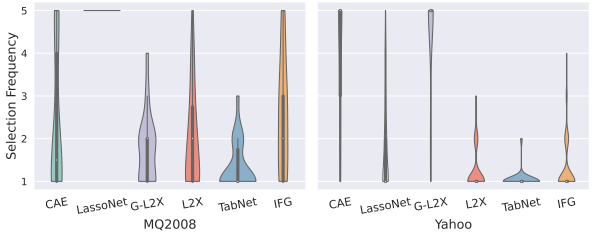


Figure 3.7: Feature Sensitivity, measured by how often a feature is repeatedly selected across 5 models. For each violin plot, the larger the bottom area is, the more diverse selections are, and thus, the more sensitive the method is.

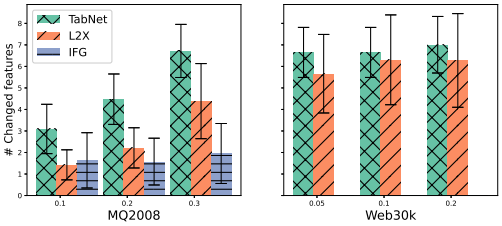


Figure 3.8: Robustness to feature perturbations; the number of selection changes (y-axis) resulting from masking-out features (x-axis indicates ratio). IFG is omitted for Web30k as it changed fewer than one feature on average .

3.7 Discussion

Are interpretable ML methods effective at feature selection for neural LTR? Our experimental results have shown that interpretable ML methods can be very effective at feature selection on neural LTR models, for both interpretability and practical efficiency purposes. Our findings have revealed large redundancies in LTR datasets, as the TabNet method needs <10 features per document yet outperforms a Dnn with access to all features. Moreover, on two large benchmark datasets (Web30k and Yahoo), our results showed that we can reduce more than 60% of feature retrieval costs without substantially affecting ranking performance. Therefore, our study has shown that the feature selection methods from the interpretable ML field can both explain neural LTR models by showing their behavior mainly depends on a few features, and produce large practical efficiency gains by greatly reducing feature costs.

Which feature selection method should be used by practitioners? The answer to this question depends on the purposes for which the method will be used. If one wishes to understand what features of individual documents reveal whether it is relevant or not, then TabNet is the best choice. Since it can select only a handful of features per document, while maintaining a ranking performance that is close the state-of-the-art Gbdt model with full feature information. On the other hand, for efficiency purposes, TabNet is a poor choice because its selections that vary per document, do not translate to good global feature selections. Instead, if one wishes to reduce the feature costs of a system, then the global LassoNet and G-L2x methods have shown to be most effective at inducing feature sparsity without compromising much ranking performance. Specifically, we estimate that their selections lead to an actual reduction in feature retrieval costs of over 60%, while maintaining more than 90% of ranking performance. However, since the computational costs of fine-tuning the hyper-parameters of LassoNet is very high, we recommend G-L2x as feature selection method that is easily applicable and highly effective at improving practical efficiency in neural LTR.

3.8 Conclusion

RQ2: Is interpretable ML applicable for building self-explainable ranking models? To answer this research question, we have investigated all (to the best of our knowledge) prevalent approaches in interpretable ML, and our results have suggested some of the methods are indeed applicable for building self-explainable LTR models.

The main goal of this work is to bring the interpretable ML and the LTR fields closer together. To this end, we have studied whether feature selection methods from the interpretable ML are effective for neural LTR, for both interpretability and efficiency purposes. Inspired by the scarcity of feature selection methods for neural ranking models in previous work, we adapted six existing methods from the interpretable ML for the neural LTR setting, and also proposed our own G-L2x approach. We discussed different properties of these methods and their relevance to the LTR task. Lastly, we performed extensive experiments to evaluate the methods in terms of their trade-offs between ranking performance and sparsity, in addition, their efficiency improvements through feature cost reductions. Our results have shown that several methods from interpretable ML are highly effective at embedded feature selection for neural LTR. In particular, the local method TabNet can

reach the upper bound with less than 10 features; the global methods, in particular G-L2x, can reduce feature retrieval costs by more than 70%, while maintaining 96% of performance relative to a full feature model.

We hope our investigation can bridge the gap between the LTR and interpretable ML fields. The future work can be developing more interpretable and efficient ranking systems, and how that interpretability could support both practitioners and the users of ranking systems. Furthermore, after extensive study of existing explainable models, we started to question if the self-extracted explanations are truly faithful, despite this being a well-established concept. Future work should reconsider the necessary conditions for faithful explanations when designing self-explainable models. To address this, we will conduct a comprehensive investigation in the next chapter.

4

Towards the Faithfulness of Interpretable Models

*Following the previous chapter, we propose **RQ3**: Are self-explainable models faithful, and if not, how to design theoretically guaranteed faithful models? Current self-explainable models mainly rely on input feature selection–either by globally selecting features across the entire dataset, or by locally varying selections per instance. Local feature selection in machine learning provides instance-specific explanations by focusing on the most relevant features for each prediction, enhancing the interpretability of complex models. However, such methods tend to produce misleading explanations by encoding additional information in their selections. In this chapter, we attribute the problem of misleading selections by formalizing the concepts of label and feature leakage. We rigorously derive the necessary and sufficient conditions under which we can guarantee no leakage, and show existing methods do not meet these conditions. Furthermore, we propose the first local feature selection method that is proven to have no leakage called SUWR. We prove that, under certain conditions, our method is the only solution without leakage. Our experimental results indicate that SUWR is less prone to overfitting and combines state-of-the-art predictive performance with high feature-selection sparsity. Our generic and easily extendable formal approach provides a strong theoretical basis for future work on interpretability with reliable explanations.*

This chapter is based on the following paper:

Harrie Oosterhuis, Lijun Lyu* and Avishek Anand. 2024. Local Feature Selection without Label or Feature Leakage for Interpretable Machine Learning Predictions. * equal contribution. In ICML 2024, PMLR 235, 38740-38761* [93].

4.1 Introduction

Feature attributions and feature selections in interpretable machine learning (ML) help users understand how much each input feature influences the output of the model [34, 90]. One prominent family of methods is designed for local feature selection, a.k.a. instance-wise feature selection, for interpretable ML [45]. These approaches aim to only select the most important features per instance and to exclude the rest during inference [76], thereby making the predictions by the model easier to interpret.

Let i refer to an instance in a dataset with $x_i \in \mathbb{R}^d$ as its d-dimensional feature vector representation and y_i as its accompanying label to be predicted. A feature selector ζ takes x_i as input and outputs a feature mask $h_i \in \{0,1\}^d$, either through a stochastic or deterministic process: $h_i \sim \zeta(x_i)$. Let $x_i \odot h_i$ indicate the masked features that are resulted from applying h_i to x_i, where all non-selected features are masked. We denote a masked feature with $\emptyset$, to clearly differentiate it from a zero value, and the jth element in a vector with $[j]$:

$$(x_i \odot h_i)[j] \coloneqq \begin{cases} x_i[j] & \text{if } h_i[j] = 1, \\ \emptyset & \text{if } h_i[j] = 0. \end{cases} \tag{4.1}$$

here ζ is a local selector and produces a different mask for each instance. We note that local feature selection differs from global feature selection which reveals feature importance on the dataset level, as it has a fixed mask for all instances [9, 70, 72, 140]. By being able to vary masks, local methods are more flexible and can give more in-depth insight into the importance of features in individual instances [7, 141].

A widely used setup for local feature selection is to follow a selector-predictor architecture that is typically jointly optimized [7, 54, 141]. More precisely, let f be the predictor model that can take masked features as input: $f(x \odot h)$, importantly, h can be inferred exactly from $x \odot h$. The optimization of the selector ζ and predictor f is usually based on a linear combination of a prediction loss L and the sparsity of a mask $\|h\|$ to enforce high sparsity (and hence interpretability). For a dataset of N instances, we use:

$$\mathscr{L}(\zeta, f) \coloneqq \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{h_i \sim \zeta(x_i)} \big[L(f(x_i \odot h_i), y_i) + \lambda \|h_i\| \big], \tag{4.2}$$

where the parameter $\lambda \in \mathbb{R}_{>0}$ balances feature sparsity against predictive performance. The loss thus incentivizes the exclusion of features that do not contribute to high predictive performance, consequently, the selector should learn only to select the features that are the most important for accurate predictions.

Whilst the reasoning behind the local feature selection approach appears intuitive, previous work has found a *fundamental flaw*: local methods can choose features that provide high predictive performance but clearly are *unfaithful* explanations of feature importance [50]. Jethani et al. [54] discuss a selector and predictor combination for digit classification where a single pixel is selected per image, yet optimal accuracy prediction is maintained. Instead of selecting features based on importance, their selector learned to encode a prediction of the digit in the selection mask h. Because they are optimized jointly, the predictor also learned the relation between the encoding and the original prediction. In other words, instead of selecting the most important features, the behavior of the selector was aimed at passing as much information about the corresponding label as possible.

The resulting selections thus provide misleading explanations that give false insights into the prediction process. As a remedy, Jethani et al. [54] add noise to the selection mask h; whilst this appears to improve the situation, it does not address the underlying problem. Therefore we formulate **RQ3** by asking whether those prevalent feature selection methods are truly faithful. To the best of our knowledge, no existing local feature selection method can guarantee that their selections never provide misleading explanations by encoding additional information.

In this chapter, we provide the first formal approach to the issue of additional information being encoded in local feature selections. We name this problem *leakage* and define it using two novel formal concepts: *label leakage*, where information about the label is encoded in a local selection as defined in Section 4.3.1, and *feature leakage*, where information about the values of non-selected features is encoded in a local selection as defined in Section 4.3.2. Subsequently, we derive the sufficient and necessary properties of a local feature selection method in Section 4.3.3, supported by theoretical proof in Section 4.9, it appears no existing method meets these criteria.

Thus we further extend **RQ3** by asking how we can design faithful models with theoretical guarantees. To address this problem, we propose two methods for optimizing local feature selection policies that are guaranteed to have no leakage. First, we introduce a novel linear programming method in Section 4.4 to search for the optimal selection and prediction policy for any desired sparsity and accuracy tradeoff. This method is highly effective but can only be applied to problems with complete knowledge that are of small scale, which means it has limited practical utility. Second, we introduce a novel method in Section 4.5 that is much more practical and widely applicable called *sequential unmasking without reversion* (SUWR). SUWR selects features over several sequential decision rounds, where each decision is based only on the values of features that were selected in previous rounds and decisions cannot be reversed in subsequent rounds. We prove that in Section 4.10 it is impossible for SUWR to encode information about non-selected features or any labels, since it never had access to those values when deciding what to select. Moreover, in Section 4.11 we conjecture that when the feature distribution fully supports the Cartesian product of possible feature values, SUWR is the only solution without leakage, because it captures all possible policies that have no leakage. Our experimental results indicate that SUWR is less prone to overfitting and combines state-of-the-art predictive performance with high feature-selection sparsity. Furthermore, the sequential decisions of SUWR provide a novel way to explain predictions by giving a narrative of how predictions are formed (e.g., Figure 4.6), a unique insight that previous methods do not provide. The SUWR method can be applied to various forms of data and types of model architectures and optimization, its approach is generic and easily extendable.

4.2 Related Work

The current mainstream lines of work in interpretable machine learning have been categorized into *explaining trained models in post-hoc manner* [82, 106, 112, 113] and *building intrinsically explainable models* [24, 141, 147]. Due to the discrepancies within post-hoc methods [124], the latter has been increasingly advocated in recent years. In the neural era, one common way of building interpretable models is via input features. The main idea is to learn to select a small set of informative input features and use those features

exclusively for the final prediction. Meanwhile, explanations come from the selected features, e.g., pixels for images and words for texts (we note that in language tasks this sort of method is more often referred as *rationale* models [11, 23, 71, 96]). Thus, *sparsity* (i.e., the number of selected features) and the final prediction performance are considered together to measure the model effectiveness and explainability [32].

4.2.1 Feature selection as explanation

One challenge of feature selection is the scarcity of ground-truth labels to indicate the importance of features. As a result, existing solutions learn to select features by jointly optimizing predictive performance and selection sparsity. This type of joint training is referred as *Joint amortized explanation methods* [27, 54, 111]. The learnable selection and prediction function (selector and predictor) can be two separated models, e.g., as for CAE [9], L2X [24], INVASE [141] and REAL-X [54], or components within a single model, e.g., as for TabNet [7]. For the former type, the training signals (e.g., the gradients or rewards) between the predictor and selector are propagated via Gumbel-relaxation [52] or policy gradient. For TabNet, the selection is generated by sparsemax activation [87] and thus trained by standard back-propogation. Additionally, CAE conducts global selection, and the others are local selection methods that vary selections per instance.

4.2.2 Irrationality of local feature selection

Nevertheless, local selection methods, particularly the joint amortized methods have raised increasing concerns in recent works [50, 55, 149]. They argue the selected features do not necessarily align with the true explanations, and thus *unfaithful* to the model behaviors. Furthermore, Jethani et al. [54] showcased the selection mask can leak prediction to the predictor, and therefore achieve unrealistic high performance whether the selected features are relevant or not. As a remedy, they proposed REAL-X, which aims to prevent the predictor overfitting on the selector by injecting noise into the selection masks. Our work shows that REAL-X is still subject to leakage (Section 4.6), and to the best of our knowledge, we have proposed the first theoretically guaranteed solutions to this problem.

4.2.3 Dynamic feature selection

Another tangentially relevant line of work is dynamic feature selection [26, 78]. Similar to SUWR, some dynamic feature selection methods also conduct a greedy selection procedure without access to the full feature set. However, dynamic feature selection is designed for settings where features are costly, and selection should be made to avoid the costs associated with retrieving specific feature values. This is a very different purpose than our work, hence their methods are not designed to address *leakage*, nor do they formally analyse interpretability for ML models.

4.3 Leakage in Feature Selection

This section introduces a formal definition of leakage based on label and feature leakage. Subsequently, we use them to prove the necessary and sufficient conditions for leakage.

To keep our terminology succinct, we define *leakage* as either *feature leakage* or *label*

leakage, thus:[1]

Definition 4.3.1. A feature selector does not have leakage, if it has neither label leakage (Definition 4.3.3) nor feature leakage (Definition 4.3.4).

Table 4.1 displays an intuitive example of leakage where a selection policy mask perfectly encodes all information about the label and non-selected features.

4.3.1 Formalization of label leakage in feature selection

Colloquially, we understand label leakage to be the problem where the selection mask h encodes information about the label. In the context of interpretable machine learning (ML), the purpose of h is to select the features that provide the most salient information. Therefore, this purpose is entirely defeated by the injection of additional information about the label in h. This problematic behavior has been observed in previous work [50, 54], however, to the best of our knowledge, no one has introduced a formal definition of this issue yet.

In our notation, we denote s^{in} as the set of indices of the selected features (included) and s^{ex} for the non-selected features (excluded). To keep our notation brief, we define:

Definition 4.3.2. Ω is the set of all possible selections of feature values and label values:

$$\begin{aligned}\Omega \coloneqq \{(x, y, s^{\text{in}}, s^{\text{ex}}) : p(x) > 0 \wedge p(s^{\text{in}}, s^{\text{ex}} \mid x, \zeta) > 0 \\ \wedge p(x[s^{\text{in}}], y) > 0 \wedge s^{\text{in}} \cup s^{\text{ex}} = \{1, 2, \ldots, d\}\}.\end{aligned} \tag{4.3}$$

Our proposed definition of label leakage is based on the idea that the selection h should not be able to provide information about the label. For a selection, $(x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega$, the predictive information in this selection can be represented by the *natural* label distribution conditioned on the selected feature values: $p(y \mid x[s^{\text{in}}])$. This distribution can be further conditioned the fact that s^{in} has been selected by the selector ζ: $p(y \mid x[s^{\text{in}}], h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0, \zeta)$. The key insight in our definition is that when there is no label leakage, these distributions should be equal.

Definition 4.3.3. A feature selector ζ does not have label leakage, if conditioning the label distribution on the selection of features by ζ does not change the label distribution:

$$\begin{aligned}&\forall (x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega, \\ &\quad p(y \mid x[s^{\text{in}}]) = p(y \mid x[s^{\text{in}}], h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0, \zeta).\end{aligned} \tag{4.4}$$

In other words, if the knowledge that a feature selection comes from a specific selector ζ changes the probability of a label, then ζ has label leakage. Imagine two masked feature values: $x_1 \odot h_1 = x_2 \odot h_2$, one made with a uniform random selection, the other with ζ, if predictions are only based on the selected feature values then both should lead to the exact same predictions: $\forall y, p(y \mid x_1 \odot h_1) = p(y \mid x_2 \odot h_2, \zeta)$.

[1] Our definition is different but related to the concept of *data leakage*: the availability of information during optimization that is unavailable during inference [59].

Table 4.1: Example of feature and label leakage in feature selection (non-selected features are omitted). The label y is the sum of the two independent features, therefore, perfect label prediction should only be possible with both features. However, each $x \odot h$ value is matched with a single label and set of feature values, thereby, this solution provides 100% accuracy in label prediction and feature reconstruction, with a 62.5% feature reduction. This combination of performance and sparsity is only possible because of leakage.

$p(x,y,h)$	$x[1]$	$x[2]$	$h[1]$	$h[2]$	$(x \odot h)[1]$	$(x \odot h)[2]$	y
0.25	1	1	1	0	1		2
0.25	0	1	0	1		1	1
0.25	1	0	0	1		0	1
0.25	0	0	0	0			0

4.3.2 Formalizing feature leakage in feature selection

Analogous to label leakage, we also propose the concept of feature leakage where the selection mask h encodes information about non-selected features. As illustrated in Table 4.1, we motivate the prevention of feature leakage with two arguments:

(i) feature leakage defeats the purpose of feature selection as information about the values of non-selected features is not actually excluded; and

(ii) when there is a correlation between features and labels, a basic assumption in machine learning [12], feature leakage implies label leakage.

Therefore, it also seems infeasible to prevent label leakage without also tackling feature leakage. We formally define feature leakage as:

Definition 4.3.4. A feature selector ζ does not have feature leakage, if conditioning the feature distribution on the selection of features by ζ does not change the feature distribution:

$$\begin{aligned} \forall (x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega, \quad & p(x[s^{\text{ex}}] \mid x[s^{\text{in}}]) \\ & = p(x[s^{\text{ex}}] \mid x[s^{\text{in}}], h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0, \zeta). \end{aligned} \tag{4.5}$$

Similar to label leakage, the intuition behind feature leakage is that knowing that a feature selection was made by ζ should not affect the probability of non-selected feature values.

4.3.3 The necessary and sufficient conditions for leakage

From these formal definitions of feature leakage and label leakage, we derive the sufficient and necessary conditions for a feature selector without leakage in Section 4.9. We find the following:

Corollary 4.3.5. *A feature selector does not have leakage if and only if every probability for every possible feature selection does not depend on any label values or any non-selected*

feature values:

$$\forall(x, y, s^{in}, s^{ex}) \in \Omega, \quad p(h[s^{in}] = 1, h[s^{ex}] = 0 \mid x[s^{in}], \zeta) = p(h[s^{in}] = 1, h[s^{ex}] = 0 \mid x[s^{in}], x[s^{ex}], y, \zeta). \tag{4.6}$$

Proof. Follows directly from Theorem 4.9.3 and Theorem 4.9.4 in Section 4.9. □

In other words, a feature selector has no leakage if the probability of a selection is only determined by the values of the selected features, and not by the label or non-selected feature values. Therefore, for any possible feature values x and any label value y and any selection mask h, any change in the label or in any of the features not selected by h should not result in a different probability for the selection: $\zeta(h \mid x)$. Thus, for any possible feature values x' and label value y', where the selected features have identical values: $x \odot h = x' \odot h$, the probability of the selection should be identical: $\zeta(h \mid x) = \zeta(h \mid x')$.

Intuitively, we can understand that if the value of the label or unselected features changes the behavior of ζ, then it could be possible to infer information about unselected features or the label from the behavior of ζ. Accordingly, we can prove a feature selector has leakage by finding a single example of two pairs of (x, y) and (x', y') for which the above condition does not hold. Conversely, to prove a feature selector has no leakage, we have to rule out the possibility of such an example entirely.

4.4 A Linear Programming Solution

We now propose our first method that meets the above criteria using linear programming [29]. It requires full knowledge of the problem setting, i.e., $p(x, y)$ is known completely, and assumes a finite set of possible values for x. In this setting, the perfect predictor is available, e.g., for a mean squared error loss:

$$f^*(x \odot h) = \mathbb{E}_x[y \mid x \odot h] = \sum_{x' \,:\, x' \odot h = x \odot h} p(x') \sum_y p(y \mid x') y, \tag{4.7}$$

and thus, only ζ has to be optimized. Corollary 4.3.5 shows that the probability of any masked feature vector $x \odot h$ should only depend on the selected features, since:

$$\forall(x, x', h), \big(p(x) > 0 \wedge p(x') > 0 \wedge (x \odot h) = (x' \odot h)\big) \longrightarrow \zeta(h \mid x) = \zeta(h \mid x'). \tag{4.8}$$

Therefore, for optimization, we only have to consider a single probability variable for every possible set of values for $x \odot h$. The probability variables should be chosen to minimize: $\mathscr{L}(\zeta, f^*)$ (Eq. 4.2), under the constraint that they describe valid probability distributions:

$$\forall x,\ p(x) > 0 \longrightarrow \Big(\sum\nolimits_{h \in \zeta(x)} \zeta(h \mid x) = \sum_{s^{\text{in}}, s^{\text{ex}} \,:\, s^{\text{in}} \cup s^{\text{ex}} = \{1,2,\ldots,d\}} p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta) = 1 \Big). \tag{4.9}$$

Section 4.12 details how this task is translated to a linear programming problem. Whilst its requirements limit it to unrealistic toy problems, this method enables us to closely approximate the Pareto optimal front of selection without leakage, which we use in our analysis of existing methods.

Algorithm 1 Inference with the SUWR method.

Input: Features: x, Max-t: T, Selector: ζ, Predictor: f
$h \leftarrow \mathbf{0}$
for $t \in [0, 1, \ldots, T-1]$ **do**
  if Bernoulli_Trial($\zeta^t_{\text{stop}}(x \odot h)$) **then**
    Return: $(f(x \odot h), h)$
  end if
  $h \leftarrow h + \text{Sample\_Mask}(\zeta^t_{\text{select}}(x \odot h))$ *# Eq. 4.10*
end for
Return: $(f(x \odot h), h)$ =0

4.5 Sequential Unmasking without Reversion

In this section, we propose a more practical method titled *sequential unmasking without reversion* (SUWR), which describes a feature selection algorithm that provenly has no leakage, but is applicable to more realistic settings than the linear programming solution. SUWR guarantees no leakage by approaching the selection of features as a sequential decision process where each decision is only based on a specific subset of feature values, and no decision can be reversed at a later step. The core of SUWR is its selection inference algorithm, which is agnostic to what underlying ML model is used and how it is optimized. Therefore, SUWR can be seen as a generic framework that can easily be extended and adapted to specific feature selection problems.

4.5.1 Feature selection inference with SUWR

From Section 4.3, we know that a feature selector ζ without leakage, should base the probability of a specific selection only on the values of the selected features. As discussed in Section 4.4, the probability distribution over each possible selection of feature values has to be valid.[2] Based on these properties, we propose SUWR which meets these criteria through sequential selection. Algorithm 1 describes inference with SUWR in pseudocode, the remainder of this section describes it step-by-step.

SUWR requires a model ζ that can output a stop probability and a distribution to sample feature indices, given an input of masked features. The feature selection process takes place over T steps, each step starts by deciding whether to stop the process, and if not, which features to select next. For a step t, where $0 \leq t < T$, a Bernoulli trial is performed according to $\zeta^t_{\text{stop}}(x \odot h^t)$ and if successful then the process is stopped and h^t is the final feature selection and $f(x \odot h^t)$ the final prediction. Otherwise, the process continues and a new set of feature indices is sampled and added to the selection mask:

$$u^t \sim \zeta^t_{\text{select}}(x \odot h^t), \qquad h^{t+1} = h^t + u^t. \tag{4.10}$$

Importantly, both the stop probability and the sampling of new features are only conditioned on the values of features selected in the previous steps ($x \odot h^t$). Accordingly, the first

[2]Meeting both of these criteria is not trivial, since a standard normalization term would depend on all possible selections for an instance x and thus also on non-selected features; i.e., $\zeta(h \mid x) = \hat{\zeta}(h \mid x) / \sum_h \hat{\zeta}(h \mid x)$ is not allowed since the normalizing denominator depends on all feature values.

step ($t = 0$) starts with an empty mask $h^0 = \mathbf{0}$, and the stop probability $\zeta^0_{\text{stop}}(x \odot h^0) = \zeta^0_{\text{stop}}(\emptyset)$ is constant over x, similarly, the feature distribution $\zeta^0_{\text{select}}(x \odot h^0)$ is the same for every x in the first step. Additionally, since each step only adds features to the selection and never removes any, the probability of the decisions that lead to h^t in a step t, only depends on the values of features selected in previous steps ($x \odot h^{t-1}$). If the final step $t = T - 1$ is reached, then the process is automatically stopped ($\zeta^T_{\text{stop}}(\cdot) = 1$) and the final selection is h^T and the final prediction $f(x \odot h^T)$.

As we can see, SUWR is completely agnostic to what the underlying model ζ and predictor f are; it only requires them to handle masked inputs and ζ to output a stop probability and feature distribution. The parameter T acts as a computational budget as it ensures the process halts within T steps. Additionally, T is also a feature budget when ζ limits the number of features to be sampled per step.

Section 4.10 provides a full proof that proves SUWR has no leakage. The intuition behind this property is straightforward: Any decision to select a feature is never based on information from (thus far) unselected features. Therefore, the value of a feature that is not in the final selection could never affect its probability. Furthermore, the process guarantees a selection is always made, thereby providing a valid probability distribution over all possible feature selections.

In addition, in Section 4.11 we conjecture that the SUWR algorithm describes every possible selection policy without leakage, when the feature value distribution provides support for the Cartesian product of possible feature values:

$$\begin{aligned} \forall i, j, a, b,\ \big(p(x[i] = a) > 0 \wedge p(x[j] = b) > 0\big) \\ \longrightarrow p(x[i] = a, x[j] = b) > 0. \end{aligned} \tag{4.11}$$

In other words, we conjecture that when Eq. 4.11 holds, the inference of any feature selection policy without leakage can be computed by the SUWR algorithm. Therefore, in this setting, SUWR captures *all* solutions to feature selection without leakage, and thus, SUWR provides the *only* solution to feature selection without leakage when Eq. 4.11 is true.

4.5.2 Optimization of SUWR feature selection policies

While SUWR inference strictly follows Algorithm 1 to prevent leakage, there are no restrictions on the optimization of the underlying ζ and f models. Therefore, any optimization method can be chosen without risking the introduction of leakage. For this work, we propose a reinforcement learning optimization approach that is evaluated in our experiments.

The set of possible feature selections grows exponentially with the number of features, it is therefore important that we avoid iterating over all possibilities. We use a REINFORCE approach [121] and repeatedly sample a set of T selection steps while ignoring the stop probabilities. Thus, we start at $t = 0$ with the zero selection: $\bar{h}^0_i = \mathbf{0}$, and for each subsequent step t, we follow the SUWR procedure: $\bar{u}^t_i \sim \zeta(x_i \odot \bar{h}^{t-1}_i)$, $\bar{h}^t_i = \bar{h}^{t-1}_i + \bar{u}^t_i$. For each datapoint x_i, this results in a sampled sequence of T selection masks: $\bar{H}_i = \{\bar{h}^0_i, \bar{h}^1_i, \ldots, \bar{h}^T_i\}$. The probability that SUWR stops at any t, conditioned on the sampled sequence is:

$$p_{\text{stop}}(t \mid \bar{H}_i) = \zeta^t_{\text{stop}}(x_i \odot h^t_i) \prod_{j=0}^{t-1} \big(1 - \zeta^j_{\text{stop}}(x^j_i \odot h^j_i)\big). \tag{4.12}$$

Using this formulation, we can create the following unbiased estimate of our generic loss function (Eq. 4.2):

$$\bar{\mathscr{L}}(\zeta, f) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T} p_{\text{stop}}(t \mid \bar{H}_i)\big(L(f(x_i \odot \bar{h}_i^t), y_i) + \lambda\|\bar{h}_i^t\|\big). \tag{4.13}$$

Computing its gradient w.r.t. ζ_{stop} is straightforward; for the gradient w.r.t. ζ_{select}, we use the log-trick from the REINFORCE method [121]. Then, we apply standard gradient descent to optimize both ζ and f based on our sampled loss $\bar{\mathscr{L}}$.

4.5.3 Discussion

Since we can prove SUWR has no leakage, each mask h is guaranteed to indicate the only features that were used to make its corresponding prediction. To the best of our knowledge, SUWR is the first method to have this guarantee, therefore, we argue it is also the first feature selection method that guarantees its explanations are *faithful* [50]. Furthermore, the sequential selection procedure can be interpreted as a step-by-step narrative of how the prediction was constructed. For example, Figure 4.6 displays multiple steps of SUWR on images of a sandal and a boot. At each step, we can see what information became available to the predictor and how this changes its predictions. Thereby, this step-by-step explanation provides even more insight than the final selection mask. We believe SUWR is the first approach that produces narrative explanations about feature importance.

While the guarantee of no leakage is a great advantage over existing methods, the SUWR algorithm could potentially require more computational costs than previous approaches. Namely, for each intermediate feature selection step, a call to ζ_{select} is made. This could pose a challenge to data with high dimensionality, e.g., if ζ only selects a single feature per step, and thus a high T should be chosen. Luckily, the SUWR framework is highly flexible and can be adapted to handle such situations better. For instance, one can choose ζ_{select} to be a lightweight model that can choose multiple features at once. In our experiments in Sections 4.6 & 4.7, we choose ζ_{select} to be a model that selects one feature per step t; in contrast, for the experiment based on image data in Section 4.8, we use a ζ_{select} that selects a patch of nine pixels per step. This makes the resulting selection easier to interpret than one where individual pixels can be selected, while at the same time reducing the number of steps needed to select a complete image. We expect that specific ζ_{select} models can be developed to increase the computational efficiency and scalability of SUWR further.

Nevertheless, we want to note that there are some unintuitive aspects of SUWR that seem to be unavoidable consequences from the definition of leakage. In particular, at the first step ($t = 0$) SUWR selects features without conditioning on any feature values, thus this first step can be seen as a *blind* selection. While $\zeta_{\text{select}}(\emptyset)$ can be optimized to select the most informative features, its distribution over features must be the same for all possible values of x. At first glance this may seem counter-intuitive, however, it appears that this is an inevitable consequence of selecting without leakage. Consider a setting where we wish to select a single feature per x without leakage, according to Corollary 4.3.5, the selection of a single feature can only depend on the value of that single feature. However,

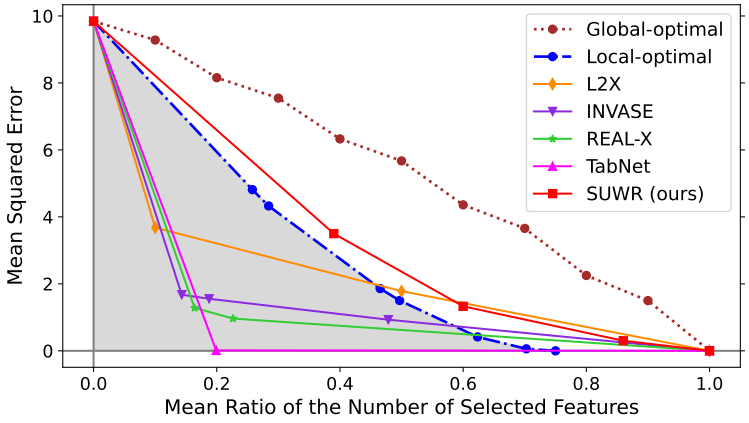


Figure 4.1: Performance curves of the first experiment. Grey area indicates performance that is impossible without leakage.

if the distribution of features supports the Cartesian product of possible feature values (Eq. 4.11), then the probability of each mask is not dependent on any feature values. To put this formally, let $h_{\text{only } i}$ indicate the mask where only feature i is selected: $h_{\text{only } i}[i] = 1, \forall j \neq i, h_{\text{only } i}[j] = 0$, if only such masks can be chosen then the probability each mask is independent of any feature value since:

$$\begin{aligned} &\zeta(h_{\text{only } i} \mid x[i]) \\ &= 1 - \max_{\{x' : p(x')>0 \wedge x[i]=x'[i]\}} \sum_{j:0<j<d \wedge i \neq j} \zeta(h_{\text{only } j} \mid x'[j]) \\ &= 1 - \max_{\{x' : p(x')>0\}} \sum_{j:0<j<d \wedge i \neq j} \zeta(h_{\text{only } j} \mid x'[j]) = \zeta(h_{\text{only } i}), \end{aligned} \tag{4.14}$$

where we rely on the fact that Eq. 4.11 implies that the maximum operator over unselected feature values is a constant w.r.t. the value of any selected feature $x[i]$. Thus, this derivation proves that, in this setting, blind selection is necessary for feature selection without leakage.

4.6 Experiment 1: Pareto Front Analysis

4.6.1 Setup

Our first experiment is designed to identify whether existing methods have leakage. For that, we design an idealized setup where complete information is available so the Pareto front can be approximated. Leakage can then be identified by performance that exceeds that front. Specifically, we design a toy problem with ten binary features: $x \in \{0,1\}^{10}$, in a uniform distribution: $p(x) = 1024^{-1}$. As labels we use a sum of the product of feature pairs: $y = (\sum_{i=1}^{5} x_{2i-1} x_{2i})^2$, this induces feature redundancies that enable interesting local feature selection. For example, if $x_1 = 0$ then the value of x_2 is irrelevant to y, but if $x_1 = 1$ then x_2 is relevant; this is a typical kind of pattern that only local feature selection methods

can capture. For this experiment, all methods are provided all possible values of x and y, this creates a fair comparison with the Pareto front which is constructed using the same complete information.

4.6.2 Methods

The comparison includes the following state-of-the-art methods:

(i) L2X [24]: `https://github.com/Jianbo-Lab/L2X`;

(ii) INVASE [141]: `https://github.com/jsyoon0823/INVASE`;

(iii) TabNet [7]: `https://github.com/dreamquark-ai/tabnet`;

(iv) REAL-X [54]: `https://github.com/rajesh-lab/realx`;

(v) our proposed SUWR method.

(vi) local-optimal, a close approximation of the Pareto front using the linear programming method from Section 4.4;

(vii) global-optimal, the Pareto front of global feature selection computed through brute-force.

All methods are built on neural networks. Among all, L2X, INVASE and REAL-X have independent selector and predictor models. The selector is constructed by feed-forward (FF) layers and outputs a selection probability for each feature. The predictor has a similar architecture but outputs the task-specific prediction, using the selected input by masking out the unselected features. CAE is slightly different, as it uses a single trainable $d \times k$ matrix as the global selector, the matrix values are used as the selection probabilities. The predictor is an FF network, which transforms the selected features (so the input dimension reduces to k) and outputs the prediction. TabNet unlike the others, has a single architecture for both selection and prediction. The selection is conducted step-wisely by a neural selection component and the final prediction is generated by ensembling the outputs from all steps.

Our method is flexible in architecture design. We choose to employ a simple model with FF networks to generate selection (u^t), prediction ($\hat{y}^t$) and stop probability (p^t_{stop}) simultaneously at the step t, defined as follows:

$$enc = \text{FF}_{\text{enc}}(x \odot h^t), \quad p^t_{\text{stop}} = \text{FF}_{\text{stop}}(enc), \quad u^t = \text{FF}_{\text{select}}(enc), \quad \hat{y}^t = \text{FF}_{\text{pred}}(enc) \quad (4.15)$$

FF_{enc} is used to encode the input to a hidden representation and across all experiments, we set it to 3 layers. FF_{stop} and $\text{FF}_{\text{select}}$ are both set to 1 layer. FF_{pred} is set to 1 layer for toy and synthetic datasets, and 2 layers for MNIST datasets. The selection for next step h^{t+1} is sampled from u^t, and to avoid repeated selection, the probabilities of selected features in corresponding u^t are set to 0 before sampling.

For sparsity-related hyper-parameters, both L2X and CAE require a pre-specified k value as the number of selected features; the rest of methods determine the number of selections by a sparsity weight λ. Additionally, TabNet also requires a number of steps n_{steps}, except for λ. For our method, we need to specify a maximum selection budget (or step) T, and a sparsity weight λ to control the number of selections.

Table 4.2: Selection and prediction performance on the synthetic benchmark of the second experiment. Results are averages over five runs.

Dataset	**Syn1** (g_1)			**Syn2** (g_2)			**Syn3** (g_3)			**Syn4** (g_4)				**Syn5** (g_5)				**Syn6** (g_6)			
Metrics	TPR↑	FDR↓	AUROC↑	TPR↑	FDR↓	AUROC↑	TPR↑	FDR↓	AUROC↑	CFSR↑	TPR↑	FDR↓	AUROC↑	CFSR↑	TPR↑	FDR↓	AUROC↑	CFSR↑	TPR↑	FDR↓	AUROC↑
w/o FS	100.	82.	.578	100.	64.	.789	100.	64.	.854	100.	100.	64.	.558	100.	100.	64.	.662	100.	100.	55.	.692
Oracle	100.	0.	.700	100.	0.	.895	100.	0.	.903	100.	100.	0.	.818	100.	100.	0.	.823	100.	100.	0.	.902
L2X	33.2	33.6	.675	44.6	55.4	.872	66.0	34.	.889	56.5	79.2	34.7	.781	51.0	71.9	43.6	.788	34.0	80.1	19.9	.876
INVASE	100.	0.	.692	100.	0.	.873	95.0	0.	.883	56.	91.	10.2	.792	40.7	76.	2.2	.780	60.7	89.4	7.0	.877
TabNet	86.4	57.9	.667	98.7	5.6	.885	96.6	9.7	.903	99.7	91.5	29.5	.789	98.9	92.5	36.2	.791	100.	97.5	23.6	.870
REAL-X	100.	24.2	.661	100.	20.0	.794	100.	7.94	.873	100.	99.9	41.9	.748	100.	99.8	52.4	.774	100.	97.2	8.27	.842
SUWR	100.	2.35	.700	97.0	0.	.895	100.	0.	.903	100.	98.0	20.0	.810	100.	99.6	20.0	.816	100.	97.4	0.37	.896

4.6.3 Results

Figure 4.1 displays the performance curves of each method in terms of mean squared error (MSE) and mean ratio of the number of selected features. There is a large gap between the Pareto fronts of local and global feature selection, showing the usefulness of local selection in this setting. However, *all* of the baseline methods produce policies that improve over the Pareto front, which is *impossible without leakage*. For instance, TabNet only needs two features to achieve perfect prediction. Clearly, given the formula for y, this is impossible with predictors that truly only use two features. Therefore, our results prove that L2X, INVASE, TabNet and REAL-X *all have leakage*, and thus, that their selectors encode additional information into their selections. Even though REAL-X was specifically proposed to mitigate this issue by adding noise to h, our results prove that this strategy is not enough to prevent leakage. In contrast, SUWR is the only method that is close to the Pareto front and stays in the range of possible performance. As expected, because SUWR is guaranteed to have no leakage. In conclusion, these results conclusively prove that all of the baseline methods have leakage. To the best of our knowledge, we can therefore conclude that SUWR is the *first* and the *only* local feature selection method without leakage.

4.7 Experiment 2: Synthetic Benchmark

4.7.1 Setup

Whilst SUWR has excellent performance for the first experiment (Section 4.6), it concerned an idealized complete-information setting. Our second experiment aims to evaluate its generalizability by considering a more realistic setup where the training and test sets are separated. For a better comparison with previous work, we use an existing benchmark [24, 54, 141]. In this setup, eleven features, $x \in \mathbb{R}^{11}$, are sampled from a normal distribution: $x[i] \sim \mathcal{N}(0,1)$. Labels are binary, $y \in \{0,1\}$, and sampled according to $p(y = 1 \mid x) = \frac{1}{1+g(x)}$. The $g(x)$ function thus determines the relation between x and y. Six different $g(x)$ functions are used, the first three use non-overlapping sets of features:

- $g_1(x) = \exp(x[1]x[2])$;
- $g_2(x) = \exp(\sum_{i=3}^{6} x[i]^2 - 4)$;

- $g_3(x) \coloneqq -10\sin(2x[7]) + 2|x[8]| + x[9] + \exp(-x[10])$.

The latter three use a selection function based on the eleventh feature: $z(x, g, g') \coloneqq \mathbb{1}[x[11] < 0]g(x) + \mathbb{1}[x[11] \geq 0]g'(x)$, to choose between the first three functions:

- $g_4(x) \coloneqq z(x, g_1, g_2)$;
- $g_5(x) \coloneqq z(x, g_1, g_3)$;
- $g_6(x) \coloneqq z(x, g_2, g_3)$.

Thereby, the latter are specifically designed for local feature selection where the eleventh feature (called the *control-flow* feature) determines the relevance of the other features. We use 10,000 independent samples for training and another 10,000 as the test set.

4.7.2 Methods

The same methods are included as in the first experiment (Section 4.6). Additionally, we also train a predictor without feature selection (w/o FS) and another with an oracle selector that only selects the features used by $g(x)$ for each x.

4.7.3 Metrics

We use the same metrics as Jethani et al. [54]:

- the *true positive rate*: TPR = $\frac{\text{\# selected relevant features}}{\text{\# relevant features}}$;
- the *false discovery rate* FDR = $\frac{\text{\# selected irrelevant features}}{\text{\# selected features}}$;
- the *control-flow selection rate* (CFSR): the frequency of selecting the eleventh feature.

To measure predictive performance, we use the *area under the receiver operating characteristic curve* (AUROC). We note that a low CFSR score indicates leakage especially when TPR or AUROC is high, because it means the feature selection method actually uses the control-flow feature but does not select it.

4.7.4 Results

Table 4.2 displays our results on the synthetic benchmark test set. Interestingly, there is a large gap in the AUROC between the baseline without feature selection and the oracle baseline in all settings, this indicates that excluding irrelevant features can make prediction substantially easier.

In terms of AUROC, SUWR consistently has the highest performance of all methods (excluding the oracle), with especially high margins on the latter three settings (Syn4-6). In the first three settings (Syn1-3), SUWR reaches oracle performance; whilst among the other methods, only TabNet is able to reach oracle performance in the third setting (Syn3). This is surprising, since the first experiments showed that the existing methods could reach extremely high performance through leakage. However, a key difference with the first experiment is that in this setting evaluation is based on a held-out test set. Therefore, leakage could instead result in heavy overfitting in this setting, whereas it could not in

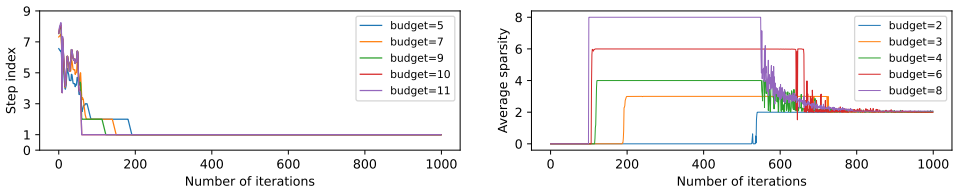


Figure 4.2: **Left**: at what step $\mathbf{x}_{11}$ is selected for Syn6. **Right**: selection budget vs. sparsity for Syn1, where only two features are relevant.

the first experiment. We believe that this explains why SUWR has substantially higher predictive performance for the second experiment: There are many more ways to overfit *with* leakage than *without*, as a result, SUWR is less prone to overfitting than the existing methods.

In terms of correct feature selection, SUWR has a near-perfect TPR that is greater than 97% across all settings and a perfect CFSR of 100% in the relevant settings (Syn4-6). REAL-X is the only baseline that has comparable TPR and CFSR across all settings. The FDR of SUWR is consistently lower than all baselines in all settings, except for INVASE which does better in the fourth and fifth setting (Syn4-5). Nevertheless, INVASE also has a very low CFSR and TPR in these settings, which strongly suggests that it is benefitting from leakage. Accordingly, the possibility of feature leakage makes it difficult to compare the feature sparsity of SUWR with the baselines. Nonetheless, in our results, SUWR has near-perfect TPR and perfect CFSR, and the best FDR of baselines with comparable TPR and CFSR.

We also show that SUWR consistently learns to select the control-flow feature first and that SUWR is very robust to the budget parameter T. Figure 4.2 shows the advantages of our method. Firstly, as shown in the left figure, our method is able to select the control-flow feature ($\mathbf{x}_{11}$) at the very first step, as its value determines the upcoming relevant features. We observed that for the other step-wise method TabNet, $\mathbf{x}_{11}$ is usually selected in a later step. Our method in this regard, provides a more interpretable reasoning logic for the selection decision. Furthermore, as the right figure shows, our method has the flexibility to allow us to either explicitly specify a selection budget without sparsity penalty, or figure out the right number of features by tuning a sparsity weight within a maximum selection budget, so that the model can squeeze out irrelevant features and converge to the optimal selection within the budget window.

In conclusion, our results on the synthetic benchmark reveal that SUWR reaches higher predictive performance than the baselines. We believe this is the case because leakage makes local feature selection methods more prone to overfitting, from which SUWR is unaffected. Furthermore, it also appears that SUWR selects nearly all relevant features while excluding more irrelevant features than baseline methods.

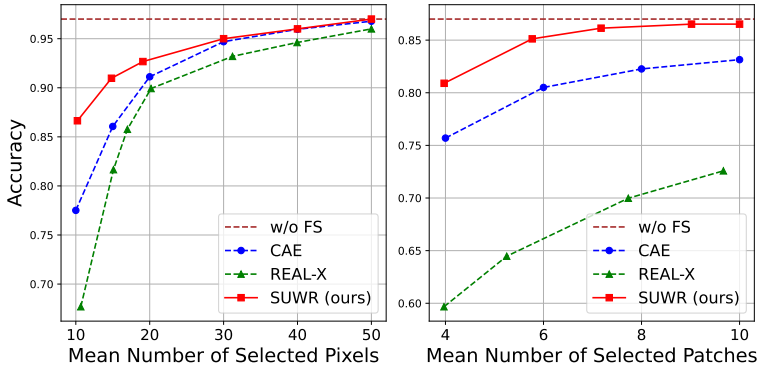


Figure 4.3: Results on MNIST: digits (left) and fashion (right).

4.8 Experiment 3: MNIST Digits and Fashion

4.8.1 Setup

Finally, we evaluate SUWR on an image classification task on two datasets: digits-MNIST [69] and fashion-MNIST [136]. Both datasets consist of 28×28 (784) pixel images and each image is annotated by one of ten classes, indicating either which digit or which type of fashion item is in the image. Because individual pixels are difficult to see in visualizations, we let the methods select 3×3 patches of pixels on the fashion dataset. As a result, the produced selection masks are much easier to interpret as selected pixels are less scattered.

4.8.2 Methods

We omit L2X and INVASE from this comparison due to their extremely unrealistic and unfaithful behavior in a previous study by Jethani et al. [54] (e.g., 96% accuracy while selecting a single pixel). Despite its leakage, we do include REAL-X since its introduction was motivated with its performance on digits-MNIST [54]. Additionally, we include the concrete autoencoder (CAE) [9], a state-of-the-art *global* feature selection method, and a predictor trained without any feature selection.

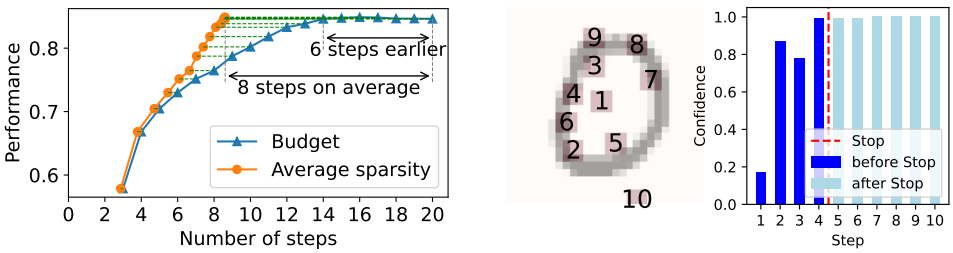


Figure 4.4: Selecting by patch on digits-MNIST. Early stopping before maximum step budget T.

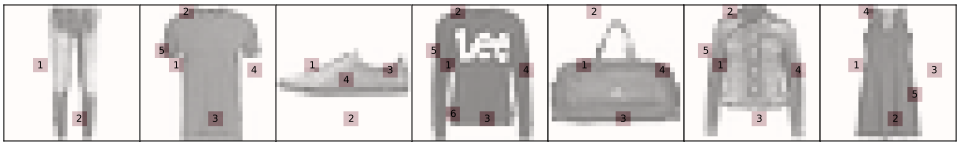


Figure 4.5: Several selection masks produced by SUWR for different fashion items from fashion-MNIST. Red squares indicate selected patches, the numbers shown inside indicate at what step each patch was selected. All items were correctly classified by SUWR.

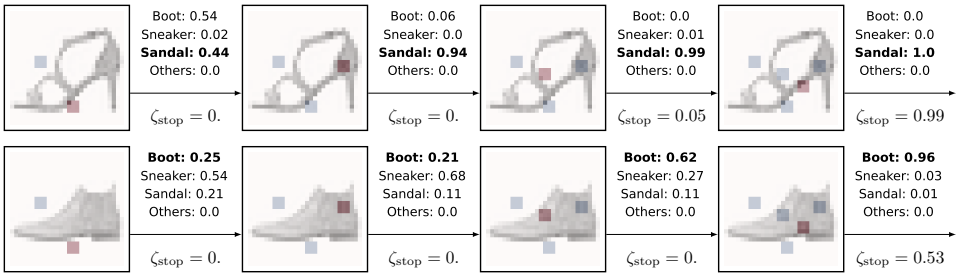


Figure 4.6: Narrative explanations derived from the SUWR inference process for a sandal (top) and boot (bottom) from fashion-MNIST. Step $t = 2$ up to $t = 5$ are visualized, red squares indicate patches selected in that step, blue squares those selected in previous steps.

4.8.3 Results

Figure 4.3 displays the performance curves of the methods in terms of accuracy and the number of selected pixels or patches on the test sets. We see that SUWR consistently outperforms both CAE and REAL-X on both datasets, and even approximates the performance of the baseline without feature selection while only selecting a fraction of the features. Admittedly, on digits-MNIST, the difference between SUWR and CAE becomes marginal when more than thirty pixels are selected, indicating that local selection is less beneficial on this dataset. In contrast, on fashion-MNIST, the differences between SUWR, CAE and REAL-X are considerably large; for instance, CAE with ten patches does not yet achieve the performance that SUWR reaches with just six patches. Surprisingly, REAL-X consistently has considerably lower performance than both SUWR and CAE. In other words, the local feature selection of REAL-X does substantially worse than even the global selections of CAE. We speculate REAL-X is overfitting due to leakage, and additionally, that its intentional injection of noise during optimization hinders its performance.

Additionally, we also include some patch-selection examples from digits-MNIST datasets. Figure 4.4 again shows the benefits of early stopping in reducing selection while maintaining performance. On the left side, we plot the average number of actual selections (i.e., the average sparsity) can be much smaller than the maximum selection budget T, under the same prediction performance. The right side gives a concrete image example of the digit 0. After 4 steps, the model (1) can correctly predict the digit with high confidence; and (2) is recommended to stop here by the stop probability. Continuing the selection will not affect the prediction performance.

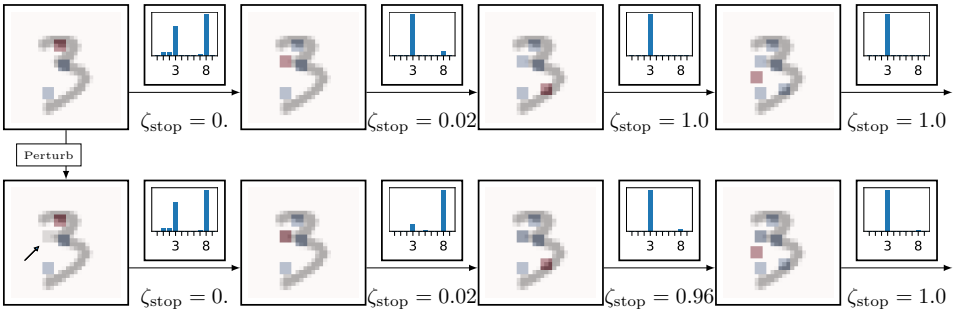


Figure 4.7: Digit 3 or 8? Each image (indicates one step) is masked by the colored squares and accompanied by a prediction bar chart and a stop probability ζ_{stop}. The red square is the new selection at the current step and the blue ones indicate the previous selections. The first two steps are omitted. The second row shows the same image as the top row, except for one particular patch which is filled with gray pixels, highlighted by the arrow. Due to this change, the prediction on the second row is flipped in the fourth step (second image from the left).

To conclude, our results on the MNIST datasets reveal that SUWR provides substantially better performance curves than REAL-X and CAE. Thereby, SUWR shows that local feature selection *without leakage* can provide considerably higher performance than global feature selection.

4.8.4 Interpretability

Lastly, we discuss several examples that illustrate how SUWR makes predictions more interpretable. Figure 4.5 displays the selection masks for several items in fashion-MNIST, and the order in which patches were selected. We see that the placement, order and number of selected patches highly varies per image. Because SUWR has no leakage, we are certain that no features outside of the selected patches were used for prediction. Thus, i.e., we know that the trousers were correctly classified based only on two patches. Similarly, the bag was classified using only four patches: three on its edges and an empty patch on the top. While these insights may be surprising, they are provenly faithful and thus provide an accurate account of the complete information that SUWR used for its predictions.

Figure 4.5 illustrates several steps in the SUWR inference process for a sandal and a boot. Besides what patches are selected per step, we also see how predictions and stop probabilities change as more features are selected. This brings numerous interesting insights; e.g., the differences in predictions between the two items at $t = 2$ can be attributed to a single pixel (top-left of the bottom patch). Additionally, we see that the predictor is already correct about the sandal after the third patch, but SUWR decides to select more features for more certainty.

Another example in Figure 4.7 shows the process of predicting an image of "3" with step-wisely selecting patches. The first three patches are enough to distinguish the image from the rest of classes except for "8", and the fourth path however, shows high discriminative information of "3" or "8". This is also supported by the minor perturbation of pixels in the fourth patch. When the fourth patch is not blank anymore, the prediction is flipped

from "3" to "8". This example shows a strong example of how the SUWR can explain the contribution of each feature to the prediction, which here gives much more insight than if one would highlight all selected features at once. To the best of our knowledge, SUWR is the first local feature selection method that provides narrative explanations that are guaranteed to be faithful.

4.9 Necessary and sufficient conditions for feature selection without label or feature leakage

Our formal proofs for the conditions for leakage will rely on two basic assumptions:

Assumption 4.9.1. The choice of selector policy has no effect on the label distribution:

$$\forall(x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega, \qquad p(y \mid x[s^{\text{in}}]) = p(y \mid x[s^{\text{in}}], \zeta). \tag{4.16}$$

Assumption 4.9.2. The choice of selector policy has no effect on the feature distribution:

$$\forall(x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega, \qquad p(x[s^{\text{ex}}] \mid x[s^{\text{in}}]) = p(x[s^{\text{ex}}] \mid x[s^{\text{in}}], \zeta). \tag{4.17}$$

Together, these assumptions entail that the *natural* distribution of features and labels is not dependent on the feature selector, i.e., ζ does not have any effect on the feature and label frequencies in the data.

Theorem 4.9.3. *A features selector does not have label leakage if and only if every probability for every possible feature selection does not depend on label values:*

$$\begin{aligned}&\Big(\neg \mathit{Label\text{-}Leakage}(\zeta)\Big)\\ &\quad\longleftrightarrow \Big(\forall(x, y, s^{in}, s^{ex}) \in \Omega, \quad p(h[s^{in}] = 1, h[s^{ex}] = 0 \mid x[s^{in}], \zeta)\\ &\quad= p(h[s^{in}] = 1, h[s^{ex}] \mathit{hspace1cm} = 0 \mid x[s^{in}], y, \zeta)\Big).\end{aligned} \tag{4.18}$$

Proof. First, we take Eq. 4.4 from Definition 4.3.3 and multiply both sides with $p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta)$, to get the following:

$$\begin{aligned}\forall(x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega, \quad p(y \mid x[s^{\text{in}}])p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta)\\ = p(y, h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta).\end{aligned} \tag{4.19}$$

From Assumption 4.9.1, we have $p(y \mid x[s^{\text{in}}]) = p(y \mid x[s^{\text{in}}], \zeta)$, from Definition 4.3.2 we know these values are positive, and thus we can divide each side of Eq. 4.19 by them:

$$\begin{aligned}\forall(x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega, \quad \frac{p(y \mid x[s^{\text{in}}])p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta)}{p(y \mid x[s^{\text{in}}])}\\ = \frac{p(y, h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta)}{p(y \mid x[s^{\text{in}}], \zeta)}.\end{aligned} \tag{4.20}$$

Reformulating each side of the above equation, results in:

$$\forall(x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega, \quad p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \,|\, x[s^{\text{in}}], \zeta) = p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \,|\, x[s^{\text{in}}], y, \zeta). \tag{4.21}$$

Thereby, we have proven that the condition for label leakage in Eq. 4.4 of Definition 4.3.3 implies the condition in Eq. 4.21. Since our derivation is still valid when reversed, it also proves Eq. 4.21 implies Eq. 4.4. Therefore, the conditions are logically equivalent, this completes our proof. □

Theorem 4.9.4. *A features selector does not have feature leakage if and only if every probability for every possible feature selection does not depend on non-selected feature values:*

$$\begin{aligned}&\Big(\neg \textit{Feature-Leakage}(\zeta)\Big)\\ &\quad\longleftrightarrow \Big(\forall(x, y, s^{in}, s^{ex}) \in \Omega, \quad p(h[s^{in}] = 1, h[s^{ex}] = 0 \mid x[s^{in}], \zeta)\\ &\quad = p(h[s^{in}] = 1, h[s^{ex}] = 0 \mid x[s^{in}], x[s^{ex}], \zeta)\Big).\end{aligned} \tag{4.22}$$

Proof. Analogous to the proof for Theorem 4.9.3, we begin by taking Eq. 4.5 from Definition 4.3.4 and multiply both sides with $p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta)$, to get the following:

$$\begin{aligned}\forall(x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega, \quad & p(x[s^{\text{ex}}] \mid x[s^{\text{in}}])p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0, \mid x[s^{\text{in}}], \zeta)\\ &= p(x[s^{\text{ex}}], h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0, \mid x[s^{\text{in}}], \zeta).\end{aligned} \tag{4.23}$$

From Assumption 4.9.2, we have $p(x[s^{\text{ex}}] \mid x[s^{\text{in}}]) = p(x[s^{\text{ex}}] \mid x[s^{\text{in}}], \zeta)$, from Definition 4.3.2 we know these values are positive, and thus we can divide each side of Eq. 4.23 by them:

$$\begin{aligned}\forall(x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega, \quad & \frac{p(x[s^{\text{ex}}] \mid x[s^{\text{in}}])p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0, \mid x[s^{\text{in}}], \zeta)}{p(x[s^{\text{ex}}] \mid x[s^{\text{in}}])}\\ &= \frac{p(x[s^{\text{ex}}], h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0, \mid x[s^{\text{in}}], \zeta)}{p(x[s^{\text{ex}}] \mid x[s^{\text{in}}], \zeta)}.\end{aligned} \tag{4.24}$$

Reformulating each side of the above equation, results in:

$$\begin{aligned}\forall(x, y, s^{\text{in}}, s^{\text{ex}}) \in \Omega, \quad & p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta)\\ &= p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], x[s^{\text{ex}}], \zeta).\end{aligned} \tag{4.25}$$

Thereby, we have proven that the condition for feature leakage in Eq. 4.5 of Definition 4.3.3 implies the condition in Eq. 4.25. Since our derivation is still valid when reversed, it also proves Eq. 4.25 implies Eq. 4.5. Therefore, the conditions are logically equivalent, this completes our proof. □

4.10 Local Feature Selection with SUWR has no Leakage

Theorem 4.10.1. *All SUWR feature-selection policies have no leakage. In other words, if the inference of a policy ζ is computable with SUWR then it has no leakage according to Definition 4.3.1.*

Proof. If ζ is computable by the inference algorithm of SUWR, then it performs at most T steps to make a selection. From Algorithm 1, we see that the creation of a selection

ends when a Bernoulli trail with a probability determined by ζ_{stop} succeeds. Therefore, the probability $\zeta(h \mid x)$ can be written as an expectation over T steps; let $q(t = i, h \mid x, \zeta)$ indicate the probability that SUWR reaches step $t = i$ and with the mask h, we can then formulate $\zeta(h \mid x)$ as:

$$\zeta(h \mid x) = q(t = T, h \mid x, \zeta) + \sum_{i=0}^{T-1} q(t = i, h \mid x, \zeta)\zeta_{\text{stop}}^{t=i}(x \odot h). \tag{4.26}$$

In less formal terms, it is a sum over the probability of reaching each possible step and the mask being h at that step multiplied with the probability of stopping at that step. Thus, $q(t = i, h \mid x, \zeta)$ is the probability of SUWR *reaching* a step, *not necessarily stopping* at that step. Accordingly, in the first step ($t = 0$), the mask is always the empty mask, therefore:

$$q(t = 0, h = \mathbf{0} \mid x, \zeta) = 1, \qquad q(t = 0, h \neq \mathbf{0} \mid x, \zeta) = 0. \tag{4.27}$$

To keep our notation brief, we call a mask a subset of another mask if it selects the same or a subset of features:

$$h' \subseteq h \longleftrightarrow (\forall i,\ h'[i] = 1 \longrightarrow h[i] = 1). \tag{4.28}$$

This enables us to give a short definition general definition of $q(t, h \mid x, \zeta)$ by using its recursive nature:

$$q(t, h \mid x, \zeta) = \begin{cases} 1, & \text{if } t = 0 \wedge h = \mathbf{0}, \\ 0, & \text{if } t = 0 \wedge h \neq \mathbf{0}, \\ \sum\limits_{h' : h' \subseteq h} q(t-1, h' \mid x, \zeta)(1 - \zeta_{\text{stop}}^{t-1}(x \odot h')) \sum\limits_{u \in \{0,1\}^d : h' + u = h} \zeta_{\text{select}}^{t-1}(u \mid x \odot h'), & \text{otherwise.} \end{cases} \tag{4.29}$$

Thus, when $t > 0$, the value of $q(t, h \mid x, \zeta)$ is a sum over the probability that the previous step $(t-1)$ was reached with a subset of $h' \subseteq h$, and that the SUWR process did not stop, and that a new feature mask u was sampled such that $h = h' + u$. This recursion ends when $t = 0$ is reached.

Clearly, we can see from Eq. 4.29 that for $t = 0$ the q function is not conditioned on x:

$$q(t = 0, h = \mathbf{0} \mid x, \zeta) = q(t = 0, h = \mathbf{0} \mid \zeta), \qquad q(t = 0, h \neq \mathbf{0} \mid x, \zeta) = q(t = 0, h \neq \mathbf{0} \mid \zeta), \tag{4.30}$$

and therefore:

$$q(t = 0, h \mid x, \zeta) = q(t = 0, h \mid \zeta). \tag{4.31}$$

Similarly, at $t = 1$ the following holds:

$$q(t = 1, h \mid x, \zeta) = q(t = 0, h = \mathbf{0} \mid \zeta)(1 - \zeta_{\text{stop}}^{t=0}(\emptyset))\zeta_{\text{select}}^{t=0}(h \mid \emptyset), \tag{4.32}$$

and therefore:

$$q(t = 1, h \mid x, \zeta) = q(t = 1, h \mid \zeta). \tag{4.33}$$

We can continue this pattern by considering Eq. 4.29, where we can see that when $t > 0$ the ζ_{stop} and ζ_{select} only take subsets of h as input. Similarly, through the recursion of q

only subsets of h are given as input to q, therefore, the recursion cannot add a dependency on any feature value not selected by h. Consequently, the value of $q(t, h \mid x, \zeta)$ does not depend on any values of x not selected by h:

$$q(t, h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x, \zeta) = q(t, h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta). \tag{4.34}$$

Finally, combining this result with Eq. 4.26, we see that the final stop probability also does not add a dependency on feature values not selected by h, therefore:

$$\zeta(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x) = \zeta(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}]). \tag{4.35}$$

According to Corollary 4.3.5, this proves that ζ does not have leakage. □

4.11 Conjecture: SUWR Describes any Selection Policy without Leakage under Full-Support Feature Distributions

Assumption 4.11.1. The feature value distribution provides support for the Cartesian product of possible feature values. In other words, if there is a positive probability that feature $x[i]$ has value a and a positive probability that feature $x[j]$ has value b, then there is a positive probability that feature $x[i]$ has value a *and* feature $x[j]$ has value b simultaneously:

$$\forall i, j, a, b, \quad \big(p(x[i] = a) > 0 \wedge p(x[j] = b) > 0\big) \longrightarrow p(x[i] = a, x[j] = b) > 0. \tag{4.36}$$

Definition 4.11.2. We define a *reversed directed Hasse diagram* (RDHD) SUWR policy as a SUWR policy where the maximum step is the number of features: $T = d$, and $\zeta^t_{\text{select}}(x \odot h)$ is a distribution over all single features that have not been selected yet:

$$\zeta^t_{\text{select}}(u \mid x \odot h) \begin{cases} \geq 0 & \text{if } \big(\exists! i,\ u[i] = 1\big) \wedge \big(\forall i \in \{1, 2, \ldots, d\},\ u[i] = 1 \longrightarrow h[i] = 0\big), \\ = 0 & \text{otherwise.} \end{cases} \tag{4.37}$$

Thereby, at each step t, the process either stops or a single feature is added to h. As a result, the number of features selected by h^t is always equal to t: $\sum_{i=1}^{d} h^t[i] = d$. An example visualization of the possible steps of a RDHD SUWR policy for three features is shown in Figure 4.8.

The naming of this type of policy is inspired by the fact that the inference process of a RDHD SUWR policy can be visualized as traversing over a Hasse diagram (e.g., in Figure 4.8). Traditionally, Hasse diagrams are constructed from the complete set and are not directed. In contrast, RDHD SUWR policies start with the empty set and explicitly only traverse in the direction where elements are added. Hence, we name it after a *reversed* and *directed* version of the Hasse diagram.

Conjecture 4.11.3. Under the assumption that the feature distribution supports the Cartesian product of possible feature values (Assumption 4.11.1), every feature selection policy ζ that has no leakage (Definition 4.3.1) has an equivalent RDHD SUWR policy. In other words, the set of all possible feature selection policies without leakage is a subset of the set of all possible RDHD SUWR policies.

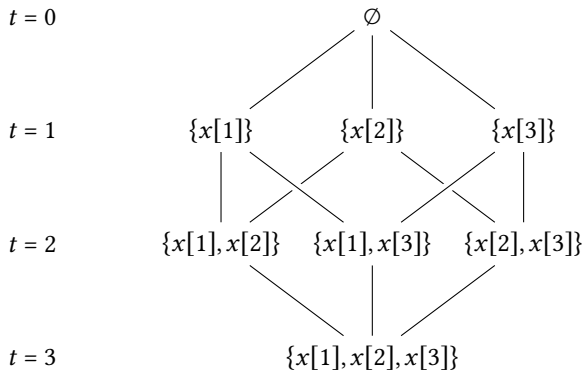


Figure 4.8: Visualization of all possible steps and transitions for a RDHD SUWR policy when selecting from a set of three features.

Support. We will provide reasons to support that, under Assumption 4.11.1, for any ζ without leakage, there exists a ζ^t_{stop} and ζ^t_{select} for a RDHD SUWR policy, such that ζ and the RDHD SUWR policy have an identical distribution over feature selections.

For this section, the same q function is used as for Theorem 4.10, but to keep our notation short, we will use $q(h \mid x \odot h)$ instead of $q(t, h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta_{\text{stop}}, \zeta_{\text{select}})$. We can do this without loss of specificity since each h can only occur at a single specific step: $t = \sum_{i=1}^{d} h[i]$, we only consider q in the context of ζ_{stop} & ζ_{select}, and we have already proven that q only depends on the features selected by h, i.e., $x \odot h$ (see Theorem 4.10). In other words, we use $q(h \mid x \odot h)$ as the probability that the SUWR process at some point *considers* mask h, this is not the probability that h is selected.

This difference reveals the requirement on the SUWR policy, the probability of considering h should be equal to or greater than the probability to select h:

$$\forall h, x, \qquad p(x) > 0 \longrightarrow q(h \mid x \odot h) > \zeta(h \mid x \odot h). \tag{4.38}$$

This requirement exists because the probability of selecting h in a RDHD SUWR policy is equal to:

$$\forall h, x, \qquad p(x) > 0 \longrightarrow \zeta(h \mid x \odot h) = q(h \mid x \odot h)\zeta_{\text{stop}}(h \mid x \odot h). \tag{4.39}$$

Therefore, the probabilities $\zeta_{\text{stop}}(h \mid x \odot h)$ have to be:

$$\forall h, x, \qquad p(x) > 0 \longrightarrow \zeta_{\text{stop}}(h \mid x \odot h) = \frac{\zeta(h \mid x \odot h)}{q(h \mid x \odot h, \zeta_{\text{stop}}, \zeta_{\text{select}}),} \tag{4.40}$$

this is a valid probability, i.e., $\zeta_{\text{stop}}(h \mid x \odot h) \in [0, 1]$, if Eq. 4.38 is true, i.e., $q(h \mid x \odot h) > \zeta(h \mid x \odot h)$.

Thus, we have to choose ζ_{select} such that Eq. 4.38 is guaranteed to hold. To keep our notation short, we denote $\zeta^t_{\text{select}}(i \mid x \odot h)$ for the selection of feature i, i.e., the sampling of a vector u such that only element i is one: $u[i] = 1$ and all other values are zero: $i \neq j \leftrightarrow u[j] = 0$, conditioned on the feature values of x selected by h.

Our key insight is that every time a RDHD SUWR policy samples a feature, it is excluding a set of possible selections, which can no longer be reached afterwards. Instead of thinking about how the SUWR process includes features into its selection, we consider the possible feature selections it excludes through the addition of each feature. The following set covers all masks that can no longer be reached after i is sampled by SUWR to be added to mask h:

$$\text{excluded}(h, i) = \{h' \,:\, h[i] = 0 \wedge \forall j \in \{1, 2, \ldots, d\}, h[j] = 1 \longrightarrow h'[j] = 1\}. \tag{4.41}$$

As we can see, each mask in $\text{excluded}(h, i)$ makes the same selections as h, in addition to every other possible selection, that does not select i as well. We note that when the set is empty when i has already been selected in h: $h[i] = 1 \longrightarrow \text{excluded}(h, i) = \emptyset$. Therefore, the probability that feature i is added to selection h should not exceed the following:

$$\begin{aligned} & \underbrace{q(h \mid x \odot h, \zeta_{\text{stop}}, \zeta_{\text{select}})(1 - \zeta_{\text{stop}}(i \mid x \odot h))\zeta_{\text{select}}(i \mid x \odot h)}_{\text{probability of reaching } h \text{ and adding } i} \\ & \leq 1 - \max_{\{x[j] \,:\, h[j]=0 \wedge i \neq j\}} \underbrace{\sum_{h' \in \text{excluded}(h,i)} \zeta(h' \mid x \odot h)}_{\text{max. prob. of selections no longer accessible afterwards}} . \end{aligned} \tag{4.42}$$

This leads to the following restriction on ζ_{select}:

$$\zeta_{\text{select}}(i \mid x \odot h) \leq \frac{1 - \max_{\{x[j] \,:\, h[j]=0 \wedge i \neq j\}} \sum_{h' \in \text{excluded}(h,i)} \zeta(h' \mid x \odot h)}{q(h \mid x \odot h, \zeta_{\text{stop}}, \zeta_{\text{select}})(1 - \zeta_{\text{stop}}(i \mid x \odot h))} \tag{4.43}$$

This maximum restriction ensures that these selections remain reachable by the RDHD SUWR policy with the required probability. Thereby ensuring the requirement in Eq. 4.38 is true. Importantly, this maximum can be inferred without knowledge of feature values that are not selected in h, thus it can be incorporated without introducing leakage.

However, not selecting feature i also excludes a possible selection. Namely, the selection that is made by only adding feature i to h, as this can no longer be reached after the addition of a different feature. We denote this mask as h^{+i}:

$$h^{+i} \in \{0, 1\}^d \quad \text{s.t.} \quad h[i] = 1 \wedge \forall j \in \{1, 2, \ldots, d\}, i \neq j \leftrightarrow h'[j] = h[j]. \tag{4.44}$$

Therefore, the probability of reaching h and selecting i must be at least as great as the maximal possible probability of h^{+i} conditioned on the feature values selected so far:

$$\max_{x[i]} \zeta(h^{+i} \mid x \odot h^{+i}) \leq q(h \mid x \odot h, \zeta_{\text{stop}}, \zeta_{\text{select}})(1 - \zeta_{\text{stop}}(i \mid x \odot h))\zeta_{\text{select}}(i \mid x \odot h). \tag{4.45}$$

This results in the following restriction on ζ_{select}:

$$\frac{\max_{x[i]} \zeta(h^{+i} \mid x \odot h^{+i})}{q(h \mid x \odot h, \zeta_{\text{stop}}, \zeta_{\text{select}})(1 - \zeta_{\text{stop}}(i \mid x \odot h))} \leq \zeta_{\text{select}}(i \mid x \odot h). \tag{4.46}$$

Again, we note that this restriction can be enforced without introducing leakage as the minimum value only depends on feature values that are not selected in h.

By combining the requirement in Eq. 4.46 and Eq. 4.43, we see that we need the following requirement to be true:

$$\max_{x[i]} \zeta(h^{+i} \mid x \odot h^{+i}) + \max_{\{x[j]\,:\,h[j]=0 \wedge i \neq j\}} \sum_{h' \in \text{excluded}(h,i)} \zeta(h' \mid x \odot h) \leq 1. \tag{4.47}$$

Since if Eq. 4.47 is not true, there is no value of $\zeta_{\text{select}}(i \mid x \odot h)$ that can satisfy both Eq. 4.46 and Eq. 4.43.

We will now show that under Assumption 4.11.1, the requirement in Eq. 4.47 is always guaranteed.[3] To start, we use the following to denote the feature values that maximize each of the maximum operations:

$$\begin{aligned} x[i]^* &= \arg\max_{x[i]} \zeta(h^{+i} \mid x \odot h^{+i}), \\ \{x[j]^*\} &= \max_{\{x[j]\,:\,h[j]=0 \wedge i \neq j\}} \sum_{h' \in \text{excluded}(h,i)} \zeta(h' \mid x \odot h). \end{aligned} \tag{4.48}$$

Importantly, feature i is not selected by any mask in the excluded set: $\forall h' \in \text{excluded}(h,i),\ h[i] = 0$. Therefore, there is no overlap between $x[i]^*$ and $\{x[j]^*\}$, this means that a possible vector of feature values exists that includes $x[i]^*$, $\{x[j]^*\}$ and $x \odot h$. We denote this combination of possible values as:

$$\exists x^* \,:\, x^*[i] = x[i]^* \wedge \left(\forall x[j]^*,\, x^*[j] = x[j]^*\right) \wedge x \odot h = x^* \odot h. \tag{4.49}$$

By the definition of x^*, $x[i]^*$ and $\{x[j]^*\}$, this vector maximizes both parts of the left side of Eq. 4.47:

$$\begin{aligned} \zeta(h^{+i} \mid x^* \odot h^{+i}) &= \max_{x[i]} \zeta(h^{+i} \mid x \odot h^{+i}), \\ \sum_{h' \in \text{excluded}(h,i)} \zeta(h' \mid x^* \odot h) &= \max_{\{x[j]\,:\,h[j]=0 \wedge i \neq j\}} \sum_{h' \in \text{excluded}(h,i)} \zeta(h' \mid x \odot h). \end{aligned} \tag{4.50}$$

Assumption 4.11.1 states that every possible combination of feature values is supported by the feature distribution, therefore: $p(x^*) > 0$. ζ is a valid probability distribution over all possible feature masks. For every possible value of x, this means the sum of all probabilities of all masks cannot be greater than one. Therefore, the same goes for this subset of masks:

$$p(x^*) > 0 \longrightarrow \zeta(h^{+i} \mid x^* \odot h^{+i}) + \sum_{h' \in \text{excluded}(h,i)} \zeta(h' \mid x^* \odot h) \leq 1. \tag{4.51}$$

Consequently, the requirement in Eq. 4.47 is guaranteed to hold under Assumption 4.11.1, and therefore, there always exists a value for $\zeta_{\text{select}}(i \mid x \odot h)$ that can satisfy both Eq. 4.46 and Eq. 4.43.

[3] For comparison, Table 4.3 displays an example where Assumption 4.11.1 is not true, and accordingly, Conjecture 4.11.3 does not hold.

Table 4.3: Example of a feature selection policy *without* leakage that is *impossible* to compute with SUWR. This happens because the feature distribution does not support the Cartesian product of possible feature values, as stated in Assumption 4.11.1. In this example, knowing that $x[1] = 1$ means one also knows $x[2] = 0$ and $x[3] = 0$, therefore, these selections can be safely removed once $x[1] = 1$ is known, without introducing leakage. Since SUWR is agnostic to the underlying feature distribution, it does not use this property to enable the removal of features after their selection. Consequently, the displayed policy cannot be executed through the SUWR algorithm. (See Table 4.1 for an explanation of the notation).

$p(x,y,h)$	$x[1]$	$x[2]$	$x[3]$	$h[1]$	$h[2]$	$h[2]$	$(x \odot h)[1]$	$(x \odot h)[2]$	$(x \odot h)[3]$	y
0.333…	1	0	0	1	0	0	1			2
0.333…	0	1	0	0	1	0		1		1
0.333…	0	0	1	0	0	1			1	0

4

Unfortunately, this does not provide a complete proof, since there is an additional requirement that we were unable to prove. Namely, Eq. 4.46 can only hold if the following is true:

$$\frac{\max_{x[i]} \zeta(h^{+i} \mid x \odot h^{+i})}{q(h \mid x \odot h, \zeta_{\text{stop}}, \zeta_{\text{select}})(1 - \zeta_{\text{stop}}(i \mid x \odot h))} \leq 1, \tag{4.52}$$

since otherwise, Eq. 4.46 implies that $\zeta_{\text{select}}(i \mid x \odot h) > 1$ which would make it an invalid policy. A simple reformulation reveals that this is a requirement on q:

$$q(h \mid x \odot h, \zeta_{\text{stop}}, \zeta_{\text{select}}) \leq \frac{\max_{x[i]} \zeta(h^{+i} \mid x \odot h^{+i})}{1 - \zeta_{\text{stop}}(i \mid x \odot h)}. \tag{4.53}$$

In other words, the probability of h being considered conditioned on $x \odot h$, ζ_{stop} and ζ_{select} needs to be great enough to provide enough probability mass for both the maximum possible probability of h and h^{+i}. For very small problems with two binary features, we are able to find a closed-form solution that gaurantees this. Unfortunately, we were unable to extend this approach to a more generic setting. Nonetheless, it appears that satisfying both Eq. 4.46 and Eq. 4.43 also guarantees Eq. 4.53, but until this is proven our claim can only remain a conjecture.

4.12 Details on the Linear Programming Approach

For our linear programming approach, we assume that the problem is fully known, thus complete knowledge of $p(x, y)$ is available. In addition, we assume that the set of possible feature and label values is finite and iterable. As a result, the optimal predictor f^* can be treated as a lookup table that stores the optimal prediction per possible selected feature values. For simplicity, we assume that the optimal prediction value is the expected label conditioned on the selected feature values:

$$f^*(x \odot h) = \mathbb{E}_x[y \mid x \odot h] = \sum_{x' \,:\, x' \odot h = x \odot h} p(x') \sum_y p(y \mid x') y = \sum_{x' \,:\, x' \odot h = x \odot h} p(x') \sum_y p(y \mid x') y. \tag{4.54}$$

Therefore, we only have to find the optimal selector policy ζ. Our linear programming approach poses the search as a constrained minimization problem in the following form [29,

125]:

$$\min_{\theta} c^T\theta \quad \text{s.t.} \quad A\theta = b \wedge \mathbf{0} \leq \theta \leq \mathbf{1}, \tag{4.55}$$

where θ is a vector where each element represents the conditional probability of a selection $\zeta(h \mid x)$. The remainder of this section will show how the vectors b and c and matrix A can be constructed so that this minimization problem is equivalent to selection policy optimization.

To start, we will show how c and θ can be chosen so that $c^T\theta = \mathscr{L}(\zeta, f^*)$ (cf. Eq. 4.2). Importantly, we want our selection policy to have no leakage, as discussed in Section 4.3.3 this means that:

$$\forall(x, x', h), \quad x \odot h = x' \odot h \longrightarrow \zeta(h \mid x) = \zeta(h \mid x'). \tag{4.56}$$

Therefore, we only have to find a single conditional probability $\zeta(h \mid x)$ for every unique $x \odot h$ value. Thus, the size of vector x is going to be the number of unique possible selected feature values, where each element corresponds to a single $x \odot h$ and contains the value for all corresponding $\zeta(h \mid x)$ values. To see that our loss can be rewritten as a dot product with such a vector, we rewrite it as follows:

$$\begin{aligned}\mathscr{L}(\zeta, f^*) &= \sum_{x,y} p(x,y) \sum_{h} \left[\zeta(x \odot h) L(f^*(x \odot h), y) + \lambda \|h\|\right] \\ &= \sum_{x \odot h} \zeta(h \mid x) \sum_{x' : x' \odot h = x \odot h} p(x') \Big(\lambda \|h\| + \sum_{y} p(y) L(f^*(x \odot h), y)\Big),\end{aligned} \tag{4.57}$$

where the summation $\sum_{x \odot h}$ sums over every possible value of $x \odot h$ once. In other words, if multiple feature values result in the same selected feature values e.g., $x \odot h = x' \odot h$, only one of them is considered in the $\sum_{x \odot h}$ sum. From the above reformulation, we see that for $c^T\theta = \mathscr{L}(\zeta, f^*)$ we require:

$$\forall(x, h), \exists! i \in \mathbb{N}_{>0}, \quad \theta_i = \zeta(h \mid x) \wedge c_i = \sum_{x' : x' \odot h = x \odot h} p(x') \Big(\lambda \|h\| + \sum_{y} p(y) L(f^*(x \odot h), y)\Big). \tag{4.58}$$

Algorithm 2 shows how we construct c accordingly: first a mapping is made for every possible selected feature value ($x \odot h$) to an index on c, next the value of each element of c is computed following Eq. 4.58 and stored in the corresponding position.

Besides minimizing $\mathscr{L}$, it is important that the ζ is a valid probability distribution. To be more precise, for all possible values of the full set of features x, ζ should produce a valid distribution over all possible selections ($\zeta(h \mid x)$). We can express this formally in the following manner:

$$\forall x,\ p(x) > 0 \longrightarrow \Big(\sum_{h \in \zeta(x)} \zeta(h \mid x) = \sum_{s^{\text{in}}, s^{\text{ex}} : s^{\text{in}} \cup s^{\text{ex}} = \{1,2,\ldots,d\}} p(h[s^{\text{in}}] = 1, h[s^{\text{ex}}] = 0 \mid x[s^{\text{in}}], \zeta) = 1 \Big). \tag{4.59}$$

For the linear program, this requirement can be expressed through the A matrix and b vector in a straightforward manner. We set $b = \mathbf{1}$ as a vector of ones with the size of the number of possible values for x. The matrix A gets a first dimension with the same size as b and the second dimension the same size as θ. Thereby, each row corresponds to a possible

value of x and each column to a possible value of $x \odot h$. Algorithm 2 iterates over each row, representing a possible value of x, and then selects each column that corresponds to a possible set of masked features that could occur for x and sets it to one. As a result, $A\theta = b$ indicates that the probability distribution $\zeta(h \mid x)$ sums to one for each possible value of x.

Having constructed A, b and c, we use SciPy to solve the linear programming problem of Eq. 4.55 [128] and find the optimal value of θ. Correspondingly, the output of Algorithm 2 is a lookup-table representing the optimal predictor f^*, the vector θ containing the optimal probabilities for ζ, and an index that maps each (x, h) to the element in θ that contains the corresponding $\zeta(h \mid x)$ value. If the linear programming solver functions correctly, this solution represents the optimal predictor and selector policies possible for the task. In our experimental analysis, we assume that the produced solutions are a close approximation of the optimal policies.

Algorithm 2 Our linear programming approach.

Input: Set of possible features: X, Set of possible labels: Y, Set of possible masks: H, Probability distribution function: $p(x, y)$, Loss: L, Sparsity weight: λ.
feat_index $\leftarrow \{\}$ *# Empty dictionary to map possible masked feature values to indices.*
feat_labels $\leftarrow \{\}$ *# Empty dictionary to keep track of label values.*
$N_{\text{unique}} \leftarrow 0$ *# Counter tracking number of possible unique selected feature values.*
for $x \in X, y \in Y : p(x, y) > 0$ **do**
  for $h \in H$ **do**
    if $x \odot h \notin$ feat_index **then**
      $N_{\text{unique}} \leftarrow N_{\text{unique}} + 1$
      feat_index$[x \odot h] = N_{\text{unique}}$ *# If unique, the value $x \odot h$ receives the next available index.*
      feat_labels$[x \odot h] = \emptyset$ *# Initialize an empty set for every possible associated label value.*
    end if
    feat_labels$[x \odot h] \leftarrow$ feat_labels$[x \odot h] \cup \{(y, p(x, y))\}$ *# Possible labels and cond. probabilities stored per $x \odot h$.*
  end for
end for
$c \leftarrow$ zero_vector(N_{unique}) *# Zero initialization of cost vector of size N_{unique}.*
$f^* \leftarrow \{\}$ *# Empty dictionary to store optimal predictor.*
for $x \odot h \in$ feat_index **do**
  $p(x \odot h) \leftarrow \sum_{(y,p(x,y)) \in \text{feat\_labels}[x \odot h]} p(x, y)$ *# Natural probability of the selected feature values.*
  $f^*(x \odot h) \leftarrow \frac{1}{p(x \odot h)} \sum_{(y,p(x,y)) \in \text{feat\_labels}[x \odot h]} p(x, y) y$ *# Assumption: Optimal prediction is the expected value.*
  $i \leftarrow$ feat_index$[x \odot h]$
  $c[i] \leftarrow \sum_{(y,p(x,y)) \in \text{feat\_labels}[x \odot h]} p(x, y) \big(L(f^*(x \odot h), y) + \lambda |h| \big)$
end for
$A \leftarrow$ zero_matrix$(|X|, N_{\text{unique}})$ *# Zero initialization of constraint matrix of size $|X| \times N_{unique}$.*
$i \leftarrow 0$
for $x \in X$ **do**
  $i \leftarrow i + 1$
  for $h \in H$ **do**
    $j \leftarrow$ feat_index$[x \odot h]$
    $A[i, j] \leftarrow 1$ *# Setting ones for every possible selected feature values per row for each x.*
  end for
end for
$b \leftarrow$ one_vector$(|X|)$ *# Vector of size $|X|$ (number of possible feature values) filled with ones.*
$\theta \leftarrow$ Linear_Program_Solver(A, b, c) *# Solves Eq. 4.55, outputs vector of size N_{unique} with ordering matching feat_index.*
Return: $f^*, \theta,$ feat_index =0

4.13 Conclusion

RQ3: Are self-explainable models faithful and how to design theoretically guaranteed faithful models? We have shown that self-explainable models via local feature selection are not always faithful and their prediction might come from additional information other than solely selected features due to feature or label leakage. This work has provided the first formal definition of feature and label leakage, which causes local feature selection methods to provide misleading explanations of what information predictions are based on. To design theoretically guaranteed faithful models, we derived the necessary and sufficient conditions for leakage and introduced the first methods that are guaranteed to have no leakage: a linear programming method and SUWR. Our experimental results reveal that existing state-of-the-art methods are all subject to leakage, in addition to being misleading, this also appears to make them more prone to overfitting. In contrast, our results indicate that SUWR combines high selection sparsity with high predictive accuracy, outperforming all our baselines across several benchmarks. Uniquely, the step-by-step SUWR process can be used as a narrative explanation itself. The SUWR approach is generic and easily extendable, we believe it can serve as a strong foundation for future work on faithful interpretable ML predictions with theoretical guarantees.

In particular, future work could consider methods to scale the SUWR approach to high-dimensional data. For instance, by developing model architectures that can be applied efficiently in the SUWR framework. Alternatively, one could investigate whether our definitions of leakage could provide a basis for novel indicators of feature importance. In order to promote the future extension of our work, we have made the implementations of our method and experiments publicly available[4].

[4] https://github.com/GarfieldLyu/SUWR

5

Conclusion

Information retrieval (IR) is the foundational step for many knowledge-intensive tasks, underscoring the urgent need for transparent and reliable IR systems. However, the complexity of IR models, often functioning as black boxes with neural components, makes it unclear how individual items are ranked or decisions are made. This opacity can encode biases, such as those related to gender and ethnicity, undermining the system's reliability. Moreover, crucial factors for model decisions may be hidden or based on unintended patterns, going unnoticed and unaligned with the correct human rationale. Additionally, neural models are known to be fragile and susceptible to subtle perturbations.

Correspondingly, this thesis aims at improving the interpretability of IR models. In general, our contributions fall into two broad diagrams including (1). post-hoc explanation of neural text ranking models, and (2). building intrinsically interpretable models. We draw inspiration from interpretable machine learning (ML) while also considering the unique aspects of IR to enhance ranking-related explanations. Our efforts extend beyond bridging research gaps between these domains; they also contribute to the theoretical foundations of interpretable ML. However, despite the potentially significant societal impact of creating transparent and trustworthy systems, our work faces several challenges, particularly in the era dominated by large language models (LLMs). In the rest of this section, we will first present a summary of the findings and contributions covered in this thesis. Then, we explore the social implications of our contributions. Lastly, we conclude this thesis by identifying limitations and proposing potential directions for future research.

5.1 Summary of Contributions

5.1.1 Query Expansion as Post-hoc Explanations

RQ1 How do we explain ranking-specific decisions from black-box text ranking models?

To address the research question at hand, we begin by highlighting the distinctions between ranking domains and other conventional machine learning tasks. In ranking, decisions are derived from pointwise predictions that are then aggregated to form rank pairs and lists. This aggregation process inherently complicates the explanation of ranking out-

puts, as it requires consideration of multiple items, often numbering in the hundreds. Secondly, we argue that classical IR relevance factors (or axioms) such as term frequency, semantic similarity, etc., are helpful for users to explain ranking decisions. Additionally, it is crucial to acknowledge that users convey their search intent through queries. Whether the machine learning model accurately perceives this intent is often unclear. Hence, we argue that the initial and most vital step in explaining a ranking model is to assess its comprehension of the search query.

In light of these considerations, we propose utilizing query expansion as a means to elucidate the model's understanding of the query, which subsequently leads to a specific ranking outcome. Our method called MULTIPLEX involves a post-hoc, model-agnostic framework designed to explain text ranking models through the use of multiple explainers. We employ classical IR models as straightforward proxy explainers to demystify the workings of complex ranking models within the framework of established relevance factors. To enhance the fidelity of these explanations, we directly optimize the degree of preference coverage using linear programming. This allows us to flexibly operate on any number of explainers and choose any size of expansions. Our extensive experimental results demonstrate that our approach can produce high-fidelity explanations for over-parameterized models, such as BERT, achieving fidelity improvements of up to 54%.

Reflections. Our method explains a ranking by a set of terms attributed to a union of multiple explainers. Balancing efficiency and fidelity (preference coverage) is crucial for developing effective explanation algorithm. Future work may consider extending our framework to account for n-grams or short terms to enhance readability. Additionally, it is interesting to examine which explainer (or ranking heuristic) contributes to which extent using which particular terms. Moreover, it is well known that validating explanations is challenging, especially in the absence of ground-truth data. While we measure fidelity in this work, it might not fully capture the underlying logic of a complex model. Therefore, incorporating human perspectives into the evaluation process and balancing the cost of annotating numerous decisions in a ranking are important areas for future exploration.

5.1.2 Adapting Interpretable ML to Interpretable Ranking

RQ2 Is interpretable ML applicable for building self-explainable ranking models?

As discussed in previous sections, self-explainable models have increasingly gained traction in interpretable machine learning (ML) due to their inherent transparency. One widely recognized method for achieving transparency is through feature selection. Although this concept is well-established and extensively researched in interpretable ML, it has not been as thoroughly explored in the field of information retrieval (IR). Therefore, the primary contribution of this work is to bridge the gap between interpretable ML and IR, specifically in the learning to rank (LTR) domain.

We begin by outlining the commonalities between ML and neural LTR, as well as the unique characteristics of ranking systems. For example, in addition to interpretability, efficiency and robustness are highly valued in IR systems. To address this, we have investigated the effectiveness of feature selection methods from interpretable ML in neural LTR, focusing on their impact on interpretability, efficiency, and robustness.

Specifically, we adapt six existing methods from interpretable ML to the neural LTR context and introduce our own G-L2x approach. We examined the distinct characteristics of these methods and their relevance to the LTR task. Subsequently, we conduct extensive experiments to evaluate the trade-offs between ranking performance and sparsity. Our findings indicate that several methods from interpretable ML are highly effective for feature selection in neural LTR. Notably, the local method TabNet can achieve optimal ranking performance with fewer than 10 features. Additionally, we analyzed the efficiency improvements through reductions in feature costs. The global methods, especially G-L2x, can decrease feature retrieval costs by more than 70% while maintaining 96% of the performance compared to a full feature model. Finally, we observe that all global methods, as well as one local method Ifg, demonstrate higher robustness and lower sensibility against perturbations and randomness.

Reflections. This work mainly focuses on bridging the gap between the LTR and interpretable ML fields. The resulting ranking performance is comparable, yet not substantially superior to the ones by decision trees. Thus, future work can take inspiration from tree models and incorporate them with neural feature selection. Furthermore, understanding correlations between features and using features that are not just interpretable, but also fast-to-compute, could support both practitioners and users to achieve transparent and efficient ranking systems.

5.1.3 Faithful Interpretable Models without Leakage

RQ3 Are self-explainable models faithful and how to design theoretically guaranteed faithful models?

This research question is divided into two sub-questions, which in turn lead to two phases of work. The first sub-question addresses a well-established concept in the field of interpretable ML: the extracted explanations are faithful to the model's prediction because the information utilized by the model comes solely from the extracted explanations. We challenge this concept by introducing our own definition of faithfulness: a faithful model should not exhibit feature leakage or label leakage. We provide formal mathematical definitions for both types of leakage. Next, we analyze several widely recognized methods in interpretable ML, and surprisingly, all of these methods are subject to leakage, which can lead to misleading explanations and unreliable performance.

Following this, the subsequent question explores how to design a model that ensures no leakage. Building upon our definition of leakage, we derive the necessary and sufficient conditions to introduce the first methods that guarantee no leakage: a linear programming approach and SUWR. The linear programming method operates with complete information, whereas SUWR optimizes based on existing knowledge and can generalize to unseen scenarios. Therefore, SUWR presents a more realistic solution.

Our extensive results show that SUWR excels in achieving high selection sparsity without compromising predictive accuracy, consistently outperforming all baseline methods across various benchmark datasets. What sets SUWR apart is its ability to provide a step-by-step process that serves as a coherent narrative explanation in itself. Given its generic and adaptable nature, we believe SUWR lays a solid groundwork for future research in

interpretable ML, offering theoretical guarantees for accurate and transparent model predictions.

Reflections. While the guarantee of no leakage is a great advantage over existing methods, our SUWR algorithm could potentially require more computational costs than previous approaches. This could pose a challenge to data with high dimensionality, e.g., if ζ only selects a single feature per step, and thus a high T should be chosen. However, the SUWR framework is highly flexible and can be adapted to handle such situations better. For instance, one can choose ζ_{select} to be a lightweight model that can choose multiple features at once, allowing SUWR approach to scale to high-dimensional data. Additionally, our definitions of leakage might serve as a foundation for novel indicators of feature importance. Exploring the potential of relaxing the no-leakage guarantee of SUWR to better suit natural language domains is also worth considering.

5.2 Ethical and Societal Impacts

Interpretable information retrieval (IR) systems have the potential to highly impact society by enhancing transparency and accountability in information-seeking processes. This thesis aims to enhance transparency and, crucially, foster trust in modern IR systems by providing clear and understandable explanations of how search results are generated by models. Furthermore, this thesis also makes a significant contribution to the field of interpretable machine learning (ML), which is crucial for the development of transparent and hence responsible machine learning systems. We show that our methods are versatile and can be applied to numerous applications, from healthcare to finance. Our research provides the theoretical foundation for further advancements in creating models that are not only effective but also intrinsically transparent and thus promote accountability. In an era where algorithmic decisions have profound impacts on individuals and societies, the methodologies presented in this thesis ensure that these systems can be scrutinized and understood by stakeholders, thereby fostering trust and facilitating the broader adoption of AI technologies in sensitive and impactful domains.

5.3 Broader Discussion and Moving Forward

The research in this thesis was carried out over a brief period, during which machine learning (ML) and information retrieval (IR) have continued to evolve rapidly. Recent advancements in retrieval-augmented generation (RAG) are merging the fields of information retrieval (IR) and large language models (LLMs). It appears IR will only continue to engage closely in modern intelligent systems. Given the constraints of timeline and research scopes, we have identified multiple limitations in achieving interpretable IR. In this section, we will discuss some major limitations in a high level. More importantly, we propose a variety of future directions that can potentially improve the transparency and accountability of intelligent systems.

5.3.1 Engaging Diverse Stakeholders

In this thesis, our main objective has been to clarify complex model predictions for *developers*. This requires a strong grasp of basic machine learning (ML) and information re-

trieval (IR) principles. For other audiences, particularly end users, our explanations may fall short of being fully comprehensible and, as a result, may not effectively enhance users' trust in the systems. Additionally, our research primarily concentrates on the fidelity of explanations and the effectiveness of models, while other important criteria, such as efficiency and fairness, have been overlooked. With the growing interest in developing ML systems across various application domains, ethical considerations have become a major topic of discussion. Information retrieval, as a fundamental application that closely involves both machines and users, must incorporate input from domain experts to address ethical requirements. We believe that the transparency and trustworthiness of comprehensive systems cannot be achieved without involving all stakeholders. Therefore, future work should take into account different audiences and potentially generate tailored explanations for specific audience groups.

Starting from rethinking the **purpose** of explanations, it becomes clear that different stakeholders have varying interests. For instance, system developers may prioritize the model's ability to uncover meaningful correlations within datasets. Therefore, explanations that highlight key features can be valuable for developers, aiding them in debugging models and ensuring that predictions are accurate for the correct reasons. However, there remains a need for explanation methods specifically designed for these sanity checks. Consequently, the **evaluation** metrics for explanations should evolve to accommodate this new role. For example, one metric could assess how effectively the explanations help developers identify undesirable biases. When considering domain experts, explanations can serve as alerts if they do not align with the experts' predefined guidelines. This could include instances where explanations contradict domain knowledge or violate ethical standards. On the other hand, end users, who typically lack technical expertise, may require more accessible explanations, preferably in natural language. In this case, the methods used for generating explanations, the final format of these explanations, and the evaluation metrics will likely differ from those used for developers or domain experts.

5.3.2 The Lack of Real-World Practice

The work we have discussed in this thesis mainly involves popular open-source models rather than comprehensive real-world systems, such as commercial search engines or domain-specific applications. Consequently, it remains uncertain what the actual challenges are or whether the explanation methods can seamlessly apply to the typically more complex real-world systems. Although the datasets used in Chapter 2 and Chapter 3 are real-world search logs, they have been in use for over a decade and may no longer meet the standards of modern information-seeking purposes. Therefore, keeping academic research aligned with real-world practices is undoubtedly beneficial for both fields. This is surely a nontrivial process.

Real-world applications typically prioritize certain aspects. Firstly, can explanations be provided to users in real-time for (ideally) every action? Since **latency** is one of the most crucial factors affecting user satisfaction, balancing the trade-offs between explanation accuracy (or faithfulness) and the time required to generate these explanations is a potential area for future work. Additionally, commercial search engines often interact directly with end users. As previously discussed, explanations for a particular ranking should be understandable to end users. Therefore, exploring the optimal formats and **pre-**

sentation methods for these explanations is also valuable. Lastly, privacy is a big concern for end users in many applications. In the context of search engines, users are often curious about what information about themselves is used for specific promotions. Does the search engine maintain a **user profile** for each individual and generate personalized advertisements based on it? Helping users understand whether such profiles exist and if they are accurate not only builds trust but also supports privacy advocacy.

5.3.3 Interpretability in the Era of Large Language Models

Large language models (LLMs) [13] have received unprecedented attention and success recently due to their great generative capabilities. Such a natural-language level of generative capabilities can be a double-edged sword. On one hand, users seem more engaged in relying on LLMs (e.g., ChatGPT [1]) to solve all kinds of real-world tasks. On the other hand, the generative results do not guarantee to be true (despite seeming *plausible*), which is commonly recognized as **hallucination** [146]. Without robust verification mechanisms including explanations for these outputs, the practical application of LLMs will certainly be constrained.

There are a few empirical efforts in improving the reliability of generated results. For example, retrieval-augmented generation (RAG) [75] seeks to incorporate in-context scenarios during generation, ideally grounding the generated outputs in these retrieved contexts. In this framework, the retrieved contexts serve partly as explanations for the generated answers, and the quality of these contextual resources can significantly impact the quality of subsequent generations. However, unless the retrieval models strictly adhere to traditional information retrieval (IR) factors, RAG does not simplify explaining the generated answers, but rather introduces one more explanation task of the retrieval results.

Another example is chain-of-thought (COT) generation [86]. It relies on manually crafted prompts consisting of detailed actions to take and the corresponding reasons, so that the model will hopefully pick up the reasoning pattern. However, recent work [66] has already shown that actions do not always align with their tailored reasons by COT. Similarly, Brown et al. [13], Dalvi et al. [28] ask the LLMs to output not just the answer, but also the explanations to a question. As much as the explanations may seem plausible, there is no guarantee that the model really understands the question or reasoning. This is not surprising, as we already found out in Chapter 4, that the model can "lie" when the explanation is part of the predicted outputs. With LLMs, it is much harder for humans to verify both answers and explanations, due to the enormous datasets and complexity of models and tasks.

Explaining generative outputs from LLMs is challenging due to multiple intertwined factors, such as the complexity of training data (including human feedback), model parameters, and prediction tasks. Simply applying our method in Chapter 4 is impractical, as it assumes structured data features and independent representations. In contrast, natural language processing has greatly benefited from contextualized pre-training. Therefore, making trade-offs in faithful explanations to maintain the performance seems inevitable. As a starting point, approximating the uncertainties of LLM outputs and their explanations [138], aided by explainable retrieval contexts, could be beneficial.

[1] `https://chatgpt.com`

Bibliography

References

[1] Ashraf M. Abdul, Christian von der Weth, Mohan S. Kankanhalli, and Brian Y. Lim. COGAM: measuring and moderating cognitive load in machine learning model explanations. In Regina Bernhaupt, Florian 'Floyd' Mueller, David Verweij, Josh Andres, Joanna McGrenere, Andy Cockburn, Ignacio Avellino, Alix Goguey, Pernille Bjøn, Shengdong Zhao, Briane Paul Samson, and Rafal Kocielnik, editors, *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*, pages 1–14. ACM, 2020. doi: 10.1145/3313831.3376615. URL https://doi.org/10.1145/3313831.3376615.

[2] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008. doi: 10.1145/1411509.1411513. URL https://doi.org/10.1145/1411509.1411513.

[3] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7786–7795, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3e9f0fc9b2f89e043bc6233994dfcf76-Abstract.html.

[4] Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, October 2002. ISSN 1046-8188. doi: 10.1145/582415.582416. URL http://doi.acm.org/10.1145/582415.582416.

[5] Avishek Anand, Lijun Lyu, Maximilian Idahl, Yumeng Wang, Jonas Wallat, and Zijian Zhang. Explainable information retrieval: A survey. *CoRR*, abs/2211.02405, 2022. doi: 10.48550/arXiv.2211.02405. URL https://doi.org/10.48550/arXiv.2211.02405.

[6] Ioannis Arapakis, Xiao Bai, and Berkant Barla Cambazoglu. Impact of response latency on user behavior in web search. In Shlomo Geva, Andrew Trotman, Peter Bruza, Charles L. A. Clarke, and Kalervo Järvelin, editors, *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 103–112. ACM, 2014. doi: 10.1145/2600428.2609627. URL https://doi.org/10.1145/2600428.2609627.

[7] Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6679–6687. AAAI Press, 2021.

[8] Xiao Bai and Berkant Barla Cambazoglu. Impact of response latency on sponsored search. *Inf. Process. Manag.*, 56(1):110–129, 2019. doi: 10.1016/J.IPM.2018.10.005. URL https://doi.org/10.1016/j.ipm.2018.10.005.

[9] Muhammed Fatih Balin, Abubakar Abid, and James Y. Zou. Concrete autoencoders: Differentiable feature selection and reconstruction. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 444–453. PMLR, 2019. URL http://proceedings.mlr.press/v97/balin19a.html.

[10] Miguel Barreda-Ángeles, Ioannis Arapakis, Xiao Bai, Berkant Barla Cambazoglu, and Alexandre Pereda-Baños. Unconscious physiological effects of search latency on users and their click behaviour. In Ricardo Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto, editors, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 203–212. ACM, 2015. doi: 10.1145/2766462.2767719. URL https://doi.org/10.1145/2766462.2767719.

[11] 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*, pages 2963–2977, 2019.

[12] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[14] Sebastian Bruch. An alternative cross entropy loss for learning-to-rank. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia, editors, *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 118–126. ACM / IW3C2, 2021. doi: 10.1145/3442381.3449794. URL https://doi.org/10.1145/3442381.3449794.

[15] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In Yi Fang, Yi Zhang, James Allan, Krisztian Balog, Ben Carterette, and Jiafeng Guo, editors, *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2019, Santa Clara, CA, USA, October 2-5, 2019*, pages

75–78. ACM, 2019. doi: 10.1145/3341981.3344221. URL https://doi.org/10.1145/3341981.3344221.

[16] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In Luc De Raedt and Stefan Wrobel, editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 89–96. ACM, 2005. doi: 10.1145/1102351.1102363. URL https://doi.org/10.1145/1102351.1102363.

[17] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 193–200. MIT Press, 2006. URL https://proceedings.neurips.cc/paper/2006/hash/af44c4c56f385c43f2529f9b1b018f6a-Abstract.html.

[18] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[19] Arthur Câmara and Claudia Hauff. Diagnosing BERT with retrieval heuristics. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part I*, volume 12035 of *Lecture Notes in Computer Science*, pages 605–618. Springer, 2020. doi: 10.1007/978-3-030-45439-5_40. URL https://doi.org/10.1007/978-3-030-45439-5_40.

[20] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In Zoubin Ghahramani, editor, *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 129–136. ACM, 2007. doi: 10.1145/1273496.1273513. URL https://doi.org/10.1145/1273496.1273513.

[21] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In Olivier Chapelle, Yi Chang, and Tie-Yan Liu, editors, *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010*, volume 14 of *JMLR Proceedings*, pages 1–24. JMLR.org, 2011. URL http://proceedings.mlr.press/v14/chapelle11a.html.

[22] Olivier Chapelle and S. Sathiya Keerthi. Efficient algorithms for ranking with svms. *Inf. Retr.*, 13(3):201–215, 2010. doi: 10.1007/S10791-009-9109-9. URL https://doi.org/10.1007/s10791-009-9109-9.

[23] Howard Chen, Jacqueline He, Karthik Narasimhan, and Danqi Chen. Can rationalization improve robustness? In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

[24] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 882–891. PMLR, 2018. URL http://proceedings.mlr.press/v80/chen18j.html.

[25] Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. Overview of the trec 2009 web track. In *TREC*, 2009.

[26] Ian Connick Covert, Wei Qiu, Mingyu Lu, Na Yoon Kim, Nathan J White, and Su-In Lee. Learning to maximize mutual information for dynamic feature selection. In *International Conference on Machine Learning*, pages 6424–6447. PMLR, 2023.

[27] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. *Advances in neural information processing systems*, 30, 2017.

[28] Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, 2021.

[29] George Dantzig. *Linear programming and extensions*. Princeton university press, 1963.

[30] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. *ACM Trans. Inf. Syst.*, 35(2):15:1–15:31, 2016. doi: 10.1145/2987380. URL https://doi.org/10.1145/2987380.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

[32] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. Eraser: A benchmark to evaluate rationalized nlp models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, 2020.

[33] Steven Diamond and Stephen P. Boyd. CVXPY: A python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.*, 17:83:1–83:5, 2016. URL http://jmlr.org/papers/v17/15-408.html.

[34] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Commun. ACM*, 63(1):68–77, 2020. doi: 10.1145/3359786. URL https://doi.org/10.1145/3359786.

[35] William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3:6, 2019.

[36] Zeon Trevor Fernando, Jaspreet Singh, and Avishek Anand. A study on the interpretability of neural retrieval models using deepshap. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 1005–1008, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6172-9. doi: 10.1145/3331184.3331312. URL http://doi.acm.org/10.1145/3331184.3331312.

[37] Besnik Fetahu, Katja Markert, and Avishek Anand. Automated news suggestions for populating wikipedia entity pages. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 323–332, 2015.

[38] Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In Jude W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 170–178. Morgan Kaufmann, 1998.

[39] Luke Gallagher, Ruey-Cheng Chen, Roi Blanco, and J. Shane Culpepper. Joint optimization of cascade ranking models. In J. Shane Culpepper, Alistair Moffat, Paul N. Bennett, and Kristina Lerman, editors, *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 15–23. ACM, 2019. doi: 10.1145/3289600.3290986. URL https://doi.org/10.1145/3289600.3290986.

[40] Irena Gao, Gabriel Ilharco, Scott Lundberg, and Marco Tulio Ribeiro. Adaptive testing of computer vision models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4003–4014, 2023.

[41] Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. Inducing causal structure for interpretable neural networks. In *International Conference on Machine Learning*, pages 7324–7338. PMLR, 2022.

[42] Xiubo Geng, Tie-Yan Liu, Tao Qin, and Hang Li. Feature selection for ranking. In Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando, editors, *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 407–414. ACM, 2007. doi: 10.1145/1277741.1277811. URL https://doi.org/10.1145/1277741.1277811.

[43] Andrea Gigli, Claudio Lucchese, Franco Maria Nardini, and Raffaele Perego. Fast feature selection for learning to rank. In Ben Carterette, Hui Fang, Mounia Lalmas, and Jian-Yun Nie, editors, *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval, ICTIR 2016, Newark, DE, USA, September 12-6, 2016*, pages 167–170. ACM, 2016. doi: 10.1145/2970398.2970433. URL https://doi.org/10.1145/2970398.2970433.

[44] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi, editors, *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 55–64. ACM, 2016. doi: 10.1145/2983323.2983769. URL https://doi.org/10.1145/2983323.2983769.

[45] Sai Gurrapu, Ajay Kulkarni, Lifu Huang, Ismini Lourentzou, Laura J. Freeman, and Feras A. Batarseh. Rationalization for explainable NLP: A survey. *CoRR*, abs/2301.08912, 2023. doi: 10.48550/ARXIV.2301.08912.

[46] Gaole He, Agathe Balayn, Stefan Buijsman, Jie Yang, and Ujwal Gadiraju. It is like finding a polar bear in the savannah! concept-level ai explanations with analogical inference from commonsense knowledge. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 10, pages 89–101, 2022.

[47] Xiaofei He, Deng Cai, and Partha Niyogi. Laplacian score for feature selection. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 507–514, 2005. URL https://proceedings.neurips.cc/paper/2005/hash/b5b03f06271f8917685d14cea7c6c50a-Abstract.html.

[48] Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. Intra-document cascading: Learning to select passages for neural document ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 1349–1358, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3462889. URL https://doi.org/10.1145/3404835.3462889.

[49] Maximilian Idahl, Lijun Lyu, Ujwal Gadiraju, and Avishek Anand. Towards benchmarking the utility of explanations for model debugging. In *TrustNLP: First Workshop on Trustworthy Natural Language Processing*, page 68, 2021.

[50] Alon Jacovi and Yoav Goldberg. Aligning faithful interpretations with their social attribution. *Transactions of the Association for Computational Linguistics*, 9:294–310, 2021.

[51] Nasreen Abdul Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah S. Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. Umass at TREC 2004: Novelty and HARD. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings of*

the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004, volume 500-261 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2004. URL http://trec.nist.gov/pubs/trec13/papers/umass.novelty.hard.pdf.

[52] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[53] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002. doi: 10.1145/582415.582418. URL http://doi.acm.org/10.1145/582415.582418.

[54] Neil Jethani, Mukund Sudarshan, Yindalon Aphinyanaphongs, and Rajesh Ranganath. Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 1459–1467. PMLR, 2021.

[55] Neil Jethani, Adriel Saporta, and Rajesh Ranganath. Don't be fooled: label leakage in explanation methods and the importance of their quantitative evaluation. In *International Conference on Artificial Intelligence and Statistics*, pages 8925–8953. PMLR, 2023.

[56] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142. ACM, 2002. doi: 10.1145/775047.775067. URL https://doi.org/10.1145/775047.775067.

[57] Thorsten Joachims. Training linear svms in linear time. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 217–226. ACM, 2006. doi: 10.1145/1150402.1150429. URL https://doi.org/10.1145/1150402.1150429.

[58] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://doi.org/10.18653/v1/2020.emnlp-main.550.

[59] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):1–21, 2012.

[60] Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna M. Wallach, and Jennifer Wortman Vaughan. Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. In Regina Bernhaupt, Florian 'Floyd' Mueller, David Verweij, Josh Andres, Joanna McGrenere, Andy Cockburn, Ignacio Avellino, Alix Goguey, Pernille Bjøn, Shengdong Zhao, Briane Paul Samson, and Rafal Kocielnik, editors, *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*, pages 1–14. ACM, 2020. doi: 10.1145/3313831.3376219. URL https://doi.org/10.1145/3313831.3376219.

[61] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3146–3154, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

[62] Fereshte Khani and Marco Tulio Ribeiro. Collaborative development of nlp models. *arXiv preprint arXiv:2305.12219*, 2023.

[63] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

[64] Hanjiang Lai, Yan Pan, Cong Liu, Liang Lin, and Jie Wu. Sparse learning-to-rank via an efficient primal-dual algorithm. *IEEE Trans. Computers*, 62(6):1221–1233, 2013. doi: 10.1109/TC.2012.62. URL https://doi.org/10.1109/TC.2012.62.

[65] Hanjiang Lai, Yan Pan, Yong Tang, and Rong Yu. Fsmrank: Feature selection algorithm for learning to rank. *IEEE Trans. Neural Networks Learn. Syst.*, 24(6):940–952, 2013. doi: 10.1109/TNNLS.2013.2247628. URL https://doi.org/10.1109/TNNLS.2013.2247628.

[66] Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

[67] Léa Laporte, Rémi Flamary, Stéphane Canu, Sébastien Déjean, and Josiane Mothe. Nonconvex regularizations for feature selection in ranking with sparse SVM. *IEEE Trans. Neural Networks Learn. Syst.*, 25(6):1118–1130, 2014. doi: 10.1109/TNNLS.2013.2286696. URL https://doi.org/10.1109/TNNLS.2013.2286696.

[68] Sören Laue, Matthias Mitterreiter, and Joachim Giesen. GENO - generic optimization for classical machine learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural*

Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 2187–2198, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/84438b7aae55a0638073ef798e50b4ef-Abstract.html.

[69] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[70] Changhee Lee, Fergus Imrie, and Mihaela van der Schaar. Self-supervision enhanced feature selection with correlated gates. In *International Conference on Learning Representations*, 2021.

[71] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, 2016.

[72] Ismael Lemhadri, Feng Ruan, Louis Abraham, and Robert Tibshirani. Lassonet: A neural network with feature sparsity. *J. Mach. Learn. Res.*, 22:127:1–127:29, 2021. URL http://jmlr.org/papers/v22/20-848.html.

[73] Jurek Leonhardt, Koustav Rudra, and Avishek Anand. Extractive explanations for interpretable text ranking. *ACM Trans. Inf. Syst.*, dec 2022. ISSN 1046-8188. doi: 10.1145/3576924. URL https://doi.org/10.1145/3576924.

[74] Piyawat Lertvittayakumjorn and Francesca Toni. Explanation-based human debugging of nlp models: A survey. *Transactions of the Association for Computational Linguistics*, 9:1508–1528, 2021.

[75] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[76] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM computing surveys (CSUR)*, 50(6):1–45, 2017.

[77] Ping Li, Christopher J. C. Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 897–904. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper/2007/hash/b86e8d03fe992d1b0e19656875ee557c-Abstract.html.

[78] Yang Li and Junier Oliva. Active feature acquisition with generative surrogate models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6450–6459. PMLR, 2021.

[79] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[80] Michael Llordes, Debasis Ganguly, Sumit Bhatia, and Chirag Agarwal. Explain like i am bm25: Interpreting a dense model's ranked-list with a sparse approximation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1976–1980, 2023.

[81] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Alberto Veneri. ILMART: interpretable ranking with constrained lambdamart. In Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai, editors, *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 2255–2259. ACM, 2022. doi: 10.1145/3477495.3531840. URL https://doi.org/10.1145/3477495.3531840.

[82] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

[83] Lijun Lyu and Avishek Anand. Listwise explanations for ranking models using multiple explainers. In *European Conference on Information Retrieval*, pages 653–668. Springer, 2023.

[84] Lijun Lyu, Maria Koutraki, Martin Krickl, and Besnik Fetahu. Neural ocr post-hoc correction of historical corpora. *Transactions of the Association for Computational Linguistics*, 9:479–493, 2021.

[85] Lijun Lyu, Nirmal Roy, Harrie Oosterhuis, and Avishek Anand. Is interpretable machine learning effective at feature selection for neural learning-to-rank? In *European Conference on Information Retrieval*, pages 384–402. Springer, 2024.

[86] Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, 2023.

[87] André F. T. Martins and Ramón Fernandez Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1614–1623. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/martins16.html.

[88] Aria Masoomi, Chieh Wu, Tingting Zhao, Zifeng Wang, Peter J. Castaldi, and Jennifer G. Dy. Instance-wise feature grouping. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/9b10a919ddeb07e103dc05ff523afe38-Abstract.html.

[89] Ryan T. McDonald, George Brokos, and Ion Androutsopoulos. Deep relevance ranking using enhanced document-query interactions. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1849–1860. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1211. URL https://doi.org/10.18653/v1/d18-1211.

[90] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[91] Shikhar Murty, Christopher D Manning, Scott Lundberg, and Marco Tulio Ribeiro. Fixing model bugs with natural language patches. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11600–11613, 2022.

[92] Rodrigo Frassetto Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019. URL http://arxiv.org/abs/1901.04085.

[93] Harrie Oosterhuis, Lijun Lyu, and Avishek Anand. Local feature selection without label or feature leakage for interpretable machine learning predictions. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 38740–38761. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/oosterhuis24a.html.

[94] Feng Pan, Tim Converse, David Ahn, Franco Salvetti, and Gianluca Donato. Feature selection for ranking using boosted trees. In David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy Lin, editors, *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 2025–2028. ACM, 2009. doi: 10.1145/1645953.1646292. URL https://doi.org/10.1145/1645953.1646292.

[95] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. Setrank: Learning a permutation-invariant ranking model for information retrieval. In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 499–508. ACM, 2020. doi: 10.1145/3397271.3401104. URL https://doi.org/10.1145/3397271.3401104.

[96] Bhargavi Paranjape, Mandar Joshi, John Thickstun, Hannaneh Hajishirzi, and Luke Zettlemoyer. An information bottleneck approach for controlling conciseness in rationale extraction. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1938–1952. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.153. URL https://doi.org/10.18653/v1/2020.emnlp-main.153.

[97] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014. doi: 10.3115/v1/d14-1162. URL https://doi.org/10.3115/v1/d14-1162.

[98] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. Kilt: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, 2021.

[99] Przemyslaw Pobrotyn, Tomasz Bartczak, Mikolaj Synowiec, Radoslaw Bialobrzeski, and Jaroslaw Bojar. Context-aware learning to rank with self-attention. *CoRR*, abs/2005.10084, 2020. URL https://arxiv.org/abs/2005.10084.

[100] Alberto Purpura, Karolina Buchner, Gianmaria Silvello, and Gian Antonio Susto. Neural feature selection for learning to rank. In Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani, editors, *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*, volume 12657 of *Lecture Notes in Computer Science*, pages 342–349. Springer, 2021. doi: 10.1007/978-3-030-72240-1_34. URL https://doi.org/10.1007/978-3-030-72240-1_34.

[101] Tao Qin and Tie-Yan Liu. Introducing LETOR 4.0 datasets. *CoRR*, abs/1306.2597, 2013. URL http://arxiv.org/abs/1306.2597.

[102] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retr.*, 13(4):346–374, 2010. doi: 10.1007/S10791-009-9123-Y. URL https://doi.org/10.1007/s10791-009-9123-y.

[103] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. Are neural rankers still outperformed by gradient boosted decision trees? In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Ut1vF_q_vC.

[104] Ashwini Rahangdale and Shital A. Raut. Deep neural network regularization for feature selection in learning-to-rank. *IEEE Access*, 7:53988–54006, 2019. doi: 10.1109/ACCESS.2019.2902640. URL https://doi.org/10.1109/ACCESS.2019.2902640.

[105] Daniël Rennings, Felipe Moraes, and Claudia Hauff. An axiomatic approach to diagnosing neural IR models. In Leif Azzopardi, Benno Stein, Norbert Fuhr, Philipp Mayr, Claudia Hauff, and Djoerd Hiemstra, editors, *Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part I*, volume 11437 of *Lecture Notes in Computer Science*, pages 489–503. Springer, 2019. doi: 10.1007/978-3-030-15712-8_32. URL https://doi.org/10.1007/978-3-030-15712-8_32.

[106] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM, 2016. doi: 10.1145/2939672.2939778. URL https://doi.org/10.1145/2939672.2939778.

[107] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[108] Leonardo Rigutini, Tiziano Papini, Marco Maggini, and Franco Scarselli. Sortnet: Learning to rank by a neural-based sorting algorithm. *CoRR*, abs/2311.01864, 2023. doi: 10.48550/ARXIV.2311.01864. URL https://doi.org/10.48550/arXiv.2311.01864.

[109] Yao Rong, Tobias Leemann, Thai-trang Nguyen, Lisa Fiedler, Tina Seidel, Gjergji Kasneci, and Enkelejda Kasneci. Towards human-centered explainable AI: user studies for model explanations. *CoRR*, abs/2210.11584, 2022. doi: 10.48550/arXiv.2210.11584. URL https://doi.org/10.48550/arXiv.2210.11584.

[110] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5): 206–215, 2019. doi: 10.1038/s42256-019-0048-x. URL https://doi.org/10.1038/s42256-019-0048-x.

[111] Patrick Schwab and Walter Karlen. Cxplain: Causal explanations for model interpretation under uncertainty. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 10220–10230, 2019.

[112] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Doina Precup and Yee Whye

Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 2017. URL http://proceedings.mlr.press/v70/shrikumar17a.html.

[113] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6034.

[114] Jaspreet Singh and Avishek Anand. Posthoc interpretability of learning to rank models using secondary training data. In *Workshop on ExplainAble Recommendation and Search (EARS 2018) at SIGIR 2018*, 2018. URL https://ears2018.github.io/ears18-singh.pdf.

[115] Jaspreet Singh and Avishek Anand. Exs: Explainable search using local model agnostic interpretability. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 770–773, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-5940-5. doi: 10.1145/3289600.3290620. URL http://doi.acm.org/10.1145/3289600.3290620.

[116] Jaspreet Singh and Avishek Anand. Model agnostic interpretability of rankers via intent modelling. In Mireille Hildebrandt, Carlos Castillo, L. Elisa Celis, Salvatore Ruggieri, Linnet Taylor, and Gabriela Zanfir-Fortuna, editors, *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pages 618–628. ACM, 2020. doi: 10.1145/3351095.3375234. URL https://doi.org/10.1145/3351095.3375234.

[117] Jaspreet Singh, Jonas Wallat, and Avishek Anand. Bertnesia: Investigating the capture and forgetting of knowledge in BERT. In Afra Alishahi, Yonatan Belinkov, Grzegorz Chrupala, Dieuwke Hupkes, Yuval Pinter, and Hassan Sajjad, editors, *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2020, Online, November 2020*, pages 174–183. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.blackboxnlp-1.17. URL https://doi.org/10.18653/v1/2020.blackboxnlp-1.17.

[118] Jaspreet Singh, Megha Khosla, Zhenye Wang, and Avishek Anand. Extracting per query valid explanations for blackbox learning-to-rank models. In Faegheh Hasibi, Yi Fang, and Akiko Aizawa, editors, *ICTIR '21: The 2021 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Canada, July 11, 2021*, pages 203–210. ACM, 2021. doi: 10.1145/3471158.3472241. URL https://doi.org/10.1145/3471158.3472241.

[119] Zhengya Sun, Tao Qin, Qing Tao, and Jue Wang. Robust sparse rank learning for non-smooth ranking measures. In James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel, editors, *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,*

SIGIR 2009, Boston, MA, USA, July 19-23, 2009, pages 259–266. ACM, 2009. doi: 10.1145/1571941.1571987. URL https://doi.org/10.1145/1571941.1571987.

[120] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017. URL http://proceedings.mlr.press/v70/sundararajan17a.html.

[121] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[122] Michael J. Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In Marc Najork, Andrei Z. Broder, and Soumen Chakrabarti, editors, *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008, Palo Alto, California, USA, February 11-12, 2008*, pages 77–86. ACM, 2008. doi: 10.1145/1341531.1341544. URL https://doi.org/10.1145/1341531.1341544.

[123] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[124] Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun Preece. Sanity checks for saliency metrics. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6021–6029, 2020.

[125] Robert J Vanderbei et al. *Linear programming*. Springer, 2020.

[126] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[127] Manisha Verma and Debasis Ganguly. LIRME: locally interpretable ranking model explanation. In Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer, editors, *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1281–1284. ACM, 2019. doi: 10.1145/3331184.3331377. URL https://doi.org/10.1145/3331184.3331377.

[128] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman,

Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[129] Michael Völske, Alexander Bondarenko, Maik Fröbe, Benno Stein, Jaspreet Singh, Matthias Hagen, and Avishek Anand. Towards axiomatic explanations for neural ranking models. In Faegheh Hasibi, Yi Fang, and Akiko Aizawa, editors, *ICTIR '21: The 2021 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Canada, July 11, 2021*, pages 13–22. ACM, 2021. doi: 10.1145/3471158.3472256. URL https://doi.org/10.1145/3471158.3472256.

[130] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, 2019.

[131] Jonas Wallat, Fabian Beringer, Abhijit Anand, and Avishek Anand. Probing bert for ranking abilities. In *Advances in Information Retrieval - 45th European Conference on IR Research, ECIR 2023, Dublin, Ireland*, Lecture Notes in Computer Science. Springer, 2023.

[132] Lidan Wang, Jimmy Lin, and Donald Metzler. A cascade ranking model for efficient ranked retrieval. In Wei-Ying Ma, Jian-Yun Nie, Ricardo Baeza-Yates, Tat-Seng Chua, and W. Bruce Croft, editors, *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 105–114. ACM, 2011. doi: 10.1145/2009916.2009934. URL https://doi.org/10.1145/2009916.2009934.

[133] Yumeng Wang, Lijun Lyu, and Avishek Anand. BERT rankers are brittle: A study using adversarial document perturbations. In Fabio Crestani, Gabriella Pasi, and Éric Gaussier, editors, *ICTIR '22: The 2022 ACM SIGIR International Conference on the Theory of Information Retrieval, Madrid, Spain, July 11 - 12, 2022*, pages 115–120. ACM, 2022. doi: 10.1145/3539813.3545122. URL https://doi.org/10.1145/3539813.3545122.

[134] Qiang Wu, Christopher J. C. Burges, Krysta M. Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Inf. Retr.*, 13(3):254–270, 2010. doi: 10.1007/S10791-009-9112-1. URL https://doi.org/10.1007/s10791-009-9112-1.

[135] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 1192–1199. ACM, 2008. doi: 10.1145/1390156.1390306. URL https://doi.org/10.1145/1390156.1390306.

[136] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[137] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando, editors, *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 391–398. ACM, 2007. doi: 10.1145/1277741.1277809. URL https://doi.org/10.1145/1277741.1277809.

[138] Tianyang Xu, Shujin Wu, Shizhe Diao, Xiaoze Liu, Xingyao Wang, Yangyi Chen, and Jing Gao. Sayself: Teaching llms to express confidence with self-reflective rationales. *arXiv preprint arXiv:2405.20974*, 2024.

[139] Zhixiang Eddie Xu, Gao Huang, Kilian Q. Weinberger, and Alice X. Zheng. Gradient boosted feature selection. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 522–531. ACM, 2014. doi: 10.1145/2623330.2623635. URL https://doi.org/10.1145/2623330.2623635.

[140] Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *International conference on machine learning*, pages 10648–10659. PMLR, 2020.

[141] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. INVASE: instance-wise variable selection using neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=BJg_roAcK7.

[142] Puxuan Yu, Razieh Rahimi, and James Allan. Towards explainable search results: A listwise explanation generator. In Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai, editors, *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 669–680. ACM, 2022. doi: 10.1145/3477495.3532067. URL https://doi.org/10.1145/3477495.3532067.

[143] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando, editors, *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 271–278. ACM, 2007. doi: 10.1145/1277741.1277790. URL https://doi.org/10.1145/1277741.1277790.

[144] Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, and Xueqi Cheng. Query understanding via intent description generation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1823–1832, 2020.

[145] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.

[146] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.

[147] Zijian Zhang, Koustav Rudra, and Avishek Anand. Explain and predict, and then predict again. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 418–426. ACM, 2021. doi: 10.1145/3437963.3441758. URL https://doi.org/10.1145/3437963.3441758.

[148] Zijian Zhang, Vinay Setty, and Avishek Anand. Sparcassist: A model risk assessment assistant based on sparse generated counterfactuals. In Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai, editors, *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 3219–3223. ACM, 2022. doi: 10.1145/3477495.3531677. URL https://doi.org/10.1145/3477495.3531677.

[149] Yiming Zheng, Serena Booth, Julie Shah, and Yilun Zhou. The irrationality of neural rationale models. In *Proceedings of the 2nd Workshop on Trustworthy Natural Language Processing (TrustNLP 2022)*, pages 64–73, Seattle, U.S.A., July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.trustnlp-1.6.

[150] Honglei Zhuang, Xuanhui Wang, Michael Bendersky, Alexander Grushetsky, Yonghui Wu, Petr Mitrichev, Ethan Sterling, Nathan Bell, Walker Ravina, and Hai Qian. Interpretable ranking with generalized additive models. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 499–507. ACM, 2021. doi: 10.1145/3437963.3441796. URL https://doi.org/10.1145/3437963.3441796.

Curriculum Vitæ

Lijun Lyu

Experience

Aug/2024–Dec/2024	Applied scientist intern at Amazon	Edinburgh, UK
Nov/2022–Jan/2025	PhD candidate at TU Delft	Delft, NL
Apr/2021–Oct/2022	PhD candidate at LUH	Hannover, DE
May/2018–Mar/2021	Research assistant at L3S research center	Hannover, DE

Education

Apr/2015–Apr/2018	M.Sc. in Computer Science, LUH	Hannover, DE
Sep/2010–Jun/2014	B.Sc. in Computer Science, JNU	Guangzhou, CN

Publications

1. *Lijun Lyu, Maria Koutraki, Martin Krickl, Besnik Fetahu*: Neural OCR post-hoc correction of historical corpora. Published at the Transactions of the Association for Computational Linguistics, 479-493, 2021 [84].

2. *Maximilian Idahl, Lijun Lyu, Ujwal Gadiraju, Avishek Anand*: Towards benchmarking the utility of explanations for model debugging. Published at the Proceedings of the 1st Workshop on Trustworthy Natural Language Processing, 68-73, 2021 [49].

3. *Yumeng wang, Lijun Lyu and Avishek Anand*: BERT rankers are brittle: a study using adversarial document perturbations. Published at Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval, 115-120, 2022 [133].

4 *Lijun Lyu and Avishek Anand*: Listwise explanations for ranking models using multiple explainers. Published at the 45th European Conference on Information Retrieval. Springer, 653–668, 2023 [83].

5 *Lijun Lyu, Nirmal Roy, Harrie Oosterhuis, Avishek Anand*: Is Interpretable Machine Learning Effective at Feature Selection for Neural Learning-to-Rank? Published at the 46th European Conference on Information Retrieval. Springer, 384-402, 2024 [85].

6 *Harrie Oosterhuis*, Lijun Lyu* and Avishek Anand*: Local Feature Selection without Label or Feature Leakage for Interpretable Machine Learning Predictions. *equal contribution. Published at the 41st International Conference on Machine Learning, PMLR 235, 38740-38761, 2024 [93].

7 *Avishek Anand, Lijun Lyu, Maximilian Idahl, Yumeng Wang, Jonas Wallat and Zijian Zhang*: Explainable Information Retrieval: A Survey. Accepted at Computing Surveys, 2024 [5].

Included in this thesis.

SIKS Dissertation Series

Since 1998, all dissertations written by PhD. students who have conducted their research under the auspices of a senior research fellow of the SIKS research school are published in the SIKS Dissertation Series.

2016
01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow
03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support
04 Laurens Rietveld (VUA), Publishing and Consuming Linked Data
05 Evgeny Sherkhonov (UvA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers
06 Michel Wilson (TUD), Robust scheduling in an uncertain environment
07 Jeroen de Man (VUA), Measuring and modeling negative emotions for virtual training
08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data
09 Archana Nottamkandath (VUA), Trusting Crowdsourced Information on Cultural Artefacts
10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms
11 Anne Schuth (UvA), Search Engines that Learn from Their Users
12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems
13 Nana Baah Gyan (VUA), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization
15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
16 Guangliang Li (UvA), Socially Intelligent Autonomous Agents that Learn from Human Reward
17 Berend Weel (VUA), Towards Embodied Evolution of Robot Organisms
18 Albert Meroño Peñuela (VUA), Refining Statistical Data on the Web
19 Julia Efremova (TU/e), Mining Social Structures from Genealogical Data
20 Daan Odijk (UvA), Context & Semantics in News & Web Search
21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
22 Grace Lewis (VUA), Software Architecture Strategies for Cyber-Foraging Systems

23 Fei Cai (UvA), Query Auto Completion in Information Retrieval
24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
26 Dilhan Thilakarathne (VUA), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media
28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
30 Ruud Mattheij (TiU), The Eyes Have It
31 Mohammad Khelghati (UT), Deep web content monitoring
32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
33 Peter Bloem (UvA), Single Sample Statistics, exercises in learning from just one example
34 Dennis Schunselaar (TU/e), Configurable Process Trees: Elicitation, Analysis, and Enactment
35 Zhaochun Ren (UvA), Monitoring Social Media: Summarization, Classification and Recommendation
36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies
37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry
38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design
39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect
40 Christian Detweiler (TUD), Accounting for Values in Design
41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance
42 Spyros Martzoukos (UvA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora
43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice
44 Thibault Sellam (UvA), Automatic Assistants for Database Exploration
45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
46 Jorge Gallego Perez (UT), Robots to Make you Happy
47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks
48 Tanja Buttler (TUD), Collecting Lessons Learned

49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis
50 Yan Wang (TiU), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

2017 01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime
02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation
03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines
04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
05 Mahdieh Shadi (UvA), Collaboration Behavior
06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
08 Rob Konijn (VUA), Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
10 Robby van Delden (UT), (Steering) Interactive Play Behavior
11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
12 Sander Leemans (TU/e), Robust Process Mining with Guarantees
13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
14 Shoshannah Tekofsky (TiU), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
15 Peter Berck (RUN), Memory-Based Text Correction
16 Aleksandr Chuklin (UvA), Understanding and Modeling Users of Modern Search Engines
17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
18 Ridho Reinanda (UvA), Entity Associations for Search
19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
22 Sara Magliacane (VUA), Logics for causal inference under uncertainty
23 David Graus (UvA), Entities of Interest — Discovery in Digital Traces
24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
25 Veruska Zamborlini (VUA), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors

28 John Klein (VUA), Architecture Practices for Complex Contexts
29 Adel Alhuraibi (TiU), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT"
30 Wilma Latuny (TiU), The Power of Facial Expressions
31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics
35 Martine de Vos (VUA), Interpreting natural science spreadsheets
36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
37 Alejandro Montes Garcia (TU/e), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
38 Alex Kayal (TUD), Normative Social Applications
39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle
42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval
44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement
46 Jan Schneider (OU), Sensor-based Learning Support
47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration
48 Angel Suarez (OU), Collaborative inquiry-based learning

2018 01 Han van der Aa (VUA), Comparing and Aligning Process Representations
02 Felix Mannhardt (TU/e), Multi-perspective Process Mining
03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction
04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks
05 Hugo Huurdeman (UvA), Supporting the Complex Dynamics of the Information Seeking Process
06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems
07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems
08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems

09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations
10 Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology
11 Mahdi Sargolzaei (UvA), Enabling Framework for Service-oriented Collaborative Networks
12 Xixi Lu (TU/e), Using behavioral context in process mining
13 Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
14 Bart Joosten (TiU), Detecting Social Signals with Spatiotemporal Gabor Filters
15 Naser Davarzani (UM), Biomarker discovery in heart failure
16 Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
17 Jianpeng Zhang (TU/e), On Graph Sample Clustering
18 Henriette Nakad (UL), De Notaris en Private Rechtspraak
19 Minh Duc Pham (VUA), Emergent relational schemas for RDF
20 Manxia Liu (RUN), Time and Bayesian Networks
21 Aad Slootmaker (OU), EMERGO: a generic platform for authoring and playing scenario-based serious games
22 Eric Fernandes de Mello Araújo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
23 Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
24 Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
25 Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
26 Roelof Anne Jelle de Vries (UT),Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology
27 Maikel Leemans (TU/e), Hierarchical Process Mining for Scalable Software Analysis
28 Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
29 Yu Gu (TiU), Emotion Recognition from Mandarin Speech
30 Wouter Beek (VUA), The "K" in "semantic web" stands for "knowledge": scaling semantics to the web

2019 01 Rob van Eijk (UL),Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification
02 Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
03 Eduardo Gonzalez Lopez de Murillas (TU/e), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
04 Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
05 Sebastiaan van Zelst (TU/e), Process Mining with Streaming Data
06 Chris Dijkshoorn (VUA), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
07 Soude Fazeli (TUD), Recommender Systems in Social Learning Platforms
08 Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes

09 Fahimeh Alizadeh Moghaddam (UvA), Self-adaptation for energy efficiency in software systems
10 Qing Chuan Ye (EUR), Multi-objective Optimization Methods for Allocation and Prediction
11 Yue Zhao (TUD), Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs
12 Jacqueline Heinerman (VUA), Better Together
13 Guanliang Chen (TUD), MOOC Analytics: Learner Modeling and Content Generation
14 Daniel Davis (TUD), Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses
15 Erwin Walraven (TUD), Planning under Uncertainty in Constrained and Partially Observable Environments
16 Guangming Li (TU/e), Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models
17 Ali Hurriyetoglu (RUN),Extracting actionable information from microtexts
18 Gerard Wagenaar (UU), Artefacts in Agile Team Communication
19 Vincent Koeman (TUD), Tools for Developing Cognitive Agents
20 Chide Groenouwe (UU), Fostering technically augmented human collective intelligence
21 Cong Liu (TU/e), Software Data Analytics: Architectural Model Discovery and Design Pattern Detection
22 Martin van den Berg (VUA),Improving IT Decisions with Enterprise Architecture
23 Qin Liu (TUD), Intelligent Control Systems: Learning, Interpreting, Verification
24 Anca Dumitrache (VUA), Truth in Disagreement - Crowdsourcing Labeled Data for Natural Language Processing
25 Emiel van Miltenburg (VUA), Pragmatic factors in (automatic) image description
26 Prince Singh (UT), An Integration Platform for Synchromodal Transport
27 Alessandra Antonaci (OU), The Gamification Design Process applied to (Massive) Open Online Courses
28 Esther Kuindersma (UL), Cleared for take-off: Game-based learning to prepare airline pilots for critical situations
29 Daniel Formolo (VUA), Using virtual agents for simulation and training of social skills in safety-critical circumstances
30 Vahid Yazdanpanah (UT), Multiagent Industrial Symbiosis Systems
31 Milan Jelisavcic (VUA), Alive and Kicking: Baby Steps in Robotics
32 Chiara Sironi (UM), Monte-Carlo Tree Search for Artificial General Intelligence in Games
33 Anil Yaman (TU/e), Evolution of Biologically Inspired Learning in Artificial Neural Networks
34 Negar Ahmadi (TU/e), EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES

35 Lisa Facey-Shaw (OU), Gamification with digital badges in learning programming
36 Kevin Ackermans (OU), Designing Video-Enhanced Rubrics to Master Complex Skills
37 Jian Fang (TUD), Database Acceleration on FPGAs
38 Akos Kadar (OU), Learning visually grounded and multilingual representations

2020 01 Armon Toubman (UL), Calculated Moves: Generating Air Combat Behaviour
02 Marcos de Paula Bueno (UL), Unraveling Temporal Processes using Probabilistic Graphical Models
03 Mostafa Deghani (UvA), Learning with Imperfect Supervision for Language Understanding
04 Maarten van Gompel (RUN), Context as Linguistic Bridges
05 Yulong Pei (TU/e), On local and global structure mining
06 Preethu Rose Anish (UT), Stimulation Architectural Thinking during Requirements Elicitation - An Approach and Tool Support
07 Wim van der Vegt (OU), Towards a software architecture for reusable game components
08 Ali Mirsoleimani (UL),Structured Parallel Programming for Monte Carlo Tree Search
09 Myriam Traub (UU), Measuring Tool Bias and Improving Data Quality for Digital Humanities Research
10 Alifah Syamsiyah (TU/e), In-database Preprocessing for Process Mining
11 Sepideh Mesbah (TUD), Semantic-Enhanced Training Data AugmentationMethods for Long-Tail Entity Recognition Models
12 Ward van Breda (VUA), Predictive Modeling in E-Mental Health: Exploring Applicability in Personalised Depression Treatment
13 Marco Virgolin (CWI), Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming
14 Mark Raasveldt (CWI/UL), Integrating Analytics with Relational Databases
15 Konstantinos Georgiadis (OU), Smart CAT: Machine Learning for Configurable Assessments in Serious Games
16 Ilona Wilmont (RUN), Cognitive Aspects of Conceptual Modelling
17 Daniele Di Mitri (OU), The Multimodal Tutor: Adaptive Feedback from Multimodal Experiences
18 Georgios Methenitis (TUD), Agent Interactions & Mechanisms in Markets with Uncertainties: Electricity Markets in Renewable Energy Systems
19 Guido van Capelleveen (UT), Industrial Symbiosis Recommender Systems
20 Albert Hankel (VUA), Embedding Green ICT Maturity in Organisations
21 Karine da Silva Miras de Araujo (VUA), Where is the robot?: Life as it could be
22 Maryam Masoud Khamis (RUN), Understanding complex systems implementation through a modeling approach: the case of e-government in Zanzibar
23 Rianne Conijn (UT), The Keys to Writing: A writing analytics approach to studying writing processes using keystroke logging
24 Lenin da Nóbrega Medeiros (VUA/RUN), How are you feeling, human? Towards emotionally supportive chatbots

25 Xin Du (TU/e), The Uncertainty in Exceptional Model Mining
26 Krzysztof Leszek Sadowski (UU), GAMBIT: Genetic Algorithm for Model-Based mixed-Integer opTimization
27 Ekaterina Muravyeva (TUD), Personal data and informed consent in an educational context
28 Bibeg Limbu (TUD), Multimodal interaction for deliberate practice: Training complex skills with augmented reality
29 Ioan Gabriel Bucur (RUN), Being Bayesian about Causal Inference
30 Bob Zadok Blok (UL), Creatief, Creatiever, Creatiefst
31 Gongjin Lan (VUA), Learning better – From Baby to Better
32 Jason Rhuggenaath (TU/e), Revenue management in online markets: pricing and online advertising
33 Rick Gilsing (TU/e), Supporting service-dominant business model evaluation in the context of business model innovation
34 Anna Bon (UM), Intervention or Collaboration? Redesigning Information and Communication Technologies for Development
35 Siamak Farshidi (UU), Multi-Criteria Decision-Making in Software Production

2021 01 Francisco Xavier Dos Santos Fonseca (TUD),Location-based Games for Social Interaction in Public Space
02 Rijk Mercuur (TUD), Simulating Human Routines: Integrating Social Practice Theory in Agent-Based Models
03 Seyyed Hadi Hashemi (UvA), Modeling Users Interacting with Smart Devices
04 Ioana Jivet (OU), The Dashboard That Loved Me: Designing adaptive learning analytics for self-regulated learning
05 Davide Dell'Anna (UU), Data-Driven Supervision of Autonomous Systems
06 Daniel Davison (UT), "Hey robot, what do you think?" How children learn with a social robot
07 Armel Lefebvre (UU), Research data management for open science
08 Nardie Fanchamps (OU), The Influence of Sense-Reason-Act Programming on Computational Thinking
09 Cristina Zaga (UT), The Design of Robothings. Non-Anthropomorphic and Non-Verbal Robots to Promote Children's Collaboration Through Play
10 Quinten Meertens (UvA), Misclassification Bias in Statistical Learning
11 Anne van Rossum (UL), Nonparametric Bayesian Methods in Robotic Vision
12 Lei Pi (UL), External Knowledge Absorption in Chinese SMEs
13 Bob R. Schadenberg (UT), Robots for Autistic Children: Understanding and Facilitating Predictability for Engagement in Learning
14 Negin Samaeemofrad (UL), Business Incubators: The Impact of Their Support
15 Onat Ege Adali (TU/e), Transformation of Value Propositions into Resource Re-Configurations through the Business Services Paradigm
16 Esam A. H. Ghaleb (UM), Bimodal emotion recognition from audio-visual cues
17 Dario Dotti (UM), Human Behavior Understanding from motion and bodily cues using deep neural networks

18 Remi Wieten (UU), Bridging the Gap Between Informal Sense-Making Tools and Formal Systems - Facilitating the Construction of Bayesian Networks and Argumentation Frameworks
19 Roberto Verdecchia (VUA), Architectural Technical Debt: Identification and Management
20 Masoud Mansoury (TU/e), Understanding and Mitigating Multi-Sided Exposure Bias in Recommender Systems
21 Pedro Thiago Timbó Holanda (CWI), Progressive Indexes
22 Sihang Qiu (TUD), Conversational Crowdsourcing
23 Hugo Manuel Proença (UL), Robust rules for prediction and description
24 Kaijie Zhu (TU/e), On Efficient Temporal Subgraph Query Processing
25 Eoin Martino Grua (VUA), The Future of E-Health is Mobile: Combining AI and Self-Adaptation to Create Adaptive E-Health Mobile Applications
26 Benno Kruit (CWI/VUA), Reading the Grid: Extending Knowledge Bases from Human-readable Tables
27 Jelte van Waterschoot (UT), Personalized and Personal Conversations: Designing Agents Who Want to Connect With You
28 Christoph Selig (UL), Understanding the Heterogeneity of Corporate Entrepreneurship Programs

2022 01 Judith van Stegeren (UT), Flavor text generation for role-playing video games
02 Paulo da Costa (TU/e), Data-driven Prognostics and Logistics Optimisation: A Deep Learning Journey
03 Ali el Hassouni (VUA), A Model A Day Keeps The Doctor Away: Reinforcement Learning For Personalized Healthcare
04 Ünal Aksu (UU), A Cross-Organizational Process Mining Framework
05 Shiwei Liu (TU/e), Sparse Neural Network Training with In-Time Over-Parameterization
06 Reza Refaei Afshar (TU/e), Machine Learning for Ad Publishers in Real Time Bidding
07 Sambit Praharaj (OU), Measuring the Unmeasurable? Towards Automatic Co-located Collaboration Analytics
08 Maikel L. van Eck (TU/e), Process Mining for Smart Product Design
09 Oana Andreea Inel (VUA), Understanding Events: A Diversity-driven Human-Machine Approach
10 Felipe Moraes Gomes (TUD), Examining the Effectiveness of Collaborative Search Engines
11 Mirjam de Haas (UT), Staying engaged in child-robot interaction, a quantitative approach to studying preschoolers' engagement with robots and tasks during second-language tutoring
12 Guanyi Chen (UU), Computational Generation of Chinese Noun Phrases
13 Xander Wilcke (VUA), Machine Learning on Multimodal Knowledge Graphs: Opportunities, Challenges, and Methods for Learning on Real-World Heterogeneous and Spatially-Oriented Knowledge
14 Michiel Overeem (UU), Evolution of Low-Code Platforms

15 Jelmer Jan Koorn (UU), Work in Process: Unearthing Meaning using Process Mining
16 Pieter Gijsbers (TU/e), Systems for AutoML Research
17 Laura van der Lubbe (VUA), Empowering vulnerable people with serious games and gamification
18 Paris Mavromoustakos Blom (TiU), Player Affect Modelling and Video Game Personalisation
19 Bilge Yigit Ozkan (UU), Cybersecurity Maturity Assessment and Standardisation
20 Fakhra Jabeen (VUA), Dark Side of the Digital Media - Computational Analysis of Negative Human Behaviors on Social Media
21 Seethu Mariyam Christopher (UM), Intelligent Toys for Physical and Cognitive Assessments
22 Alexandra Sierra Rativa (TiU), Virtual Character Design and its potential to foster Empathy, Immersion, and Collaboration Skills in Video Games and Virtual Reality Simulations
23 Ilir Kola (TUD), Enabling Social Situation Awareness in Support Agents
24 Samaneh Heidari (UU), Agents with Social Norms and Values - A framework for agent based social simulations with social norms and personal values
25 Anna L.D. Latour (UL), Optimal decision-making under constraints and uncertainty
26 Anne Dirkson (UL), Knowledge Discovery from Patient Forums: Gaining novel medical insights from patient experiences
27 Christos Athanasiadis (UM), Emotion-aware cross-modal domain adaptation in video sequences
28 Onuralp Ulusoy (UU), Privacy in Collaborative Systems
29 Jan Kolkmeier (UT), From Head Transform to Mind Transplant: Social Interactions in Mixed Reality
30 Dean De Leo (CWI), Analysis of Dynamic Graphs on Sparse Arrays
31 Konstantinos Traganos (TU/e), Tackling Complexity in Smart Manufacturing with Advanced Manufacturing Process Management
32 Cezara Pastrav (UU), Social simulation for socio-ecological systems
33 Brinn Hekkelman (CWI/TUD), Fair Mechanisms for Smart Grid Congestion Management
34 Nimat Ullah (VUA), Mind Your Behaviour: Computational Modelling of Emotion & Desire Regulation for Behaviour Change
35 Mike E.U. Ligthart (VUA), Shaping the Child-Robot Relationship: Interaction Design Patterns for a Sustainable Interaction

2023 01 Bojan Simoski (VUA), Untangling the Puzzle of Digital Health Interventions
02 Mariana Rachel Dias da Silva (TiU), Grounded or in flight? What our bodies can tell us about the whereabouts of our thoughts
03 Shabnam Najafian (TUD), User Modeling for Privacy-preserving Explanations in Group Recommendations
04 Gineke Wiggers (UL), The Relevance of Impact: bibliometric-enhanced legal information retrieval

05 Anton Bouter (CWI), Optimal Mixing Evolutionary Algorithms for Large-Scale Real-Valued Optimization, Including Real-World Medical Applications
06 António Pereira Barata (UL), Reliable and Fair Machine Learning for Risk Assessment
07 Tianjin Huang (TU/e), The Roles of Adversarial Examples on Trustworthiness of Deep Learning
08 Lu Yin (TU/e), Knowledge Elicitation using Psychometric Learning
09 Xu Wang (VUA), Scientific Dataset Recommendation with Semantic Techniques
10 Dennis J.N.J. Soemers (UM), Learning State-Action Features for General Game Playing
11 Fawad Taj (VUA), Towards Motivating Machines: Computational Modeling of the Mechanism of Actions for Effective Digital Health Behavior Change Applications
12 Tessel Bogaard (VUA), Using Metadata to Understand Search Behavior in Digital Libraries
13 Injy Sarhan (UU), Open Information Extraction for Knowledge Representation
14 Selma Čaušević (TUD), Energy resilience through self-organization
15 Alvaro Henrique Chaim Correia (TU/e), Insights on Learning Tractable Probabilistic Graphical Models
16 Peter Blomsma (TiU), Building Embodied Conversational Agents: Observations on human nonverbal behaviour as a resource for the development of artificial characters
17 Meike Nauta (UT), Explainable AI and Interpretable Computer Vision – From Oversight to Insight
18 Gustavo Penha (TUD), Designing and Diagnosing Models for Conversational Search and Recommendation
19 George Aalbers (TiU), Digital Traces of the Mind: Using Smartphones to Capture Signals of Well-Being in Individuals
20 Arkadiy Dushatskiy (TUD), Expensive Optimization with Model-Based Evolutionary Algorithms applied to Medical Image Segmentation using Deep Learning
21 Gerrit Jan de Bruin (UL), Network Analysis Methods for Smart Inspection in the Transport Domain
22 Alireza Shojaifar (UU), Volitional Cybersecurity
23 Theo Theunissen (UU), Documentation in Continuous Software Development
24 Agathe Balayn (TUD), Practices Towards Hazardous Failure Diagnosis in Machine Learning
25 Jurian Baas (UU), Entity Resolution on Historical Knowledge Graphs
26 Loek Tonnaer (TU/e), Linearly Symmetry-Based Disentangled Representations and their Out-of-Distribution Behaviour
27 Ghada Sokar (TU/e), Learning Continually Under Changing Data Distributions
28 Floris den Hengst (VUA), Learning to Behave: Reinforcement Learning in Human Contexts
29 Tim Draws (TUD), Understanding Viewpoint Biases in Web Search Results

2024
01 Daphne Miedema (TU/e), On Learning SQL: Disentangling concepts in data systems education
02 Emile van Krieken (VUA), Optimisation in Neurosymbolic Learning Systems
03 Feri Wijayanto (RUN), Automated Model Selection for Rasch and Mediation Analysis
04 Mike Huisman (UL), Understanding Deep Meta-Learning
05 Yiyong Gou (UM), Aerial Robotic Operations: Multi-environment Cooperative Inspection & Construction Crack Autonomous Repair
06 Azqa Nadeem (TUD), Understanding Adversary Behavior via XAI: Leveraging Sequence Clustering to Extract Threat Intelligence
07 Parisa Shayan (TiU), Modeling User Behavior in Learning Management Systems
08 Xin Zhou (UvA), From Empowering to Motivating: Enhancing Policy Enforcement through Process Design and Incentive Implementation
09 Giso Dal (UT), Probabilistic Inference Using Partitioned Bayesian Networks
10 Cristina-Iulia Bucur (VUA), Linkflows: Towards Genuine Semantic Publishing in Science
11 withdrawn
12 Peide Zhu (TUD), Towards Robust Automatic Question Generation For Learning
13 Enrico Liscio (TUD), Context-Specific Value Inference via Hybrid Intelligence
14 Larissa Capobianco Shimomura (TU/e), On Graph Generating Dependencies and their Applications in Data Profiling
15 Ting Liu (VUA), A Gut Feeling: Biomedical Knowledge Graphs for Interrelating the Gut Microbiome and Mental Health
16 Arthur Barbosa Câmara (TUD), Designing Search-as-Learning Systems
17 Razieh Alidoosti (VUA), Ethics-aware Software Architecture Design
18 Laurens Stoop (UU), Data Driven Understanding of Energy-Meteorological Variability and its Impact on Energy System Operations
19 Azadeh Mozafari Mehr (TU/e), Multi-perspective Conformance Checking: Identifying and Understanding Patterns of Anomalous Behavior
20 Ritsart Anne Plantenga (UL), Omgang met Regels
21 Federica Vinella (UU), Crowdsourcing User-Centered Teams
22 Zeynep Ozturk Yurt (TU/e), Beyond Routine: Extending BPM for Knowledge-Intensive Processes with Controllable Dynamic Contexts
23 Jie Luo (VUA), Lamarck's Revenge: Inheritance of Learned Traits Improves Robot Evolution
24 Nirmal Roy (TUD), Exploring the effects of interactive interfaces on user search behaviour
25 Alisa Rieger (TUD), Striving for Responsible Opinion Formation in Web Search on Debated Topics
26 Tim Gubner (CWI), Adaptively Generating Heterogeneous Execution Strategies using the VOILA Framework
27 Lincen Yang (UL), Information-theoretic Partition-based Models for Interpretable Machine Learning

28 Leon Helwerda (UL), Grip on Software: Understanding development progress of Scrum sprints and backlogs
29 David Wilson Romero Guzman (VUA), The Good, the Efficient and the Inductive Biases: Exploring Efficiency in Deep Learning Through the Use of Inductive Biases
30 Vijanti Ramautar (UU), Model-Driven Sustainability Accounting
31 Ziyu Li (TUD), On the Utility of Metadata to Optimize Machine Learning Workflows
32 Vinicius Stein Dani (UU), The Alpha and Omega of Process Mining
33 Siddharth Mehrotra (TUD), Designing for Appropriate Trust in Human-AI interaction
34 Robert Deckers (VUA), From Smallest Software Particle to System Specification - MuDForM: Multi-Domain Formalization Method
35 Sicui Zhang (TU/e), Methods of Detecting Clinical Deviations with Process Mining: a fuzzy set approach
36 Thomas Mulder (TU/e), Optimization of Recursive Queries on Graphs
37 James Graham Nevin (UvA), The Ramifications of Data Handling for Computational Models
38 Christos Koutras (TUD), Tabular Schema Matching for Modern Settings
39 Paola Lara Machado (TU/e), The Nexus between Business Models and Operating Models: From Conceptual Understanding to Actionable Guidance
40 Montijn van de Ven (TU/e), Guiding the Definition of Key Performance Indicators for Business Models
41 Georgios Siachamis (TUD), Adaptivity for Streaming Dataflow Engines
42 Emmeke Veltmeijer (VUA), Small Groups, Big Insights: Understanding the Crowd through Expressive Subgroup Analysis
43 Cedric Waterschoot (KNAW Meertens Instituut), The Constructive Conundrum: Computational Approaches to Facilitate Constructive Commenting on Online News Platforms
44 Marcel Schmitz (OU), Towards learning analytics-supported learning design
45 Sara Salimzadeh (TUD), Living in the Age of AI: Understanding Contextual Factors that Shape Human-AI Decision-Making
46 Georgios Stathis (Leiden University), Preventing Disputes: Preventive Logic, Law & Technology
47 Daniel Daza (VUA), Exploiting Subgraphs and Attributes for Representation Learning on Knowledge Graphs
48 Ioannis Petros Samiotis (TUD), Crowd-Assisted Annotation of Classical Music Compositions

---

2025 01 Max van Haastrecht (UL), Transdisciplinary Perspectives on Validity: Bridging the Gap Between Design and Implementation for Technology-Enhanced Learning Systems
02 Jurgen van den Hoogen (JADS), Time Series Analysis Using Convolutional Neural Networks
03 Andra-Denis Ionescu (TUD), Feature Discovery for Data-Centric AI
04 Rianne Schouten (TU/e), Exceptional Model Mining for Hierarchical Data

05 Nele Albers (TUD), Psychology-Informed Reinforcement Learning for Situated Virtual Coaching in Smoking Cessation
06 Daniël Vos (TUD), Decision Tree Learning: Algorithms for Robust Prediction and Policy Optimization
07 Ricky Maulana Fajri (TU/e), Towards Safer Active Learning: Dealing with Unwanted Biases, Graph-Structured Data, Adversary, and Data Imbalance
08 Stefan Bloemheuvel (TiU), Spatio-Temporal Analysis Through Graphs: Predictive Modeling and Graph Construction
09 Fadime Kaya (VUA), Decentralized Governance Design - A Model-Based Approach
10 Zhao Yang (UL), Enhancing Autonomy and Efficiency in Goal-Conditioned Reinforcement Learning
11 Shahin Sharifi Noorian (TUD), From Recognition to Understanding: Enriching Visual Models Through Multi-Modal Semantic Integration
12 Lijun Lyu (TUD), Interpretability in Neural Information Retrieval