UNIVERSITY OF MODENA AND REGGIO EMILIA

DEP. OF ENGINEERING “ENZO FERRARI"

DOCTORATE SCHOOL IN ICT

XXXVII CYCLE

Ph.D. DISSERTATION

Explainable Artificial Intelligence for Textual Applications: Open Challenges and Opportunities for Entity Matching and Fact-checking.

Candidate: **Andrea Baraldi** - *andrea.baraldi96@unimore.it*
Supervisor: Prof. Francesco Guerra - *francesco.guerra@unimore.it*

15 January 2025

Abstract

In the era of Artificial intelligence, advancements in machine and deep learning enabled the creation of models that effectively tackle a wide and rapidly expanding set of tasks. While these models frequently achieve extraordinary performance, sometimes surpassing human abilities, their complex architectures, which include millions of parameters, frequently obscure their internal reasoning. Furthermore, rather than using logic that people would find reasonable or reliable, models may achieve high accuracy by taking advantage of data artifacts. This lack of transparency becomes especially concerning in sensitive domains where AI decisions can significantly impact people's lives. AI systems can behave unfairly for various reasons: societal biases may be reflected in the training data, in the decisions made during the development, or in the systems themselves. The field of Explainable Artificial Intelligence (XAI) has emerged to address these issues, focusing on both interpreting existing models and designing inherently transparent AI architectures. This thesis explores and advances the frontiers of XAI through a systematic investigation across two critical domains: Entity-Matching (EM) and computational fact-checking. In EM, this research first explores novel techniques to explain black-box models, addressing unique challenges in this field. Traditional post-hoc XAI techniques, when directly applied to EM, do not account for its characteristics, i.e. comparing pairs of descriptions originating from independent sources. The research advances the state of the art by developing adaptations and enhancements of explanation techniques that specifically address these challenges and provide "what-if" scenarios that enable users to understand how change in input data would affect matching outcomes. These innovations improve the fidelity and trustworthiness of generated explanations.

The thesis then conducts an in-depth analysis of BERT's internal architecture when applied to EM, revealing crucial insights about how this state-of-the-art model processes and leverages information in entity descriptions. Experimental evaluations reveal that: fine-tuning on EM task mainly affects the last layers of BERT; this model recognizes the unique structure of EM datasets (records consisting of entity descriptions pairs); and BERT-based EM models do not heavily rely on pair-wise semantic similarity between tokens.

Then, an inherently interpretable EM system, and the novel concept of "decision units" are presented. This approach provides a new perspective on how to perform EM by generating effective, compact explanations tailored to this task. Results show this method achieves accuracy comparable to other state-of-the-art models while providing highly interpretable predictions.

The research extends to the emerging field of computational fact-checking, where explainability is crucial for trust and adoption. To assess how fact-checking models' reasoning resembles human logic, a novel evaluation methodology is introduced. This is accomplished by adapting XAI techniques, LIME and SHAP, to fact-checking models. The proposed method can effectively demonstrate how models exploit evidence pieces for the veracity prediction of claims. The explanations generated provide improved transparency and can highlight criticalities.

The research also contributes to the field of fair AI systems by introducing ExpGrad++, a novel approach based on the reduction method to address the problem of scalability in fair AI systems, will be presented. The two major innovations are adaptive sampling and interleaved exponentiated gradient with column-generation updates. These enhancements maintain accuracy and fairness guarantees, essential for practical applications, while increasing computational efficiency by an order of magnitude.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Overview and Scope of the Research

Digitalization is pervading the world in various aspects, from industry to personal life. In this scenario, advancements in Artificial Intelligence (AI) enabled the creation of models that effectively address an already wide yet rapidly expanding set of tasks, from healthcare and finance to education, law, and beyond. Increased automation in numerous applications is made possible by these models, which perform on par with or even better than humans. However, the rapid advancement of this technology is surpassing our capacity to comprehend it completely. These systems exploit complex machine-learning architectures that reach billions of parameters. Therefore, when looking into their internal architectures, their operation becomes obscure and incomprehensible. As a result, these models are employed in "black-box" mode, which presents issues on several levels since they are used without considering how they leverage data to make predictions.

It is not possible to audit and validate their decision-making processes. It's impractical to know with certainty how these models are using the data and whether they are leveraging subtle patterns within data, i.e. patterns that can sometimes deceive reliable, logical reasoning and instead reflect biases or artifacts in the data itself. The only way to assess their quality is to check the predictions made, regardless of the internal reasoning. This "black-box" nature erodes trust and makes it more difficult to figure out how to improve them.

For sensitive applications, such as healthcare, criminal justice, or financial lending, AI decisions could significantly affect people. Some form of explanation and justification of decisions helps to build trust and may be necessary for applying these models; regulatory standards (e.g., GDPR, AI Act) also increasingly require transparent and interpretable AI. Additionally, societal biases present in the training data, in the decisions made during development, or biases in the systems themselves tend to make AI act unfairly, causing

harm. This is a significant task requiring validation and control of the presence of biases in order to allow a safe deployment of AI. To responsibly and ethically apply AI models for direct-to-human applications it is essential to care about the aspects of transparency, interpretability, and fairness in model development.

This thesis focuses on addressing three key research problems connected to applied and operational machine learning, specifically concerning explanation, fairness, and integration. Each problem is tackled from a distinct perspective, but they are all deeply interrelated.

First, the research investigates the explainability of AI systems through several complementary studies: Chapters 3 and 5 develop novel Explainable Artificial Intelligence (XAI) techniques for Entity-Matching in data integration, while Chapter 4 analyzes how these systems make decisions from multiple perspectives.

Second, the research delves into computational fact-checking in 6, where explainability plays a crucial role. Fact-checking models are studied to evaluate how their reasoning aligns with human logic. XAI techniques such as LIME and SHAP are adapted to demonstrate how these models exploit evidence pieces for the veracity prediction of claims, offering improved transparency and the ability to highlight criticalities.

Third, the thesis addresses the fairness of AI systems in 7, with a particular focus on improving the scalability of fairness-aware methods. A novel approach, ExpGrad++, is proposed to enhance the computational efficiency of fairness algorithms while maintaining accuracy and fairness guarantees, ensuring their applicability in large-scale real-world scenarios.

Data Integration In the Big Data era data is being generated, collected, and analyzed at an unprecedented scale. According to Precedence Research: 'The global data integration market size was accounted at USD 13.6 billion in 2023 and it is projected to reach around USD 43.38 billion by 2033.' [107] While this opens up several new, exciting opportunities, it also introduces challenges. Business, government, and scientific organizations increasingly rely on massive amounts of data collected from both internal (e.g., CRM, Databases) and external data sources (e.g., the Web). Much of the value of increased digitization depends on the integration of different data sources. Data integration is complicated by issues like incompleteness (missing data), redundancy (duplicate information), inconsistency (conflicting records), and inaccuracy (errors). These challenges necessitate systems that can detect and resolve such problems to create a unified, reliable data source. Although humans can manually spot duplicate records, automatically aligning different data sources remains a complex challenge with no simple solution.

Entity matching is the field of research dedicated to solving the problem of identifying which records refer to the same real-world entity. The records are usually assumed to either be from two different data sources without duplicates or from the same data source with duplicates. It is not a new problem. A group of similar problems has been studied for a long time in a variety of fields under different names. Despite being researched for decades, entity matching remains a challenging problem in practice. State-of-the-art approaches based on Machine Learning (ML) and Deep Learning (DL) models are highly accurate but suffer from low interpretability. From the user's perspective, these models act as oracles. This is a critical problem in many operational scenarios where traceability, scrutiny, and users' confidence in the model are fundamental requirements as well as the model accuracy.

Applying explainable AI techniques to the field of Entity Matching opens new doors for practitioners over innovative methods to debug, scrutinize, verify, and validate the behaviors of any EM model. Explanation techniques provide a new set of tools to verify the robustness and improve the quality of existing approaches.

This thesis research begins by developing **novel adaptations of post-hoc XAI techniques** to account for EM's unique task of comparing entity descriptions from independent sources. Standard explanation methods are modified to address these specific challenge, offering users "what-if" scenarios illustrating how input changes impact matching outcomes. This approach enhances the reliability of explanations, making complex black-box models more interpretable for EM tasks.

Building on this, an **in-depth analysis of BERT's architecture** for EM tasks provides insights into how BERT processes and contextualizes paired entity descriptions. Experimental findings indicate that fine-tuning for EM significantly affects BERT's last layers, revealing that BERT recognizes the distinct structure of EM datasets, which consist of entity pairs, and relies minimally on pairwise semantic similarity.

This work additionally advances explainability in EM by presenting an **inherently interpretable EM model**, which introduces the concept of 'decision units' to produce concise, tailored explanations for EM. This innovative approach enables to achieve accuracy comparable to other state-of-the-art models, but differently from others, it provides explanations along with each prediction, thus making its workings more interpretable.

Automated Fact-Checking Fake news is thought to have an annual direct cost to the economy of about $78 billion across a variety of sectors. These costs span financial misinformation, reputation management, public health misinformation, online platform safety, brand safety, and political spending. Fake news is further diminishing public trust in specific sectors, trust in the news media has dropped from 55% in 2015 to 32% in 2019. The risks

and costs of fake news will only increase as technology evolves, despite efforts at detection. As long as the ad market incentivizes the production and fake news and it remains human nature to be reactive to news, then the global economy will continue to be at severe risk of harm. [30]

The process of determining whether statements made orally or in writing are true is known as fact-checking. This is a crucial journalism task that is usually done by hand by dedicated organizations like PolitiFact. In order to cope with the pace with which information and misinformation spread in modern media channels the problem of fact-checking has become crucial. Therefore, researchers have been exploring how fact-checking can be automated, using techniques based on natural language processing, machine learning, knowledge representation, and databases to automatically predict the veracity of claims. Automated fact-checking consists of three stages: (i) *claim detection* to identify claims that require verification; (ii) *evidence retrieval* to find sources supporting or refuting the claim; and (iii) *claim verification* to assess the veracity of the claim based on the retrieved evidence. *Evidence retrieval* and *claim verification* are sometimes tackled as a single task referred to as *factual verification*, while *claim detection* is often tackled separately. *Claim verification* can be decomposed into two parts that can be tackled separately or jointly: *verdict prediction*, where claims are assigned truthfulness labels, and *justification production*, where explanations for verdicts must be produced. [59] The first two phases can be addressed with satisfactory results using existing NLP approaches, providing useful tools to support human work. Claim verification, on the other hand, necessitates superior reasoning abilities that challenge current artificial intelligence techniques. To improve these models, explainability techniques can be used to support their actions and identify critical issues that, when solved, can lead to new, improved models.

The scope of this thesis in the area of automated fact-checking focuses on the specific task of claim verification. Although some fact-checking models are capable of generating a textual justification for the veracity prediction, it remains unclear how they make use of the input information. The purpose is to apply XAI tools that will assess how automated models use evidence and claim texts for claim verification. The intent is to determine which pieces are used the most and whether the model can engage in rational decisions.

Fairness in Machine Learning As machine learning algorithms become increasingly integrated into decision-making processes, fairness concerns have received extensive attention, there is a growing need to ensure its applications do not disproportionately impact historically disadvantaged populations and groups [36, 98, 23, 96, 19]. Fairness of AI systems has therefore become of paramount importance for governmental bodies, tasked

with regulating a society where decision-making increasingly relies on automated AI-based systems [46, 99, 71], as well as for large technology corporations, which need to mitigate fairness harms of AI systems [35, 57]. Many real-world applications of machine learning models, such as determining admission to a university, screening job applicants, disbursing government subsidies, identifying persons at high risk of disease, and so on, are prone to bias. Inherent biases and limitations in algorithms and training data can produce discriminatory outcomes and endorse societal biases. Discriminatory outcomes are situations where machine learning models generate predictions or decisions that consistently favor or disadvantage specific groups over others. [149, 136]. Societal biases are the preconceived notions, prejudices, or stereotypes in a society that can lead to unfair advantages or disadvantages for specific individuals or groups. ML and AI researchers have raised many questions about the source and the solution to fairness issues in AI and discussed various types of biases. [18, 72] In this context, fair treatment of individuals constitutes that decisions are made independent of sensitive attributes such as gender or race, such that individuals are treated based on merit [73, 89]. For example, one can aim for an equal probability of population groups to receive a positive treatment, or an equal treatment of individuals that only differ in sensitive attributes. [66] There are many different kinds of fairness harms [37, 138, 120], and there are many different ways to mitigate them by intervening at different points of the AI lifecycle [138], including during task definition, data collection, and when deciding whether to design the system at all. Chapter 7 study algorithmic techniques that seek to mitigate two broad categories of harms, *allocative harms* and *quality-of-service harms*, at the model training stage. Allocative harms occur when AI systems are used to allocate opportunities or resources in ways that can have significant negative impacts on people's lives such as in hiring, policing, education, and access to health-care [5, 97, 108, 124]. Quality-of-service harms occur when a system does not work as well for members of one group as it does for members of another group [26, 75]. For example, suppose a hospital is training an AI model to predict 30-day hospital readmission to prioritize high-risk patients for more intensive post-discharge care. Allocative harms might occur if certain subgroups of patients (e.g., Black patients) are disproportionately under-prioritized for more intensive care (i.e., have low selection rates) or are under-selected relative to their observed rate of readmission (i.e., have high false negative rates). To mitigate this harm, the hospital could train a model that incorporates suitable fairness constraints by constraining, for example, the difference between selection rates across different race groups.

The deployment of these systems to real applications additionally requires them to be capable of handling massive amounts of data with efficiency and speed. This is a crucial requirement for real-time applications.

To this end, this thesis presents ExpGrad++, a novel approach based on the reduction method to address the problem of scalability in fair AI systems, will be presented. The two major innovations are adaptive sampling and interleaved exponentiated gradient with column-generation updates. These enhancements maintain accuracy and fairness guarantees, essential for practical applications, while increasing computational efficiency by an order of magnitude.

Research Works The remainder of this thesis comprises a collection of appropriately extended research papers, where the key research themes were introduced and systematically developed. These works are organized into three main research areas:

The first area focuses on explanation and interpretability in entity matching. Key contributions in this domain include: *Using landmarks for explaining entity matching models*, published in EDBT [13]; *Landmark explanation: An explainer for entity matching models*, published in CIKM [12]; *An Intrinsically Interpretable Entity Matching System*, published in EDBT [11]; and *Analyzing how BERT performs entity matching*, published in the Proceedings of the VLDB [100].

The second area addresses explainability in computational fact-checking. This topic is explored in the article *Explaining The Role of Evidence in Data-Driven Fact Checking*, which is under submission at NACL and emphasizes the role of explainability techniques in understanding the reasoning of fact-checking models.

The third area investigates fairness and scalability in machine learning systems. This includes the article *Fair Classification with a Scalable Reduction Approach*, currently under submission at EDBT, which presents a novel solution to address these critical challenges.

Chapter 2

Related Work

Explainable AI in Entity Matching

Explainable AI The interpretation of machine learning techniques represents a hot topic. The approaches proposed in the literature can be grouped into intrinsic or post-hoc considering the phase when explanations are generated and model-specific or model-agnostic considering how they interact with models. In intrinsic explanation systems, the explanation is achieved by constructing self-explanatory models that incorporate interpretability directly into their structures. Simple models like decision trees, rule-based, and linear models belong to this category as they rely on structures humans can interpret directly. On the other hand, post-hoc techniques analyze the behavior of a model that is not directly interpretable by applying interpretable methods to the model of interest [92]. These can produce explanations by looking only at model predictions, treating the observed model like a black-box, thus being model-agnostic (i.e. they apply to any ML / DL model); or also leveraging its internal parameters, thus being model-specific to certain classes of architectures. Post-hoc model-agnostic approaches generally operate via a second intermediate model built from the first. The explanation is provided by building a surrogate interpretable model that mimics the behavior of the observed model in a specific area of data (local example). Researchers found a clear trend between the accuracy and interpretability of models, this is consistent with the fact that as models become more complex they become more accurate but less interpretable. Inherently interpretable models generally provide more faithful and undistorted explanations but may sacrifice prediction performance to some extent. Complex, non-interpretable models generally achieve better performance, but require additional explanation tools to be explained and the explanations obtained using post-hoc methods tend to be less faithful than the ones obtained from intrinsically interpretable models. [41]

Regardless of the explanation technique adopted, it is further possible to distinguish between global and local interpretations [41]. In the first case, the entire functioning of an ML / DL model is examined, while in the second its behavior is studied only locally (i.e. by explaining its logic on individual predictions).

The main exponent of the category of local post-hoc interpretation techniques is LIME [110], which exploits an interpretable linear surrogate model (e.g. Lasso) to evaluate the behavior of the original model in the neighborhood of a specific data instance. It will be used in our experiments, and an extension of it is Anchor [111], which generates explanations based on if-then rules. Some examples of global explanation systems are BRL [79] and Skater[1]. Similar techniques are *permutation feature importance* and *drop-column importance* [22], which can be used to detect the global relevance of features in any model.

The main problem in post-hoc interpretability is that there is no guarantee that the explanations they generate faithfully represent the behavior of the analyzed model. This is because these systems roughly reconstruct the boundaries of the real model behavior [147]. An alternative approach is the design of intrinsically explainable systems, which build predictions based on human interpretable data structures thus making the attribution of their decisions univocal with respect to human-understandable reasons [24].

Transformer architectures and derivative models (e.g., *BERT* and variants) [39, 143] that have recently achieved high importance in solving NLP tasks [70, 141] showed that the weights of the attention modules may produce explanations. This represents a very controversial aspect as the presence of some form of attention mechanism [10] is typically considered as evidence of their interpretability. Although this fact can be considered generically true, the complexity of modern transformer architectures makes the veracity of this consideration more uncertain [118, 134]. Since these attention modules are typically followed by several non-linear transformations, it seems that a univocal and direct association between the attention weights and the output of the model cannot be derived [70, 118, 100]. Nevertheless, these explanations have been considered as "relaxed" [113], since they provide a plausible (but not necessarily faithful) reconstruction of the decision-making process.

Entity matching Despite the effort put in the past 30 years, Entity Matching (EM), the task that identifies data items that refer to the same real-world entity, is still an open challenge. It represents one of the main steps of data integration and has been under study for several years. Many techniques have been proposed: from the more traditional rule-based approaches to the most modern machine learning and deep learning methods. Some examples of the first

[1] https://github.com/oracle/Skater

category are [139, 123]. They are intrinsically interpretable, however, the identification of the most effective set of matching rules is a complex and non-trivial task [102].

Recently, several approaches based on Deep Learning have proved particularly effective in solving this task. Some examples are DeepER [45], DeepMatcher [93], DITTO [82] and many others [148]. They leverage recurrent neural networks, possibly integrated with attention modules, to encode pairs of entities in multi-dimensional vectors and create a binary classifier based on the similarity of these embeddings to generate the matching decision. With the successful application of transformer architectures [135] in the NLP domain, EM models have also integrated this new technology. These are complex neural networks trained on large generalist corpus in a self-supervised manner, which are typically re-used in downstream tasks after the application of a fine-tuning process. Their application to EM tasks has pushed the state-of-the-art performance [101]. Some examples of BERT-based EM systems are [25, 81, 104]. In [25] the most recent transformer-based models are fine-tuned on the EM task, empirically demonstrating their high efficacy in solving the task even in dirty or textual datasets and without the need for a task-specific architecture. [82] proposes DITTO, which is now the most performing EM model proposed in the literature. It consists of a BERT architecture fine-tuned on the EM task which is further optimized by injecting domain knowledge (separators are added to mark the attributes in the EM entries), applying text summarization methods based on TF-IDF, and adapting data augmentation techniques for text to add (difficult) examples in the training data. In [104] a dual-objective training technique for BERT is proposed, which forces the model to predict the entity identifier in addition to the match / non-match decision. Another recent DL-based framework for EM is CorDEL [140], which exploits a novel contrastive approach to derive a matching decision. Its main intuition is to identify in pairs of entities components of similarity and dissimilarity deriving respectively from shared terms and unique terms. Recently these architectures have also found application in the blocking phase [131] and a survey on the adoption of DL architectures in EM is available in [15].

In addition to requiring a significant amount of annotated data and a complex configuration, the main problem with these systems is the inability to interpret their behavior, affecting their usability in business environments [32]. Nevertheless, their adoption in real business scenarios is hampered by several factors, including the need for large amounts of training data, the need for expert users for the configuration of their hyper-parameters and the inability to easily interpret how the models make their decisions. Explaining the behavior of ML and DL models is now a challenging research topic [41]. Its application to EM could facilitate the adoption of EM techniques in business scenarios. An improved ability to interpret the models would increase 1) user confidence in the adoption of ML and DL techniques, 2) the

ability to debug erroneous behaviors and diagnose unexpected results, and 3) improve the functionality of the approaches. Moreover, it would decrease the need for domain experts to evaluate the effectiveness of EM approaches, task that is typically executed through manual, expensive, and time-consuming processes.

Explainable Entity Matching Although several explanation systems have already been proposed in the literature (e.g., LIME [110], Shapley [54], Anchor [111], and Skater[2]), their application to EM tasks is not straightforward and only few approaches have partially addressed it [44, 132, 33, 90]. The main motivation is that EM is conceived by ML and DL systems as a binary classification problem, where the class shows if the pairs of entities described in the dataset records are matching, and the dataset entry is composed of pairs of attributes describing the same feature of different entities. This structure is "unusual" in ML and DL, where the records conversely describe single evidence. Moreover, the datasets are usually imbalanced: the number of records belonging to the matching class is far less than the non matching ones. Finally, the attributes describing the same features of different entities have close statistical distributions (or close word distributions in case of categorical attributes) even when they refer to different entities.

This motivated the realization of several studies on the use of interpretation techniques in the entity matching area [90, 132], and tools, like Mojito [33] and ExplainER [44], *LEMON* [14], and CERTA [129] have been proposed. ExplainER provides a unified interface for applying well-known interpretation techniques (e.g., LIME, Shapley, Anchor, and Skater) in the EM scenario. Mojito adapts LIME for the explanation of single EM predictions and represents the work closer to our approach. It extends LIME in two ways: 1) it exploits the subdivision of EM data into attributes, 2) it introduces a new form of data perturbation, called LIME-COPY [3], which allows generating match elements starting from non-match elements. They do so by copying attribute values belonging to one of the entities in a record into the corresponding attribute values of the second entity. The aim is to introduce a perturbation that increases the match probability between pairs of entities. But duplicating entire attribute values does not allow the approach to discriminate among the tokens that, thanks to the copy, will provide the same contribution in the explanation. CERTA introduces attribute-level explanations that typically align well with the way input data is structured and understood by users of structured relational databases. LEMON can aggregate tokens from the same entity description in single features.

[2]https://github.com/oracle/Skater

[3]In Section 3.2 this technique is referred to as Mojito Copy to emphasize that this technique is part of the Mojito tool.

BERT architecture overview. BERT [39] is a large transformer-based architecture whose main component is a self-attention module [135]. It takes as input a sequence of token representations, learns the reciprocal attention that each token of the sequence directs towards each other token (i.e. the attention weights), and outputs a new sequence obtained from the weighted average between the original token representations and the attention weights. BERT organizes self-attention modules on multiple layers, and each of them is divided into multiple parallel "heads" which act on separate linear transformations applied to the same input sequence of token representations. Several variants of this architecture have been proposed where a different dimension (e.g., different number of layers, etc.) or a different vocabulary (i.e. cased or uncased) are used. BERT is pre-trained on 3.3 billion tokens of English text to perform two tasks: the masking language modeling, which consists of predicting the token that has been masked by the input text, and the next sentence prediction, which consists of predicting the next sentence of an input text. Although this architecture can be used in a pre-trained form to obtain advanced token input representations, a fine-tuning process is usually applied. It typically consists of adding one or more fully connected layers to the BERT architecture and training the resulting network with respect to a reference task. In chapter 4, both the pre-trained version and a fine-tuned version are created to solve an Entity Matching task will be considered.

BERT inspection Recently, a thriving collection of works, identifiable under the term BERTology [112], has inspected the BERT architecture to assess its ability to learn correct linguistic artifacts. A first category of these works exploits probing classifiers built on top of different BERT intermediate representations (such as contextual embeddings or attention heads) to understand if these components capture specific linguistic patterns (e.g., dependencies between part-of-speech) [56, 106, 84, 128, 127]. From these studies, it emerged that BERT is able to encode a great variety of syntactic and semantic relationships in different regions of the network and in a hierarchical way (i.e. through syntactic tree structures) [83, 64]: simple syntactic information is captured in the first layers, while more complex relationships in the deeper layers. Despite these findings, the reliability of these probing tasks has recently been debated as their results can be easily misinterpreted [128] or perturbed by the evaluation methodology itself [84, 28, 63, 144]. Parallel to these works, several approaches directly inspected the BERT architecture [34, 67, 77]. Unlike probing models, they do not depend on any auxiliary supervised task and therefore they do not require additional training. The study proposed in chapter 4 belongs to the latter category, and focuses on the analysis of BERT's data structures when employed to solve EM tasks. Although many

studies analyze BERT's behavior in various tasks [67, 60], to the best of our knowledge, this is the first work about Entity Matching.

Automated Fact-Checking

The opaqueness in automated fact-checking models diminishes user confdence and impedes the discovery of harmful biases, it is essential to understand the "reasoning" behind model predictions, i.e, the explainability of the fact verifcation approaches. [76] Many authors define explanations for fact verifcation as rationale extraction [7, 122], i.e. searching for a minimal portion of input (rationales) that can be suffcient (solely based on the rationales) to derive a veracity prediction. The evidence retrieval process often fails to produce a perfect set of evidence; therefore, a more advanced model is employed to refine the selection, providing a minimal, sufficient, and more consistent subset of evidence for accurate claim verification. Unlike these approaches, which focused on creating inherently explainable verification models [76] (intrinsic explainability), the work in 6 aims to provide explanations for decisions made by black-box models (post-hoc explainability). In contrast with some attempts to provide textual justification for the verdict [8], we emphasize the need to explicitly enable the users to check how the input evidence contributes to the prediction.

Several verifiers come with intrinsic explainers. Some systems decide each piece of evidence's usefulness before making a claim label prediction [38]. The evidence label is thus not based on how a fact-checking model used it but on an LLM's decision on how pertinent the evidence could be to label the claim. Other approaches imply the training of an explainer [122, 121], while the post-hoc methods do not require training. These systems cannot provide information on the role played by the evidence in the model decision, i.e., our approach measures how an evidence piece steers a claim's labels towards *'Refutes'*, while another piece steers it toward *'Supports'*. Neither they can be used out of the box on any verifier.

We focus on popular verification models based on pre-trained language models, which have demonstrated competitive performance in practice and the best trade-off when considering also cost/energy issues [103]. This focus ensures that our approach aligns with current state-of-the-art models while addressing the need for transparency in fact-checking systems. For our study, we use two local post-hoc attribution methods, LIME [110] and SHAP [87], which explain a prediction by measuring how different parts of the input (the evidence, in our setting) contribute to the prediction.

Fairness of AI

Fairness of AI systems is a topic of many academic venues (FAccT, AIES, FORC), books [16], and surveys [105, 29, 89]. The topic has also seen a growing interest in the database community. For example, Islam et al. [68] carry out an extensive evaluation of 13 fair classification approaches on several benchmark datasets using a variety of fairness metrics; Stoyanovich et al. [125] study issues related to responsible data management; and Shahbazi et al. [119] analyze fairness of entity matching.

In this paper, we conceptualize fairness through the lens of fairness harms [37, 138, 120], by which we mean negative impacts on groups of people, such as those defined in terms of race, gender, age, or disability status. In fairness literature, this is also referred to as group fairness [43]. There are various alternative frameworks of fairness of AI systems, which we do not pursue here, such as individual fairness [43] and fairness based on causal reasoning [86].

Previous fairness mitigation techniques can be broadly divided into three large groups. Pre-processing techniques seek to mitigate fairness issues by transforming the input data before it is fed to any standard ML method. CALMON [27] and FELD [51] are the reference baselines of pre-processing techniques that we use in the experiments in Section 7.3. In-training techniques incorporate quantitative definitions of fairness into existing ML techniques. ZAFAR [145, 146] is the representative approach from this category that we consider as a baseline in the experiments. The reduction approach also belongs in this category, but compared with ZAFAR, it is implemented as a wrapper of any generic ML approach. Finally, post-processing techniques apply adjustments to outputs of an already trained model to mitigate fairness issues in downstream decisions, and they typically require access to sensitive attributes at inference time. HARDT [62] is the post-processing approach we consider as a baseline. As already mentioned in the Introduction, often the sensitive attributes cannot be accessed at inference time because e.g., prohibited by law. Therefore, even if HARDT provides good accuracy and runtime performance (see Section 7.3), in practice it cannot be deployed in many use cases. As a general concern across all three groups, it is unclear which approaches scale to large datasets since most previous studies use datasets with fewer than 50,000 data points [49].

Chapter 3

Using Landmarks for Explaining Entity Matching Models

In this chapter, Landmark Explanation is presented, a system for explaining EM model predictions, that extends the capabilities of a post-hoc perturbation-based local explainer over this specific scenario. The approach leverages textual datasets, and the explanations, as usual for this kind of system, are expressed in the form of the impacts attributed to each token in the record. Post-hoc perturbation-based explainers analyze the records to explain and build a surrogate linear model where the features are the tokens (e.g., the words in case of textual attributes) composing the attribute values. The explanation is directly generated from the surrogate model: its linear coefficients represent the importance of the tokens. This model is trained with synthetic data, generated via a two-step approach where the values of the records to explain are properly altered (in the perturbation phase) and then passed to the original model to get their class (in the reconstruction phase).

Example 1. *Figure 3.1 shows an example of a record describing a pair of entities. A suffix is added to the attribute names to show which entity they are describing. The application of a DL based model to the record (e.g. DeepMatcher) let us know that the entities in the record do not refer to the same real-world entity. This is evident for a human person, being clear from the attributes that the left entity is a digital camera and the right entity is a leather case. But the EM model is not able to explain the reasons for its choice. The application of a post-hoc explainer gives us an explanation of the behavior of the model for the example at hand in the form of token contribution over the match probability. In our examples, tokens like* `sony`, `lens`, `dslra200w` *can be considered as an evidence of the fact that the record is describing a non-matching entity, thus a good explanation should assign them a negative contribution over the match probability.*

left_name	right_name	left_description	right_description	left_price	right_price
sony digital camera with lens kit dslra200w	nikon digital camera leather case 5811	sony alpha digital slr camera with lens kit dslra200w 10.2 megapixels	leather black	849.99	7.99

Fig. 3.1 Example of EM record to explain.

Post-hoc systems require building a surrogate model to provide explanations for an existing model. The explanations are usually obtained following a four-step approach: (1) *perturbation generation*: the record to explain is altered in multiple ways (mainly by removing tokens). This procedure generates a new set of synthetic records called *perturbation dataset*. (2) *labeling the perturbation dataset*: The perturbation dataset is supplied as input to the original model to get the class scores (the approach is model-agnostic). The result of this process is a synthetic labeled dataset. (3) *surrogate model generation*: the synthetic labeled dataset is used to train *the surrogate model*, i.e., a linear model that approximates the behavior (the class scores) of the original model in the *perturbation dataset*. (4) *explanation generation*: the surrogate model learns a coefficient for each input term. The set of scores defines the explanation of the original model on the target record. A similar idea has been applied in many proposals as LIME [110], Shapley [54], Anchor [111], and Skater[1]).

The application of explanation systems to EM scenarios is still at the early stage. The adaptation of state-of-the-art explanation systems to this scenario is not straightforward. The direct application of a perturbation mechanism based on token removals is not effective for EM datasets. Firstly, the dataset structure used in ML and DL datasets typically represents single "entities" described by multiple features. On the other hand, EM datasets describe the features of two entities. Removing random tokens is likely to affect both the entities of an EM pair. The generated synthetic records may then contain *null perturbations* where a pair of tokens representing the same feature are removed from both entity descriptions. These inconsistent perturbations lead to biased explanations. Secondly, considering non-matching entities, which constitute the majority of the EM datasets, perturbations that remove tokens have little effect on their match probability, which begins and remains near zero, resulting in unfit explanations.

Landmark Explanation addresses these issues by introducing two main innovations. The first is the generation of *two explanations for each EM record*, each one explaining the model decision from the perspective of one of the two entities described in the record. These explanations are generated by selecting one of the entities constituting a dataset entry in turn

[1]https://github.com/oracle/Skater

as a *landmark*. The landmark is preserved from the perturbation, which is subjected to the other entity (the varying entity). This produces two sets of impacts each detected for the tokens of the perturbed description given the other description as *landmark*. The second is a mechanism of *injection of tokens* between the descriptions to increase the informativeness of the explanations, which is particularly helpful for computing explanations of non-matching records. Before the perturbation, additional tokens extracted from the landmark entity are injected into the varying entity. The perturbation of the varying entity with injected tokens produces a set of synthetic entities. These will all be concatenated with the landmark entity to generate the synthetic EM dataset used to train the surrogate model. The idea is to contrast the asymmetric nature of the problem: an explanation of a matching pair is always composed of "interesting" tokens since they express the reason why the entities have been considered as matching. The same does not happen for non-matching entities, since non-matching entities have many reasons to be different. So it is difficult to generate an explanation with interesting tokens for non-matching pairs. Therefore the problem is to generate the most interesting explanations. These are the ones involving tokens from one entity that if used to describe the second entity would have brought the EM model to classify the record as matching. Thanks to the injection described above non-matching pairs are pushed to be match and the resulting explanation will be more interesting.

Example 2. *To explain the inference of an EM model applied to the record in Figure 3.1* , Landmark Explanation *generates two explanations. The top 3 tokens generated by the explanation with the left entity as a landmark are leather, nikon and 5811. These tokens are the ones that best differentiate entities. This means that if the left entity were described by these tokens, the record would probably be classified in the matching class by the EM model. The top 3 tokens generated by the second explanation with the right entity as a landmark are dsira200w, lens and 849.99.*

Landmark Explanation is evaluated coupled with LIME. The results of the experiments show that the explanations generated outperform the ones of the competing approaches in accuracy and "interest" for the users.

Summarizing, the main contributions of this chapter are: (1) the introduction of Landmark Explanation, a tool that extends the capability of a generic post-hoc perturbation-based explainer to generate accurate local explanations of EM models; (2) the realization of an extensive experimentation of Landmark Explanation coupled with the LIME explainer [110] to demonstrate the effectiveness and the quality of the approach in comparison with different competitor systems.

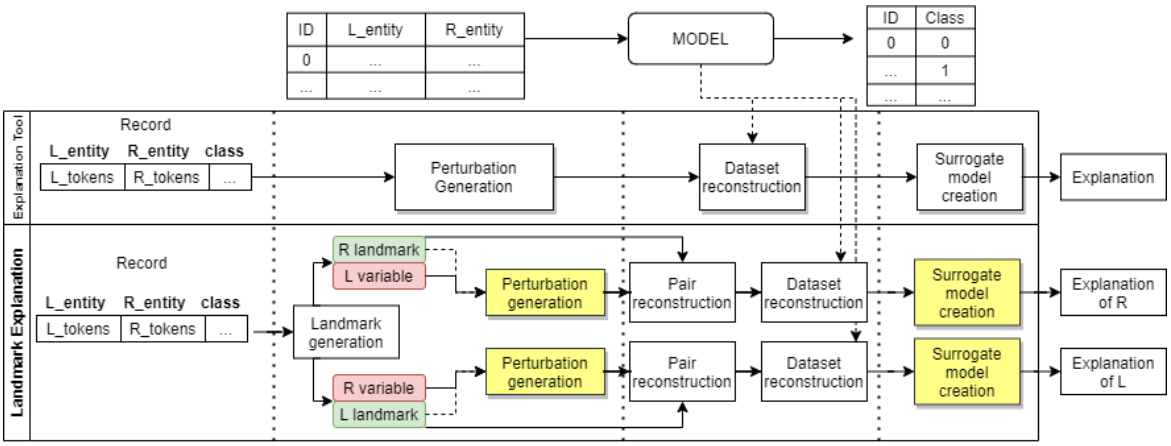


Fig. 3.2 A generic post-hoc perturbation based explanation system (at the top of the image) compared with its extension with the Landmark Explanation Framework (at the bottom).

3.1 The Landmark Explanation approach

Landmark Explanation is a generic and extensible framework that can extend a generic local post-hoc and model-agnostic perturbation based explanation systems to the interpretation of EM model predictions. The main assumption of these generic systems is that the prediction of a model computed on a given instance can be approximated by a linear function of the predictions calculated in the neighborhood of that input instance. Their functional architecture is shown at the top of Figure 3.2 and can be schematized in 3 main blocks: the component for the *Perturbation generation*, for the *Dataset reconstruction* and for the *Surrogate Model Creation*.

The *Perturbation generation* component takes the record to analyze as input and generates a number of perturbations from it. These perturbations constitute a new features space, defined in the neighborhood of the record, which, once integrated by the *Dataset reconstruction* component with the predictions of the original model, can be used by the *Surrogate Model creation* component to infer its behavior in the locality of the record.

Landmark Explanation extends the generic explanation system to improve its effectiveness on datasets representing EMs. In particular, as shown at the bottom, of Figure 3.2, Landmark Explanation adds the *Landmark generation* component before the perturbation. This component generates for the record to explain two representations, where only one of the two entities composing the item will be subject to the perturbation and the other one will be kept fixed as a landmark. This constitutes the input for the *Perturbation generation* component that will be called twice, once for each representation. In this way, each perturbation obtained with this process will involve the attributes of one entity. The tokens of the second entity (the landmark entity) will be added by the *Pair reconstruction* component before the

Dataset reconstruction. This allows Landmark Explanation to perturb the information of one entity at a time while preserving the pairwise structure of the EM data. A perturbation of the input entity pair is generated by varying only one of the two entities while preserving the pairwise structure of the EM data.

Note that the component performs differently when the record to explain is referring to a non-matching class. In this case, the tokens of both entities are concatenated into the varying entity and passed to the *Perturbation generation* component. The behavior of the *Pair reconstruction* component does not change, by concatenating after the perturbation the contribution of the landmark entity. This mechanism has been implemented to contrast the dataset imbalance and to generate explanations that can be more interesting for the users since based on a richer set of tokens. Finally, as for the generic explanation system, the synthetic dataset just created is used to train a linear model whose coefficients constitutes the explanation. These coefficients can be positive or negative thus indicating which tokens should be added (the one with a positive score) and which should be removed (the one with negative score) to create a description that is closed to the reference entity.

The yellow-shadowed components in the Figure are the ones provided by the explanation system that is extended. In our experiments, these components are provided by the LIME explainer. Since Landmark Explanation conceives them as black box modules, other explanations systems can be easily coupled with our approach.

3.1.1 Landmark Explanation components

Landmark Explanation is composed of three main components for the *Landmark generation*, the *Pairs reconstruction* and the *Dataset reconstruction*.

Landmark generation component. The goal of this component is to generate input for the *Perturbation generation* component. A *Tokenizer* is firstly needed to transform the dataset entry in a format suitable for generating meaningful explanation. A tokenization mechanism is implemented similarly to the one adopted in other systems as Mojito [33] that preserves the structure of the pair of entities described in the record. A token is generated for each space-separated term in the attribute values. A prefix is introduced to each token to indicate the attribute where the original value is located in the entity schema. The prefix enumerates the tokens, to manage multiple occurrences of the same word in an attribute value.

After the tokenization, Landmark Explanation implements two mechanisms for performing this task. With the *single-entity generation*, the tokens composing the entities are separated and the perturbation component is called twice, each time with the element of a different entity. The output for each execution are the tokens of one entity (the landmark)

and a number of perturbations for the second entity. This technique generates a perturbation that highlights the differences of one entity with respect to the other. It is then particularly effective when the record to explain is belonging to the matching class.

With the *double-entity generation*, the perturbation component receives as input the tokens of an artificial entity created by concatenating, for each attribute, the tokens of both the entities. The output for each execution are then the tokens of one entity (the landmark) and a number of perturbations of the artificial entity created by the concatenation. The idea of this technique is to generate the perturbation of a more extensive set of tokens (obtained by the union of the tokens of both the entities) that is effective for generating explanations for records classified as non-matching items.

Unlike Landmark Explanation, Mojito treats attributes atomically, distributing its impact equally to its constituent tokens. Furthermore, Landmark Explanation analyzes the diversified impact that the same token can generate depending on the entity considered as a landmark for the explanation.

Pair reconstruction component. The component receives as input the landmark entity and a number of perturbations of the tokens of the varying entity and "reconstructs" the corresponding pairs of entities (one for each perturbation). The prefixes introduced by the *Tokenizer* are exploited for this purpose and removed from the generated records.

Dataset reconstruction component. This component generates the synthetic dataset to be used for training the surrogate linear model. This is obtained by passing each pair of entities reconstructed by the previous component to the original EM model for getting the predicted class.

3.2 Experimental evaluation

The explanations generated by Landmark Explanation are evaluated according to two main perspectives: their reliability in representing the EM Model (in Section 3.2.2) and the "quality" of the explanation provided (in Section 3.2.3).

3.2.1 Experimental setup

The experiments run on a VM deployed on Google Cloud with 12 GB of RAM, GPU K80, and Intel(R) Xeon(R) CPU @ 2.30GHz.

Dataset and Model. The EM model explained in the experiments is a Logistic Regression Classifier. Landmark Explanation has been experimented against the datasets provided by

Dataset	Type	Datasets	Size	% Match
S-BR	Structured	BeerAdvo-RateBeer	450	15.11
S-IA		iTunes-Amazon	539	24.49
S-FZ		Fodors-Zagats	946	11.63
S-DA		DBLP-ACM	12,363	17.96
S-DG		DBLP-GoogleScholar	28,707	18.63
S-AG		Amazon-Google	11,460	10.18
S-WA		Walmart-Amazon	10,242	9.39
T-AB	Textual	Abt-Buy	9,575	10.74
D-IA	Dirty	iTunes-Amazon	539	24.49
D-DA		DBLP-ACM	12,363	17.96
D-DG		DBLP-GoogleScholar	28,707	18.63
D-WA		Walmart-Amazon	10,242	9.39

Table 3.1 Magellan Benchmark

the Magellan library[2] which is considered as a standard benchmark for the evaluation of EM tasks. The datasets are listed in Table3.1, where the size and the percentage of records representing matching entities are shown. The records in all datasets represent pairs of entities described with the same attributes. A label is provided to express if the record represents a matching / non-matching pair of entities. In the experiments, 100 records per label were first sampled and their explanations were computed. Note that all records are sampled when the dataset contains less than 100 records (see for example the dataset S-BR which contains only 68 records labeled as matching entity).

3.2.2 Reliability of the explanations

The goal of the experiment is to evaluate the reliability of the explanations generated by Landmark Explanation in interpreting the behavior of an EM model through single predictions. An explanation is considered reliable if it is able to consistently recognize the importance of the features with the EM model. To evaluate this, two kinds of experiments were performed, one analyzing the weights assigned by Landmark Explanation to the *tokens* it generates, the second the weights assigned by the EM model to the dataset *attributes*.

Token-based evaluation

Through this first kind of experiment, it has been evaluated if the weights assigned by Landmark Explanation to the tokens generate a surrogate model consistent with the EM model. An experiment similar to the one proposed in the evaluation of LIME was performed: 25% of tokens are randomly selected and removed from the record to explain, defining

[2]https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

(a) Matching label.

	Single		**Double**		**LIME**	
	Accuracy	MAE	Accuracy	MAE	Accuracy	MAE
S-BR	**0.923**	**0.121**	0.796	0.136	0.830	0.147
S-IA	**0.940**	**0.226**	0.793	0.251	0.847	0.240
S-FZ	**0.934**	**0.228**	0.841	0.237	0.865	0.236
S-DA	0.887	0.171	**0.894**	**0.164**	0.573	0.337
S-DG	**0.836**	**0.196**	0.823	**0.196**	0.757	0.200
S-AG	0.896	**0.074**	**0.903**	0.112	0.698	0.148
S-WA	**0.954**	**0.071**	0.928	0.115	0.659	0.228
T-AB	**0.908**	**0.066**	0.854	0.146	0.758	0.118
D-IA	0.899	**0.090**	**0.975**	0.112	0.780	0.156
D-DA	0.942	**0.030**	**0.979**	0.041	0.940	**0.025**
D-D	0.929	**0.107**	**0.963**	0.152	0.891	0.115
D-WA	**0.916**	**0.045**	0.901	0.090	0.813	0.074

(b) Non-matching label.

	Single		Double		LIME		Mojito Copy	
	Accuracy	MAE	Accuracy	MAE	Accuracy	MAE	Accuracy	MAE
S-BR	0.747	0.092	**0.927**	**0.037**	0.843	0.100	0.011	0.369
S-IA	0.669	0.248	**0.736**	**0.127**	0.624	0.267	0.022	0.569
S-FZ	0.811	0.188	0.853	**0.134**	**0.953**	0.189	0.032	0.681
S-DA	0.975	**0.021**	0.590	0.287	**0.985**	0.066	0.005	0.574
S-DG	0.895	**0.086**	0.660	0.306	**0.935**	0.107	0.005	0.504
S-AG	0.835	0.107	0.895	**0.056**	**0.905**	0.097	0.010	0.445
S-WA	**0.990**	**0.028**	0.955	0.217	0.890	0.352	0.000	0.746
T-AB	**0.860**	0.076	0.680	**0.047**	0.795	0.092	0.045	0.328
D-IA	**0.874**	**0.019**	0.291	0.070	0.390	0.129	0.242	0.191
D-DA	0.615	0.071	0.300	**0.027**	**0.690**	0.036	0.010	0.173
D-DG	0.540	0.305	0.375	**0.118**	**0.640**	0.235	0.040	0.437
D-WA	0.500	0.184	**0.785**	**0.078**	0.500	0.192	0.005	0.380

Table 3.2 Token-based evaluation.

a new item. The probability scores obtained by passing the new item to the EM model were compared with the ones of the original record, subtracting the sum of the coefficients associated with the removed tokens. If the explanation model correctly represents the EM model these two values should be close. The experiments have been repeated 100 times for each class (see the beginning of Section 3.2), and the performance obtained was measured by means of two metrics: the mean average error (MAE) and the accuracy on the predicted class. The experiments were performed for all datasets and testing all techniques for generating the perturbations as reported in Table 3.2. Note that column LIME shows the results obtained with LIME / Mojito Drop[3] with the same setting. Non-matching settings also include a comparison with the Mojito Copy technique, which has been designed for this kind of record.

[3]the Mojito Drop technique implements the LIME approach

Discussion. Table 3.2a shows that Landmark Explanation, applied to records labeled as matching entity, performs better than LIME in the datasets when the perturbation is generated with the single-entity technique (it obtains better accuracy in all datasets and low MAE in 11/12 datasets). The double-entity generation technique performs slightly worse: in 9/12 it obtains better accuracy and in 6/12 lower MAE). Nevertheless, the scores, when worst, are very close to LIME. Note that in some datasets there is some small contradiction between accuracy and MAE scores computed for the same dataset. For example, Landmark Explanation applied to the S-BR dataset with the double-generation configuration has a better MAE score than LIME/Mojito Drop. This is not the same for the accuracy value, where LIME performs better. This is motivated by the fact that the probability scores generated by the model are close to the decision threshold (fixed to 0.5). Then, small fluctuations in the surrogate model can generate mismatches in the class predicted by the explained model for the records, even if the EM model and the surrogate model are very close. Table 3.2b shows the accuracy and the MAE obtained analyzing records referring to non-matching labels. In this scenario, the double entity perturbation obtains the best scores with an accuracy better than LIME/Mojito Drop in 4/12 datasets and a lower MAE in 10/12 datasets. The reason is due to the effect of the duplicated tokens inserted in the dataset used for training the surrogate model. These tokens, being similar to the ones of the entities described in the record, push the EM model to classify the record towards a matching label even in the case of an imbalanced dataset. By using the same tokens, the entities will likely be considered by the model as similar. Conversely, the perturbations generated by LIME / Mojito drop are subsets of the original record, which, in this case, was classified as non-match. By removing tokens from descriptions of entities classified as non-matching, the probability score of the EM model usually decreases and it is unlikely to obtain descriptions of entities that a classifier evaluates as matching. If the decision threshold is pushed to 0.4 (instead of 0.5), Landmark Explanation would obtain a better performance than LIME/Mojito drop in 10/12 datasets.

Note that the copying technique introduced by Mojito to manage records associated with non-matching labels does not show high performance. The reason is that Mojito generates a perturbation by duplicating entire attributes.

The result of this operation is that the tokens of the replaced attribute have the same weights, thus decreasing the performance.

Lesson learned. The surrogate model built by Landmark Explanation with the single-entity perturbation is an accurate representation of the EM model for records representing matching pairs of entities. The model built with the double-entity perturbation is an accurate representation of the EM model for record representing non-matching pairs of entities.

(a) Matching label.

	Single	Double	LIME
S-BR	1.000	1.000	1.000
S-IA	0.261	**0.538**	0.495
S-FZ	**0.592**	**0.592**	0.143
S-DA	**0.520**	**0.520**	0.200
S-DG	1.000	1.000	1.000
S-AG	**1.000**	0.545	0.545
S-WA	0.901	**1.000**	0.544
T-AB	0.545	0.545	0.545
D-IA	0.892	**0.939**	0.848
D-DA	1.000	1.000	1.000
D-DG	1.000	1.000	1.000
D-WA	0.526	**0.681**	0.526

(b) Non-matching label.

	Single	Double	LIME	Mojito Copy
S-BR	0.733	**1.000**	**1.000**	**1.000**
S-IA	0.312	0.538	0.687	**0.756**
S-FZ	0.333	0.518	**0.864**	0.414
S-DA	0.200	**1.000**	0.200	0.520
S-DG	**1.000**	0.520	**1.000**	0.333
S-AG	**1.000**	0.545	0.545	0.545
S-WA	0.573	**1.000**	0.872	**1.000**
T-AB	0.545	0.545	0.545	**1.000**
D-IA	0.925	0.899	0.776	**0.939**
D-DA	0.813	**1.000**	**1.000**	**1.000**
D-DG	**1.000**	**1.000**	**1.000**	0.813
D-WA	0.681	0.681	0.681	0.681

Table 3.3 Attribute-based evaluation (weighted Kendall measure applied on the ranked list of attributed as generated by the EM and the surrogate model).

Attribute-based evaluation

The attribute-based evaluation proceeds in the opposite direction: it starts from the internal structure of the EM model and evaluates if the weights it gives to the attributes are close to the ones derived from the tokens obtained by Landmark Explanation. For this reason, the weights given to the dataset attributes by the Logistic Regression model used as EM model in the experiments are analyzed and the attributes are ranked according to their absolute values. A similar operation have been done with the surrogate model, where the weights of the attributes have been computed by summing the absolute weights of their composing tokens. The idea is that the order of the attributes computed on the basis of their weights should be the same in both models. In Table 3.3, the correlation are computed by applying the weighted Kendall tau correlation measure, between the ranked list of attributes of the EM and surrogate model.

Discussion. Table 3.3a shows the experiments on the records representing matching entity pairs. The correlation scores achieved by Landmark Explanation with the double-entity perturbation approach are better or equal to the ones achieved by LIME/Mojito for all dataset. Table 3.3b shows the experiments on the records representing non-matching entity pairs. In this case, the single-entity configuration obtained better/equal results than LIME/Mojito Drop in 7/12 datasets (4/12 against Mojito Copy); the double-entity configuration obtained better/equal results than LIME/Mojito drop in 9/12 datasets (the same against Mojito Copy). Note that Mojito Copy, that has been explicitly designed for non-matching entities, performs

better than LIME/Mojito Drop in 5/12 datasets only and equal/close to LIME/Mojito Drop in 4/12 datasets and worst in the remaining 3 datasets.

Lesson learned. Landmark Explanation creates surrogate models that maintain a relative importance of the attributes similar to the ones of the EM model to explain.

3.2.3 Quality of the explanations

To introduce this experiment, let us consider an application that aims to provide the explanation for a record labeled as a non-matching entity. The tokens of non-matching are "less polarized": there are many reasons to be dissimilar for two entities. For this reason, it is easy for this application to say why two entities do not match, since there is plenty of tokens that do not match between the entities in the record. Nevertheless, the explanation would be more interesting if the tokens returned would be the ones changing class of the record from non-matching to matching. In other words, an interesting explanation for non-matching entities should identify the tokens that, if present in the second entity, would result in the record being classified as matching.

This section describes the evaluations performed to evaluate the aforementioned situation. The experiments are similar to the first experiments described in Section 3.2.2, but in this case the tokens to remove are selected. For sake of completeness, A similar experiment with records classified as matches even if this evaluation is less meaningful is performed. When the record is associated with a matching label, all positive tokens are removed (all tokens that contribute to the decision). The negative tokens are removed when the label represents a non-matching record. In Table 3.4 the *interest* is measured to evaluate the experiment , which is the accuracy computed on the records where the removal of the tokens was able to generate a change in the label.

Discussion. Table 3.4a shows that Landmark Explanation is good but slightly worse than LIME in terms of interest, when the records are labeled as matching class. This happens even if the surrogate model is really accurate (the MAE score is the lowest for all experiments with the single-entity configuration). The problem is that in most of the cases, even removing all tokens, the explanation created by Landmark Explanation belongs to the same class as before the token removal. Note that if a decision threshold is set to 0.4, our approach has the best results in all datasets. Table 3.4b shows that the explanations of non-matching entities generated by Landmark Explanation in the setting double-entity outperform the ones of Lime/Mojito Drop and Mojito Copy.

(a) Matching label.

	Single	Double	LIME
S-BR	0.643	0.593	**0.686**
S-IA	0.652	0.404	**0.702**
S-FZ	0.606	0.447	**0.612**
S-DA	**1.000**	0.940	0.965
S-DG	0.660	0.610	**0.925**
S-AG	0.955	0.800	**0.990**
S-WA	**1.000**	0.785	0.870
T-AB	0.985	0.575	**0.995**
D-IA	**0.561**	0.278	0.311
D-DA	0.695	0.715	**0.800**
D-DG	0.635	0.530	**0.735**
D-WA	**0.915**	0.545	0.880

(b) Non-matching label.

	Single	Double	LIME	Mojito Copy
S-BR	0.298	**0.927**	0.331	0.011
S-IA	0.545	**0.736**	0.393	0.000
S-FZ	0.079	**0.853**	0.047	0.000
S-DA	0.000	0.030	0.000	0.005
S-DG	0.020	**0.545**	0.020	0.000
S-AG	0.075	**0.895**	0.070	0.010
S-WA	0.015	**0.955**	0.000	0.000
T-AB	0.305	**0.680**	0.340	0.045
D-IA	**0.670**	0.291	0.379	0.027
D-DA	0.205	**0.300**	0.125	0.000
D-DG	0.200	**0.375**	0.160	0.030
D-WA	0.190	**0.785**	0.130	0.005

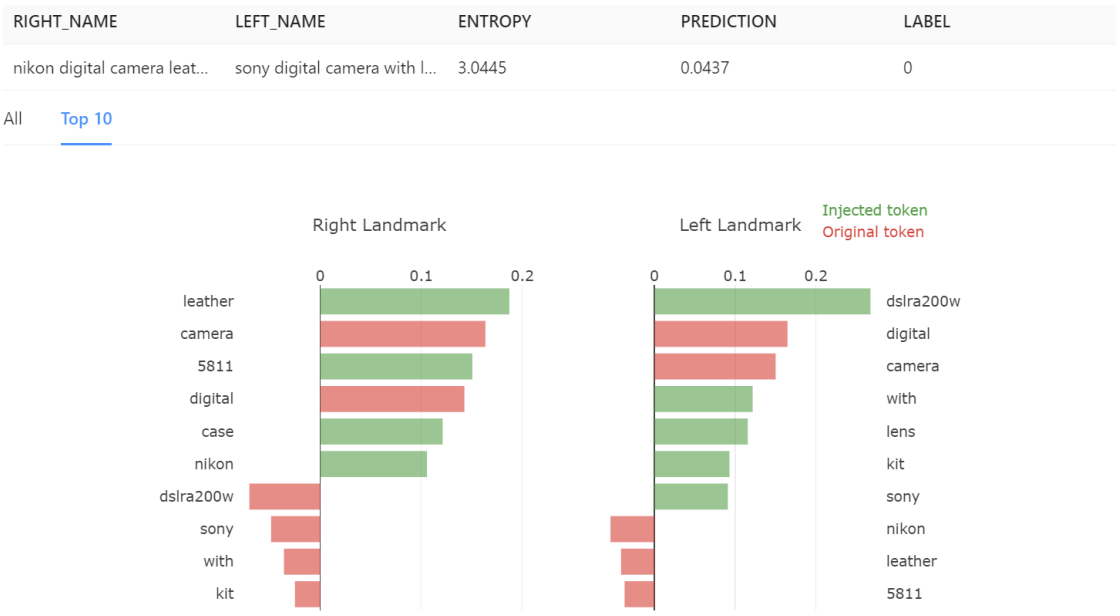
Table 3.4 Evaluation of the interest associated to the computed explanations.

Lesson learned. Landmark Explanation generates interesting explanations, and the perturbation made with the double-entity generation technique effectively increases "the interest" of non-matching record explanations.

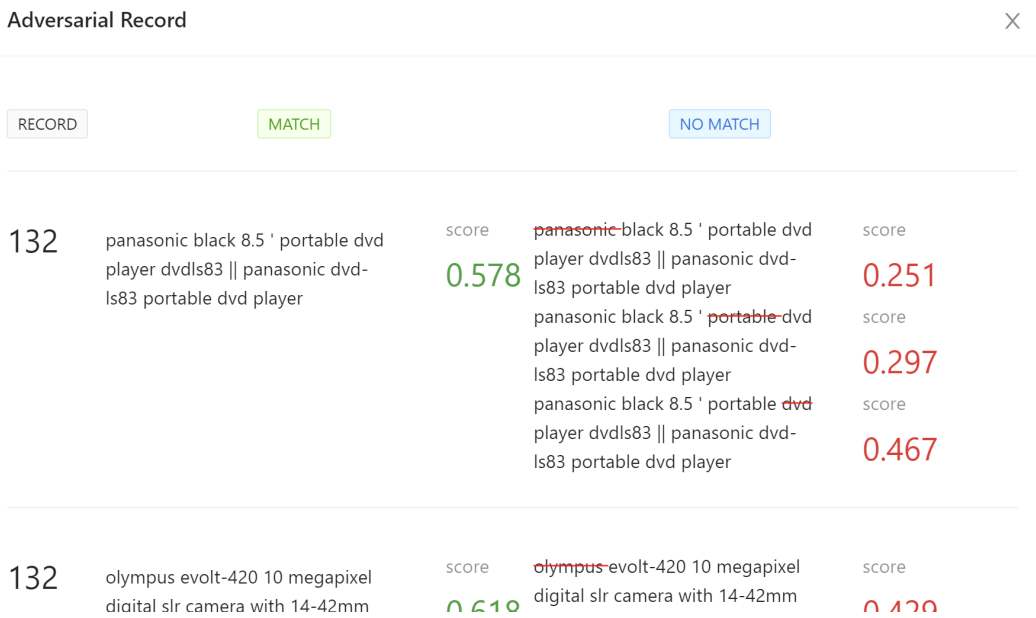
The Demo. In this demo, we show how Landmark Explanation supports users in understanding EM models by generating explanations of their predictions. The application of Landmark Explanation to real scenarios automatizes the expensive and time-consuming operations required to evaluate matching results, which are typically manually performed by domain experts. The analysis of explanations provides precise indications about matching errors and directions for improving the EM model. The demonstration will showcase how Landmark Explanation is able to (1) explain the behavior of an EM model through the explanation of predictions performed on specific records, (2) test the EM model reliability by producing *adversarial* examples, and (3) discovering *emblematic* explanations (e.g. by maximizing the low-entropy criterium, Landmark Explanation identifies the explanations with more unbalanced distribution in the term impacts. In this way, users can easily identify the elements that mostly contribute to the prediction).

3.3 Demo walkthrough

This demonstration (see https://youtu.be/btMDN7ii6Tg for a preview) shows that Landmark Explanation can (1) generate meaningful explanations of matching/non-matching predictions of EM models to support the understanding and validation of the matching policies; (2) find *adversarial* examples to evaluate the robustness of the EM model; (3) discover *emblematic*



(a) Explaining the EM prediction



(b) Adversarial examples

Fig. 3.3 EM prediction explanation workflow.

explanations according to some user's preference. Participants will be asked to upload an already trained model (DL or ML) along with a dataset to be explained (which must be consistent with the structure of the training set). The model must expose a predict method to

call in the reconstruction and prediction phase of the explanation process. In alternative, they can select amongst the Magellan ML models [4] and datasets[5] which are already preloaded into the tool. Currently, Landmark Explanation implements the perturbation technique provided by *LIME*, but it has been designed to be a flexible and extensible framework that can take advantage of new and more advanced perturbation techniques. It adopts a micro-kernel architecture, and plug-in components (as the ones implementing new perturbations) have only to implement a standard interface to extend the tool.

3.3.1 EM prediction explanation

The generation of a double explanation (one for each entity in the record), along with the token injection technique for increasing the expressiveness allows users to understand the importance assigned by the EM model to each token through the prediction.

Scenario 1. Figure 3.3 provides an example of an explanation created by Landmark Explanation. To generate it, a Logistic Regression model was trained on Magellan's Abt-Buy dataset. Then a record was randomly selected for its explanation. A non-matching pair of entities describing a digital camera (the left entity) and a leather case (the right entity) was selected. Landmark Explanation shows for each entity (the digital camera and the leather case labeled in turn as *landmark* entities) the impact of the relative tokens on the prediction. In particular, the plot on the left reports the importance of the tokens of the left description (digital camera) with respect to the right entity (the leather case). In the plot on the right, the opposite scenario is shown. A token has a positive impact when its presence positively contributes to the recognition of the pair of entities as a match. Tokens with negative impacts push the model's prediction towards no-match. For each explanation, tokens in red are the ones originally belonging to the entity, tokens in green are the one opportunely injected into it the explanation. In the example, the original red tokens that have the greatest impact for a match decision are `digital` and `camera` (this is confirmed by both explanations). Conversely, the most distinctive tokens are `dslra200w`, the code for the camera, and the `leather` token for the case. This conclusion is also confirmed by the analysis of the impacts of the injected tokens (in green): if the `leather` token was included in the description of the camera, the prediction score would have increased by about 0.1941; the increase would have been even more marked (0.2565) if the digital camera code `dslra200w` was injected into the description of the leather case. The analysis performed with Landmark Explanation can therefore confirm that the EM model adopts a correct behavior on this record. In particular,

[4] https://sites.google.com/site/anhaidgroup/projects/magellan/py_entitymatching
[5] https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

it's noticeable how the augmentation of tokens (the green ones in the Figure) provides a wider subject for the explanations and allows users to also understand how, by injecting tokens from a description to the other, the match score of the explained model increases.

3.3.2 Generating *adversarial* examples

Landmark Explanation supports the generation of *adversarial* examples, i.e. explanations obtained by altering a dataset entry to detect borderline behaviors of the model. A simple heuristic to generate these explanations is to produce a "normal" explanation of the EM model on the considered record and remove from the original record the tokens in descending order by impact. The EM model is applied over the altered records to generate new prediction scores, and these will eventually change the prediction class. The analysis of these borderline behaviors of the model can highlight anomalous behaviors whose resolution can improve the EM model.

Scenario 2. Figure 3.3b shows an *adversarial* example, generated again considering a Logistic regression model trained on Magellan's Abt-Buy dataset. The Figure shows a record describing a matching pair of entities. This is a DVD player represented through two textual representations. The EM model correctly recognizes this record as a matching element, however possible contradicting behaviors have been identified by perturbing some tokens in the dataset entry. For example, it can be observed that the removal of the `portable` token pushes the model towards a "non-matching" decision. The impact of the token in the prediction is about 0.28, i.e. its absence decreases the prediction from the initial score of 0.58 (match) to 0.30 (non-match). This can represent a possibly unexpected result for the user since the token can be considered as not discriminatory for a DVD player entity. Nevertheless, the explanation shows that the model relies on this token for its prediction. The explanations identify the most determinant tokens from the model perspective, and users can compare the model's behavior with their desiderata to evaluate its reliability.

3.3.3 Discovery of *emblematic* explanations

Landmark Explanation provides a functionality for automatically discovering *emblematic* explanations among the ones generated for the entire dataset. These are evaluated by considering how they jointly satisfy the following three metrics: (1) *explanation entropy*: an explanation with an unbalanced distribution of token impacts (i.e. low entropy) is considered more *interesting* than one with an almost-uniform distribution (i.e. high entropy). The reason is that low-entropy explanations allow users to identify more clearly the most relevant tokens for prediction. (2) *prediction-relevant tokens*: an explanation where a low percentage of

tokens are relevant for the prediction (i.e. the ones with minimum sets of tokens that if removed from the record cause the model to generate a different prediction) is preferable as it provides a more compact and usable interpretation of the model's behavior; (3) *explanation overlap*: to achieve a more exhaustive interpretation of the EM model's behavior, it is useful to analyze sets of explanations with complementary characteristics (e.g. explanations that have complementary token impact distributions at the attribute level) .

The user defines the weight of each metric in the discovery of the emblematic explanations and the *Smooth Local Search* technique [50] is applied to optimize the selection process. Figure 3.4 shows two examples of emblematic explanation, computed by introducing as user's preference criterium only the maximization (in the first image) and the minimization (in the second) of the explanation entropy. The first record is a false positive for the model. It is predicted as non-matching (the prediction score is less than 0.5), even if the label is 1. The token augmentation is applied: the same token is associated with the left and the right description. The second record is predicted as a match and there is no injection. Minimizing the entropy results in explanations, as the one in Figure 3.4a, where only few tokens strongly push for the matching decision (the ones with the highest score: `2005`, `tournament`, and `poker`). Conversely, maximizing the entropy let us obtain emblematic explanations like the one in Figure 3.4b, where the impact on the tokens is more uniformly distributed.

3.4 Conclusion

This paper introduces Landmark Explanation a tool that makes a post-hoc perturbation-based explainer able to deal with ML and DL models describing EM datasets. The approach has been experimented coupled with the LIME explainer, which is one of the most used state-of-the-art approaches. The results show that the explanations generated by Landmark Explanation outperform the ones generated by the competing approaches in accuracy. Moreover, the explanations generated by Landmark Explanation have been experimented to be "more interesting" for the users. Furthermore, it has been showcased how Landmark Explanation performs the interpretation of EM models through (1) the explanation of the behavior of EM models on single predictions, (2) the creation of *adversarial* examples, (3) the automatic selection of *emblematic* explanations.

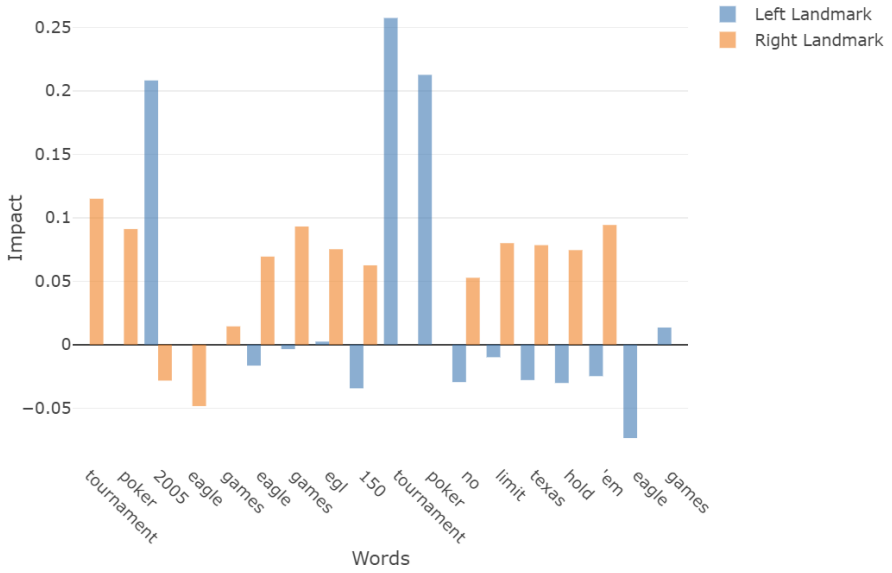
Future work includes the study of techniques for summarizing the explanations to facilitate the interpretation of the EM model as a whole.

While Landmark Explanation successfully provides post-hoc explanations, interpreting more complex and powerful deep learning models, such as transformers, remains challenging. Understanding the internal mechanisms of these models is crucial for further advancing

explainability in entity matching. Hence, in the next chapter, we deepen our investigation by inspecting how BERT—a prominent transformer-based architecture—internally performs Entity Matching tasks.

LEFT_TITLE	LEFT_MANUFAC...	RIGHT_TITLE	RIGHT_MANUF...	ENTROPY	PREDICTION	LABEL
tournament pok...	eagle games	eagle games egl ...	eagle games	0.9498	0.2375	1

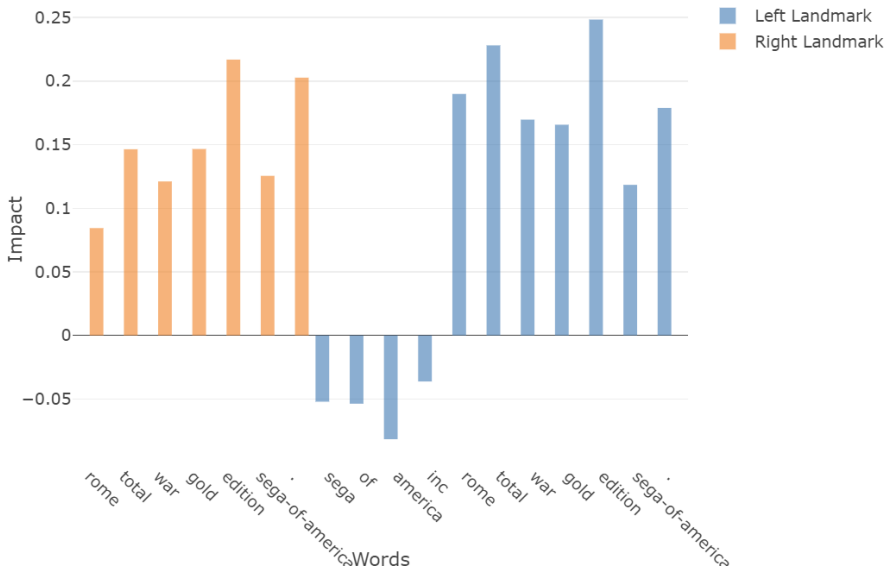
All Top 10



(a) Low-entropy explanation

LEFT_TITLE	LEFT_MANUFAC...	RIGHT_TITLE	RIGHT_MANUF...	ENTROPY	PREDICTION	LABEL
rome total war g...	sega-of-america...	sega of america i...	sega-of-america...	2.1849	0.899	1

All Top 10



(b) High-entropy explanation

Fig. 3.4 Emblematic explanations by varying entropy.

Chapter 4

Analyzing How BERT Performs Entity Matching

BERT, and more generally transformers, are black-box architectures and it is not easy to understand which are the internal mechanisms that allow them to obtain such outstanding results. Providing an answer to this question is crucial to increase their trustworthiness and promote their application in real-world scenarios.

The NLP research community recently put a big effort in analyzing which knowledge is learned and applied by transformer-based architectures. The term BERTology [112] was coined to refer to the large number of papers that have investigated BERT-based architectures [56, 106, 84, 128, 127, 83, 64, 34, 67, 77]. These analyses typically follow two main research directions: they directly evaluate the contribution of specific architecture components (such as contextualized embedding or attention modules) or they examine the parameters of probing classifiers trained on top of them.

Inspired by this line of work, this chapter proposes to inspect the ability of BERT-based approaches to perform EM. This operation is usually conceived as a binary classification problem, where the class shows if pairs of entities described by the dataset records are (or not) matching, i.e., they refer to the same real-world entity. The structure of the datasets makes this task far from the ones typically studied in ML and DL: EM datasets describe two evidences per record, whereas the records in ML and DL usually describe single evidences. It is worth examining how transformers manage this unique dataset structure and whether fine-tuning enables transformer components to learn matching logic from the data. Therefore, three research questions are formulated, providing methodological guidance to our research. They concern (1) the impact of fine-tuning on the effectiveness of the EM task; (2) the capability of BERT to detect and exploit the special structure of the EM datasets and (3) the extent to which BERT-based EM models rely on the semantic similarity of pairs of tokens.

Sections 4.2-4.4 propose deep experimental evaluations that provide answers to the aforementioned questions. Their overall analysis allows us to get three main findings (Section 4.5): 1) BERT architectures fine-tuned on the EM task are more efficient than ad-hoc EM models. Fine-tuning impacts the last layers of the attention modules and modifies the space of the embeddings differently depending on whether the records refer to matching/non-matching entity descriptions. 2) The attention weights are consistent with the structure of the EM datasets, and recognize that records describe pairs of entities through the same set of attributes. A special pattern is found in the attention modules that gives attention to attributes describing the same entity property in different entity descriptions. Moreover, BERT is able to discover the attributes which mostly contribute to solving the EM task. 3) The pair-wise semantic similarity of tokens is not particularly exploited by the model. BERT seems to introduce and use a more contextualized, pragmatic kind of knowledge that involves more tokens and attributes. The importance of the one-to-one relationships defined by the semantic similarity decreases with the fine-tuning process and, at the same time, increases the importance given to the EM structure.

The chapter synthesizes the complete set of experiments introduced in the technical report [88], to which interested readers can refer. A GitHub with the datasets used in the experiments and the code is available [88].

The rest of the chapter is organized as follows. Section 4.1 defines the boundaries of our experimental evaluation that results in an analysis of the impact of fine-tuning on the effectiveness (Section 4.2); of what BERT learns from the dataset structure (Section 4.3); and the importance of the pair-wise semantic similarity of tokens in the EM process (Section 4.4). Section 4.5 points out some lessons learned, and Section 4.6 sketches out some conclusions and future work.

4.1 The Experimental Analysis

Methodology. Answering the following three research questions can lead to understanding how BERT-based models support EM, what knowledge is learned through the tuning process, and how this knowledge improves the matching process.

1. To what extent and for what reasons fine-tuning is able to improve the effectiveness of the results achieved by BERT-based EM models? (Section 4.2)

2. Does BERT detect and exploit through the fine-tuning the specific structure of the EM datasets composed of pairs of entity descriptions, sharing the same set of attributes? (Section 4.3)

Dataset	Type	Datasets	Size	% Match	Sample Size	# Attr
S-FZ	Structured	Fodors-Zagats	946	11.63	132	6
S-DG		DBLP-GoogleScholar	28,707	18.63	6414	4
S-DA		DBLP-ACM	12,363	17.96	2664	4
S-AG		Amazon-Google	11,460	10.18	1398	3
S-WA		Walmart-Amazon	10,242	9.39	1152	5
S-BR		BeerAdvo-RateBeer	450	15.11	80	4
S-IA		iTunes-Amazon	539	24.49	156	8
T-AB	Textual	Abt-Buy	9,575	10.74	1232	3
D-IA	Dirty	iTunes-Amazon	539	24.49	156	8
D-DA		DBLP-ACM	12,363	17.96	2664	4
D-DG		DBLP-GoogleScholar	28,707	18.63	6414	4
D-WA		Walmart-Amazon	10,242	9.39	1152	5

Table 4.1 Magellan Benchmark

3. How much does BERT rely on pair-wise semantic similarity of tokens, how this knowledge changes with the fine-tuning process? To what extent does this similarity support the EM process? (Section 4.4)

Datasets. The experiments are performed against the datasets provided by the Magellan library[1] which is the reference benchmark for the evaluation of EM tasks. The datasets describe pairs of entity descriptions sharing a common structure.

Table 4.1 summarizes some statistical measures describing the datasets, reporting for each of them the total number of records (fourth column) and the percentage of records associated to a match label (fifth column). In the experiments that require to train and evaluate the effectiveness of BERT in performing EM, the datasets are divided into train, validation and test sets with a proportion of 60, 20, 20. The remaining experiments, which apply a-posteriori analyses of BERT components, are instead performed on random samples of records, with a size depending on the dataset as reported in column *Sample Size* of Table 4.1. The samples are balanced, including the same number of matching and non-matching entity descriptions. The experiments were all repeated three times and the average value is reported in this chapter. In this chapter, the reference BERT variant is the bert-base-uncased, which consists of 12 layers, each having a hidden size of 768 and 12 attention heads (110M parameters).

Dimensions of the Analysis The experimental evaluations are performed along 3 main dimensions, (1) data encoding, (2) data unit representation, and (3) model application. Two techniques are tested for *encoding the data*. *Sentence-pair (SP)* consists of supplying BERT with two distinct phrases (separated by the special token [SEP]), where each phrase corresponds to the textual representation of an entity description obtained by concatenating

[1]https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

all attribute values. The second, *attribute-pair (AP)*, modifies the previous approach by using the special token [SEP] to delimit the content of the attributes. In this way, BERT is aware of the subdivision of information by attributes existing in the datasets. A similar encoding has also been adopted in [81]. The experiments are performed with different *granularities for the data representation*. Some tests evaluated the attention given to *tokens (TK)*. In other tests, the scores are aggregated by the attribute they belong to. Two techniques for *representing the attention for attributes* are experimented: by considering the *average (AV)* of the attention given to their composing tokens or the *maximum value (MA)*. Finally, concerning the *model*, the experiments are performed with both a *pre-trained (PT)* and a *fine-tuned (FT) BERT model*. The architecture of the pre-trained model is composed of two fully connected layers with 100 and 2 neurons respectively (where the 2 output neurons represent the match and non-match classes) added on the top of the original BERT's language model[2]. These additional layers have been trained on the EM task to predict whether pairs of input entities are matching, by keeping unaltered the BERT's original pre-trained model. The fine-tuned architecture consists of a single classification layer inserted on top of the embedding corresponding to the [CLS] token. This is the usual standard practice adopted for fine-tuning BERT to a downstream classification task [81, 25]. The whole architecture is here trained on the EM task, thus modifying the weights of the attention modules and the consequent embeddings of the original BERT model.

An *experiment setting* is a proper selection of the dimensions of the analysis, namely $setting = (DE, AR, MO)$, where DE is one of the techniques implemented for data encoding (SP or AP), AR for the attribute representation (AV or MA) and MO for the model application (PT or FT).

Data Structures. The analysis of the attention modules relies on two special data structures that show the attention provided by the BERT-based architecture. The *attention head* [39] is a squared matrix with cells showing the attention scores that tokens in the rows give to the token in the columns. The BERT architecture consists of 12 layers each of which contains 12 heads, for a total of 144 attention heads. In the *attribute attention head* the attention scores are aggregated by attribute according to one of the techniques introduced (average or max value). Since the EM dataset describes pairs of entities, attention matrices can be decomposed into four quadrants (see for example Figure 4.4). The top-left quadrant

[2]The application of the BERT model to a record generates a 768-dimensional embedding for each constituting token. A 768-dimensional vector is then generated for each entity in the description by averaging (across the last 4 layers) the embeddings of their associated tokens. A difference vector is then calculated by subtracting the representation of the first and second entity and supplied as input to the fully connected layer. The 768-dimensional vector is then compressed into a 100-dimensional representation and reduced via a softmax layer to a matching / non-matching probability score.

shows the attention given to the attributes of the first entity from the attributes of the same entity. The bottom-right quadrant describes the same information for the second entity. The bottom-left quadrant shows the attention given to the attributes of the first entity by the ones of the second entity and the top-right quadrant the opposite score: the attention to the first entity from the second one. Note that both the attention and the attribute attention matrices can be aggregated per layer (by averaging the values in all the heads) and per dataset (by averaging the attention data structures across all the records of a dataset).

Limitations. The pre-trained and fine-tuned EM models proposed are one of the simplest and most minimal architectures possible based on the BERT model. This increases the generality and applicability of the findings obtained to all BERT-based architectures (e.g., *Ditto* [81] and [25]). Nevertheless, the analysis is affected by similar limitations as other works in the literature sharing the same methodology. Concerns have been raised about the methodology of inspecting individual components of such complex architectures. Studies have shown that the knowledge acquired by these models is spread throughout the entire architecture and an analysis of the individual components may not be sufficient [77]. In particular, the analysis of the role of attention modules in complex models has recently been discussed. [70, 118, 24] discovered that limited correlations exist between attention weights and the predictions of the model. This thesis is further exacerbated by the fact that in recent transformer architectures these modules are followed also by several non-linear transformations. Nevertheless, this is a controversial point since other papers, as [141, 134], demonstrated that low correlations happen only in limited conditions.

4.2 Impact of the fine-tuning on the EM task

The goal of this Section is to evaluate to what extent and in which way the fine-tuning process improves the ability of a BERT-based model to perform EM tasks. The first experiment proposed in Section 4.2.1 evaluates the effectiveness of pre-trained and fine-tuned BERT EM models by evaluating both the data encodings proposed for the entity descriptions. The fine-tuning process determines a change in the attention modules and on the generated embeddings. The experiment in Section 4.2.2 examines the impact of fine-tuning on the attention modules by comparing the attention weights before and after the process. Lastly, Section 4.2.3 assesses whether fine-tuning affects the embeddings of matching and non-matching word pairs.

	Pre-trained (attr-pair)	Pre-trained (sent-pair)	Fine-tuned (attr-pair)	Fine-tuned (sent-pair)	DM+	Ditto
S-FZ	97.67	97.67	100.00	97.67	100.00	100.00
S-DG	92.80	92.40	94.92	94.78	94.70	95.80
S-DA	97.52	97.41	98.42	98.65	98.45	99.17
S-AG	65.19	63.26	70.21	68.52	70.70	75.58
S-WA	54.81	59.89	79.79	78.85	73.60	86.76
S-BR	82.76	82.76	77.78	84.85	78.80	94.37
S-IA	86.21	85.19	90.00	93.10	91.20	97.80
T-AB	62.35	59.50	81.42	83.51	62.80	89.79
D-IA	70.59	84.21	94.74	94.74	79.40	95.65
D-DA	96.85	96.10	98.43	98.42	98.10	99.08
D-DG	91.63	92.27	95.07	94.77	93.80	95.75
D-WA	56.60	50.76	79.59	77.33	53.80	85.69

Table 4.2 The effectiveness of pre-trained and fine-tuned BERT-based models: $settings = (SP/AP, -, PT/FT)$ (F1 score).

4.2.1 Effectiveness

Implementation. BERT's capability to address EM tasks is evaluated by incorporating a binary on top of its modules, as described in Section 4.1. Four different settings are tested by varying the data encoding and the model application, i.e. $settings = (SP/AP, -, PT/FT)$.

The results of the experiment are shown in Table 4.2, and compared with *DeepMatcher+* (*DM+*)[3] [93], a reference DL-based EM approach that does not rely on a transformer architecture, and *Ditto* [81], one of the best BERT- based EM approaches.

Discussion. Even if *DM+* obtains good results in most the datasets, the techniques based on BERT outperform them in particular in the resolution of textual and dirty datasets (i.e., the ones with a higher percentage of missing values and misalignment between attributes, identified with the prefix T and D in the Table). This result is consistent with the literature [25]. Moreover, *Ditto*, which extends BERT with data augmentation techniques and advanced EM data encoding, further improves the results. Finally, data encoding (attribute or sentence pair) does not have on average a real impact on the results.

4.2.2 Attention

Implementation. Motivated by investigating the reasons that make the fine-tuned architecture so effective on the EM task, an evaluation is conducted to observe how the attention weights of a pre-trained BERT architecture change after the fine-tuning. The experiment follows an adapted version of the methodological procedure that was applied in [77] to perform NLP

[3] Table 4.2, considers the results published in [81].

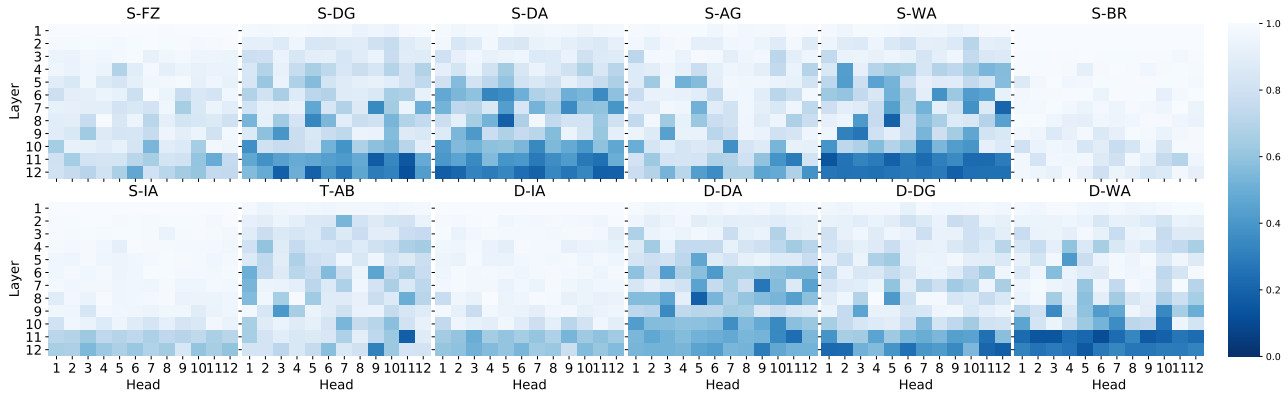


Fig. 4.1 Similarity between pre-trained and fine-tuned attention scores, $settings = (SP, -, PT/FT)$. The darker the cell, the greater the difference between the attention scores of fine-tuned and pre-trained models.

tasks. The cosine similarity between (flattened versions of) the attention heads associated to the pre-trained and fine-tuned models is computed for every head and layer of each dataset record. The average of these similarities for all the records in the dataset is shown in Figure 4.1 for the $settings = (SP, -, PT/FT)$.

Discussion. The heads that undergo the greatest variations are those located in the last layers (i.e. the overall similarity of the last layers is generally closer to 0). This is particularly evident for the structured and dirty versions of DG, DA and WA. This result suggests that more EM-specific information is encoded in the last layers, while shallow layers capture more general linguistic information mainly deriving from the pre-train. This finding is consistent with similar experiments performed in other NLP scenarios, as described in [77, 61, 60, 112].

4.2.3 Embeddings

Implementation. The previous experiment are complemented by analyzing how the fine-tuning alters the space of the pre-trained embeddings. It is expected that the model's improved performance on the EM task will be reflected in the distribution of the embeddings of the words belonging to matching/non-matching pairs. The hypothesis is that the fine-tuned model increases the similarity of the embeddings of the words appearing in matching pairs and vice versa decreases the one of words occurring only in non-matching pairs. To analyze the validity of this consideration, a sample of 1000 pairs of random words is selected (where the first word derives from the left entity and the second from the right one) that occur exclusively in matching, non-matching records. An analysis is also performed on random records to provide a baseline. The (cosine) similarity of their embeddings was then

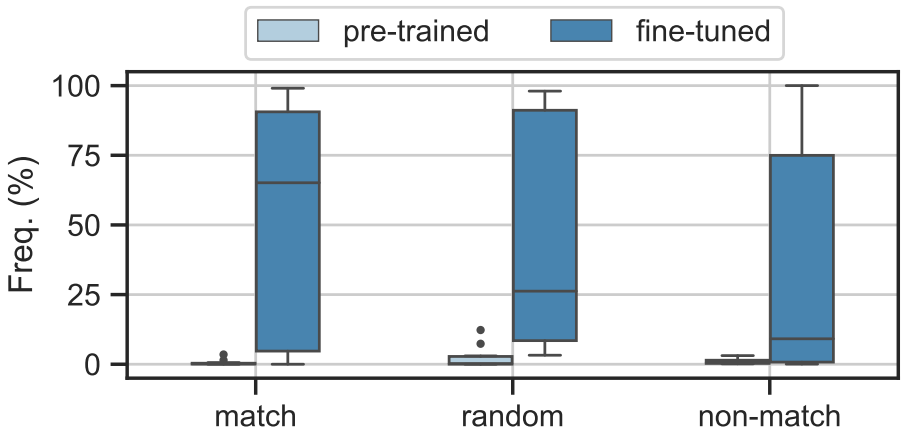


Fig. 4.2 Impact of the fine-tuning on the similarity of the embeddings: $settings = (SP, -, PT/FT)$. The y-axis shows frequency of the similar embeddings in the entity descriptions.

calculated, and the percentage of instances where this similarity exceeded 0.7 was evaluated. This threshold was chosen to indicate words with medium-high similarity. The results of this experiment for the $settings = (SP, -, PT/FT)$ are shown in Figure 4.2.

Discussion. Firstly, it is observed how the fine-tuning process increases the similarity between all pairs of tokens examined compared to the pre-trained version. Secondly, the similarity between the pairs of tokens belonging to records describing matching entities is on average higher than those of the tokens occurring in non-matching and random records. At the same time, the similarity associated with non-matching pairs is lower than the other categories of records. This is because the number of similar words in descriptions of non-matching entities is generally less than in matching entities. Despite this, there is a high variability in the results: there are pairs of tokens with a high similarity regardless the fact they belong to descriptions of matching or non-matching entities.

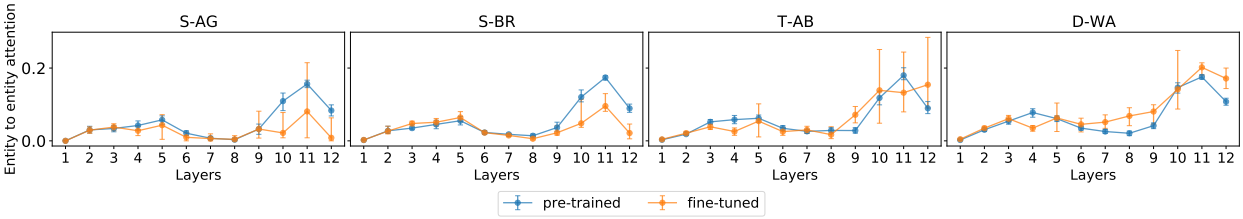
4.3 Relying on the dataset structure

The experiments in this Section aim to evaluate if the fine-tuning process detects and exploits the special data structure adopted by the EM datasets. In particular, the experiment in Section 4.3.1 investigates whether and to what extent the fine-tuned BERT model exploits the relationships between the pairs of entities that appear in each EM data entry. Therefore the attention given to the pairs of tokens belonging to two different entity descriptions in the same EM record is analyzed and the changes determined by the fine-tuning is evaluated. The

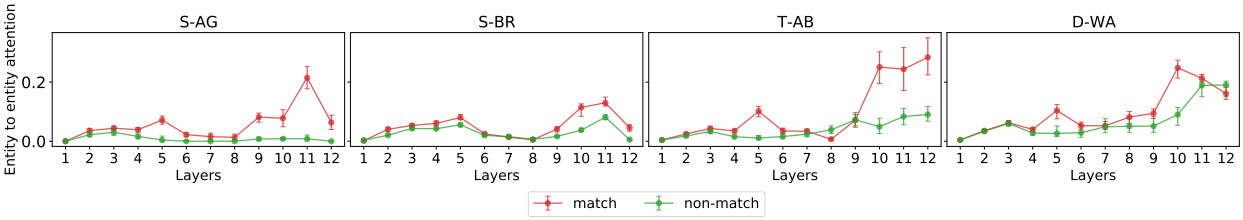
experiments described in Section 4.3.2 study the presence of frequently occurring patterns in the BERT's attention modules. In particular, the analysis focus on the patterns that show relationships between attributes. The experiments in Section 4.3.3 evaluate wheater the attention provided by the pre-trained and fine-tuned BERT models reflects a variable contribution of attributes in performing the EM task.

4.3.1 The entity-to-entity (E2E) pattern

Implementation. The goal is to discover if there is attention between the pairs of entities described in the same record. This is expected to be a frequent pattern in EM datasets where the task is to identify the correspondences between the tokens of two entities belonging to the record. The idea is to understand: 1) the contribution of this pattern in the attention generated by BERT, and 2) which are the layers where the pattern is mainly active. To perform the experiment, an average attention head is built for each layer by averaging the scores of all its heads. Then, for each average attention head, the percentage of cells that refer to tokens from descriptions of different entities and have an attention score within the last quintile of the matrix is calculated. The scores are then averaged considering the dataset records.



(a) Pre-trained vs fine-tuned models, $setting = (SP, -, PT/FT)$.



(b) Matching vs. non-matching entities, $setting = (SP, -, FT)$.

Fig. 4.3 Entity-to-entity attention. The y-axis reports the normalized number of times where pairs of tokens from descriptions of different entities are assigned an attention score in the last quintile of an average attention head calculated on a certain layer.

Discussion. Figure 4.3a shows the entity-to-entity attention pattern generated by the pre-trained and fine-tuned models on a selection of the datasets. Similar results are obtained with the other settings (the technical report [88] provides details of this and the following

experiments). First of all, the entity-to-entity attention pattern contributes to a maximum of 30% on the entire attention produced by BERT. Then, this pattern mainly occurs in the last 3 layers of the architecture, suggesting that BERT uses these layers to encode "cross-entity" information. This trend is confirmed by both the pre-trained and fine-tuned models, which generate also very similar absolute values, suggesting that this behavior is inherited almost exclusively from the initial training of the architecture. The impact of the fine-tuning on the pattern largely depends on the dataset. In some cases, the entity-to-entity attention pattern is more marked in fine-tuned models, in other cases on the pre-trained. Nevertheless, it is worth to note that the interquartile range markedly increases with the layer in almost all datasets and especially for the fine-tuned models. In Figure 4.3b let us inspect the variability introduced by the fine-tuning by comparing the intensity of the pattern for records referring to matching and non-matching entities. The diagrams show that the high variability is due to a diversified contribution to the E2E pattern from matching/non-matching records. This therefore suggests that the fine-tuned model learned to distribute attention in a different way according to the type of record analyzed.

4.3.2 The attention patterns involving the dataset attributes

Implementation. The goal of this experiment is to observe if there are frequently occurring patterns in the BERT's attention modules when applied to EM tasks. A special matrix is introduced with the aim of providing a compact and clear representation of the most expressive patterns for a dataset. The matrix is built upon the attribute attention heads, where the actual values are substituted by a boolean mask showing the pairs of attributes measuring an attention score above the average. There is a boolean mask for each record, head and layer. These masks are averaged over the 12x12 grid to generate a single mask for each record and then the masks are further averaged across all records.

Discussion. Figure 4.4 shows the average boolean mask for all datasets with the $setting = (SP, MA, FT)$. Similar results are obtained with the other settings [88]. The visual inspection of the matrices shows the existence of four main occurring patterns: (1) **diagonal**: high scores on the main diagonal (and close elements) of the matrices are obtained. This means that an attribute gives high attention towards itself and neighboring attributes. (2) **vertical**: vertical lines show that all attributes have high attention towards the same target attribute. (3) **diagonal+vertical**: the diagonal and vertical patterns jointly occur in almost all datasets. (4) **matching attribute attention (MAA)**: there are elements with high scores in the main diagonals of the bottom-left and top-right quadrants composing each matrix. The first three patterns have been already observed as frequently occurring in other experiments concerning

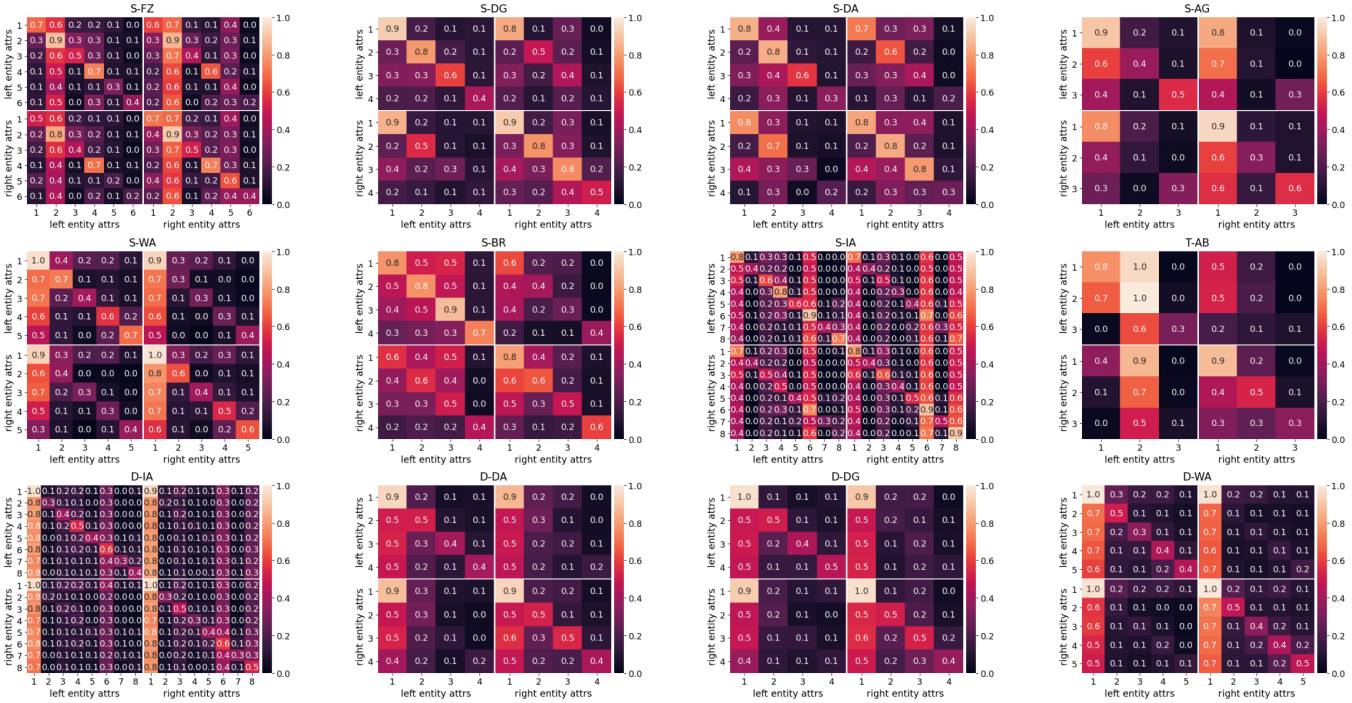


Fig. 4.4 Attention between attributes: $setting = (SP, MA, PT)$. The cells show the attention given by the attribute on the row to the attribute on the column, divided by left and right entity. The lighter the color, the higher the attention.

the analysis of NLP tasks [77]. The MAA is a new pattern emerging in the EM scenario: an attribute gives high attention towards its corresponding attribute (i.e., the matching attribute) of the other entity. This is because, by construction, the entity descriptions share a common schema and the matching attributes have the same relative positions in the dataset (i.e., dividing the attributes of an EM entry in two ordered set, one for each entity, matching attributes share the same position in both the sets). Figure 4.5 shows the attribute attention head located in layer 11 and head 11 for the S-FZ dataset, where the MAA pattern is strongly visible.

To provide a measure of the consistency of the patterns in the datasets, the frequency of the vertical, diagonal and matching patterns is evaluated in all experimental settings. In particular, given a layer and head for a specific setting, a vertical pattern is considered as existing if there is a column having all values greater than the average of the attribute attention head; a diagonal pattern as existing if the average scores of the elements in the main diagonal are greater than the other diagonals; and, a MAA pattern if the average scores of the elements in the main diagonal of the top-right or bottom-left sub-matrices are greater than the other diagonals in the same sub-matrix. Figure 4.6 shows the percentage of the attribute attention heads where the patterns are found. The experiment shows that data encoding does

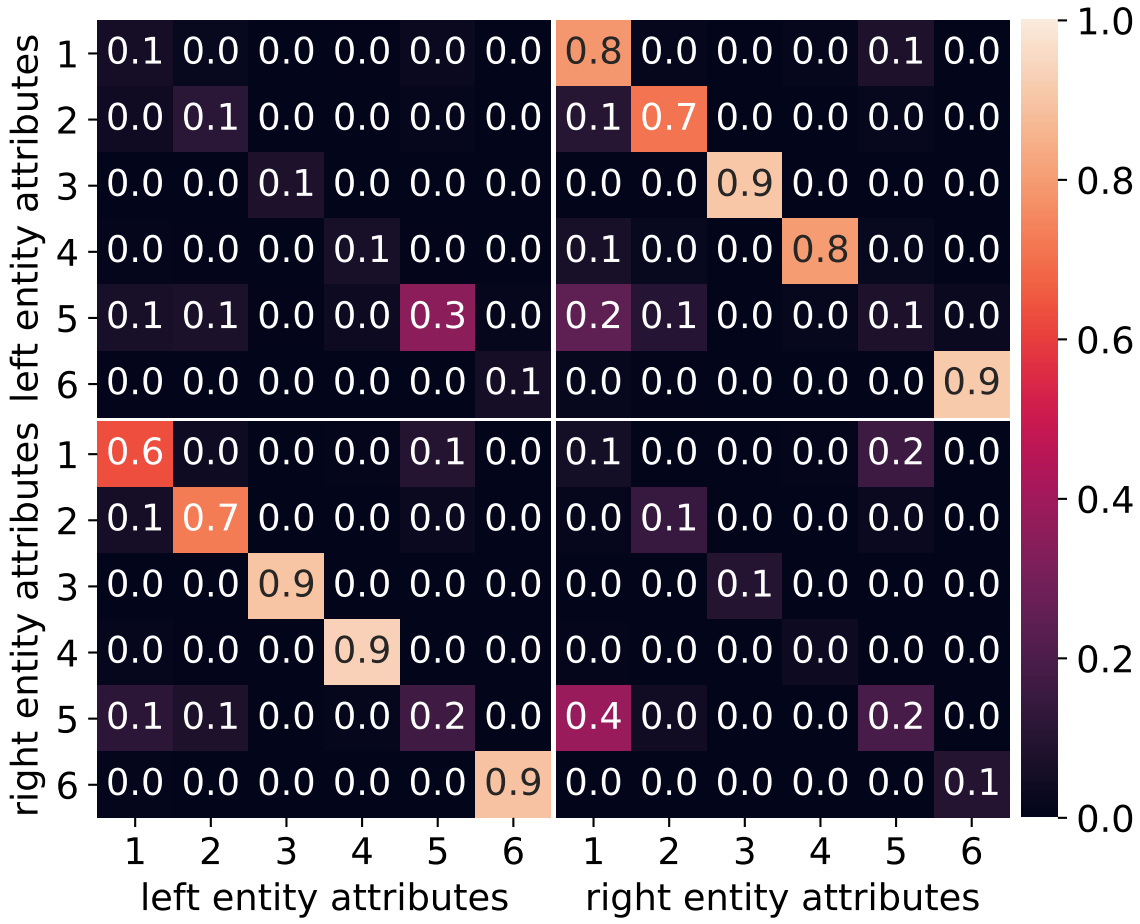


Fig. 4.5 The MAA pattern: head 11 - layer 11 of S-FZ.

not largely affect the results, and that the diagonal is the most common pattern. This is somewhat expected: this means that the terms give attention to themselves and to the other terms in the same attribute. The new MAA pattern is the second frequently occurring pattern in all datasets. This means that the structure of the EM dataset is recognized by BERT and preserved with the fine-tuning. Moreover, the dirty versions of the datasets (the ones with prefix D) show a reduced frequency of this pattern with respect to their structured versions, due to the misalignment of the values.

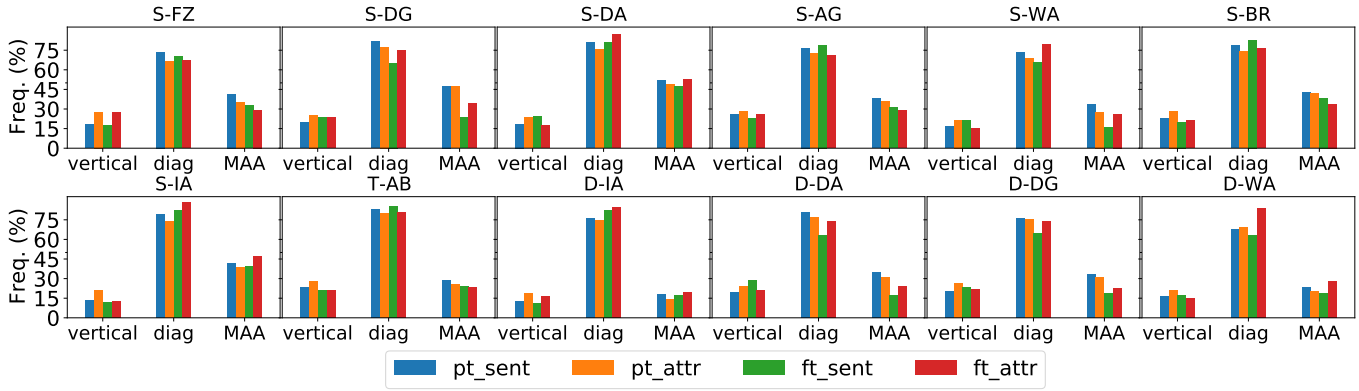


Fig. 4.6 Comparison of pattern frequency in all the settings $(AP/SP, MA, PT/FT)$. The bars shows the percentage of the attribute attention heads where the patterns are found.

The MAA pattern.

A deeper analysis of the MAA pattern is performed by analyzing its localization in the BERT layers and evaluating its contribution to the effectiveness of the model.

Localization. Figure 4.7 shows how the frequency of the MAA pattern varies across the layers of the architecture (the $setting = (SP, MA, PT/FT)$ is reported). It can be observed that the fine-tuning process leads to a reduction of the occurrences of the MAA pattern, in particular in the last three layers. This appears as a sort of counter-intuitive result: fine-tuning may be expected to introduce attention to the matching attributes. Nevertheless, the results of the experiment in the next section shows that this knowledge is crucial for the effectiveness of the model.

Impact of the MAA pattern on the effectiveness. Although previous experiments showed that the MAA pattern occurs less frequently in the fine-tuned model than in the pre-trained version, below the goal is to understand whether and to what extent it contributes to the capability of the model to perform the EM task. To carry out the experiment, firstly the

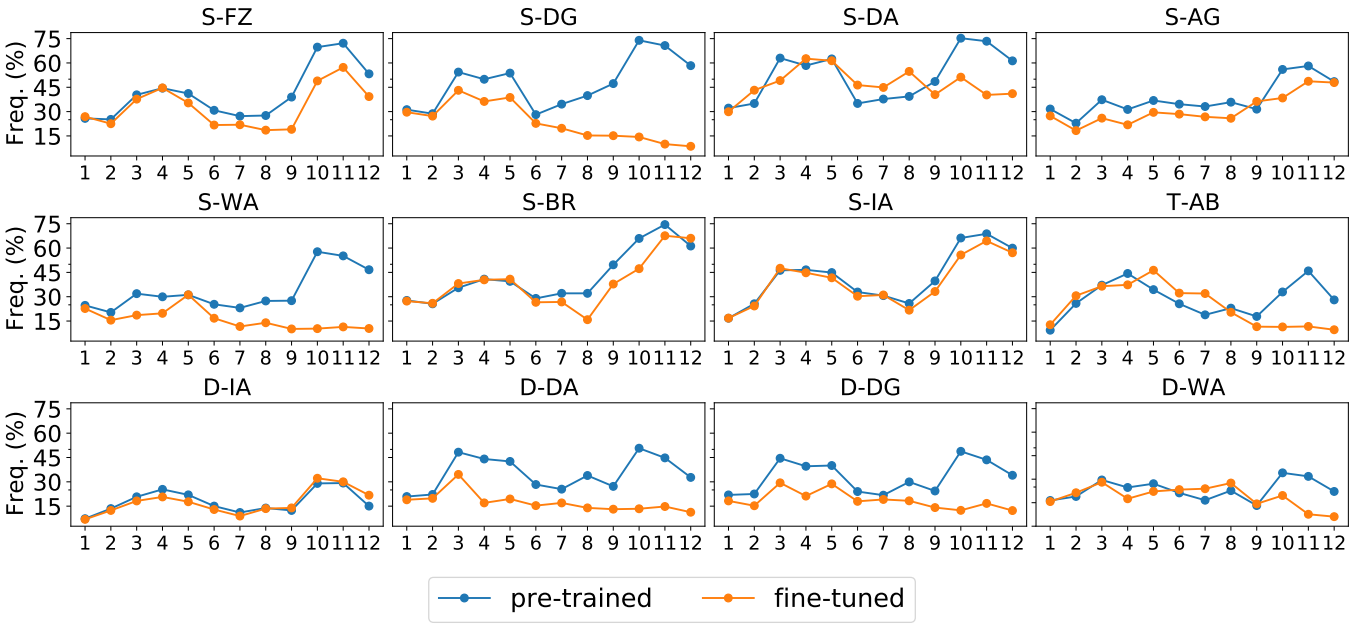


Fig. 4.7 Frequency of the MAA pattern: $setting = (SP, MA, PT/FT)$. The diagrams show per layer the percentage of attribute attention heads where the pattern is found.

average frequency of occurrence of the MAA pattern in the attribute attention heads is calculated. Then a number of heads are removed in descending order with respect to the pattern frequency, and the variations of F1 score of the fine-tuned model is evaluated. These results are compared with 2 baselines. The first (random) consists in pruning the same number of randomly selected heads. In the second baseline (importance) the same number of heads is pruned, but according to their importance (as defined in [91]). In the experiments, an increasing number of heads are removed (5, 10, 20 and 50) and the effectiveness of the model is evaluated. Note that the pruning reduces the overall number of parameters of the BERT architecture from $108.5M$ to $99.6M$. The results of this experiment are reported in Figure 4.8 in the $setting = (SP, MA, FT)$.

It is noticeable that the masking techniques generate a diversified impact on the performance of the model: while the random heads' removal does not determine substantial variations in the F1 score, the other techniques generate substantial reductions in the effectiveness of the model. This is particularly evident in the S-DG, S-DA, S-BR, T-AB, D-DA and D-WA datasets, where F1 scores close to zero are obtained. In these scenarios, despite some fluctuations, it is possible to observe how the removal of heads with the highest frequency of occurrence of the MAA pattern produces a more drastic reduction in performance compared to the two considered baselines. In many cases this behavior is noticeable even after removing only 5 heads. This result could provide a justification for the previous open

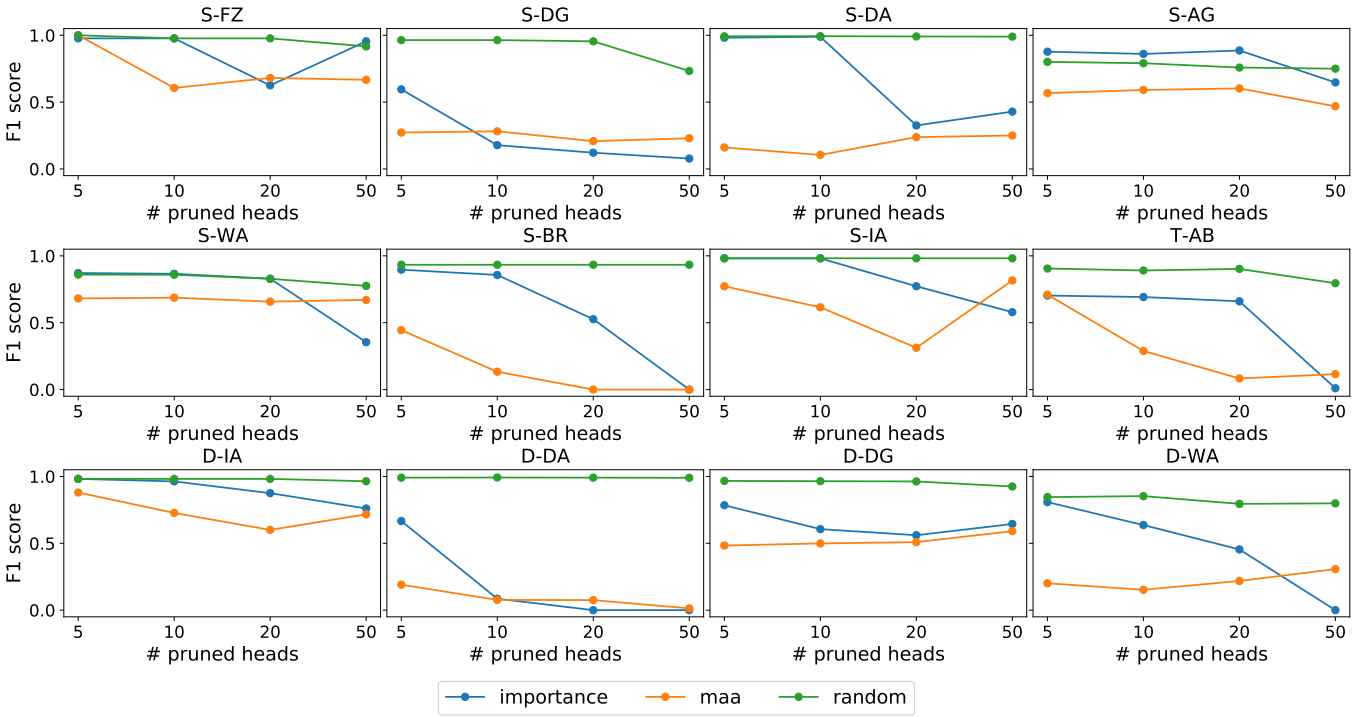


Fig. 4.8 Impact of the MAA pattern on the effectiveness (F1-score) of the EM task performed with fine-tuned BERT models.

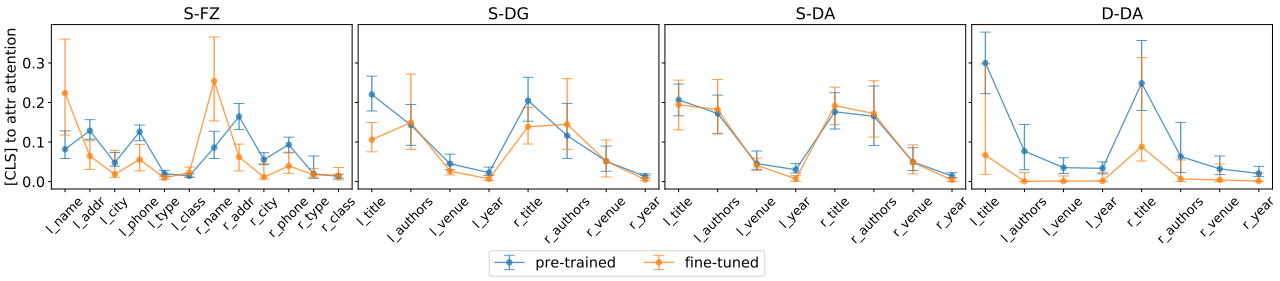
problem: the fine-tuned model reduces the number of heads exhibiting the MAA pattern, but the information encoded within these heads is more largely used by the model to solve the EM task.

4.3.3 Importance of the attributes

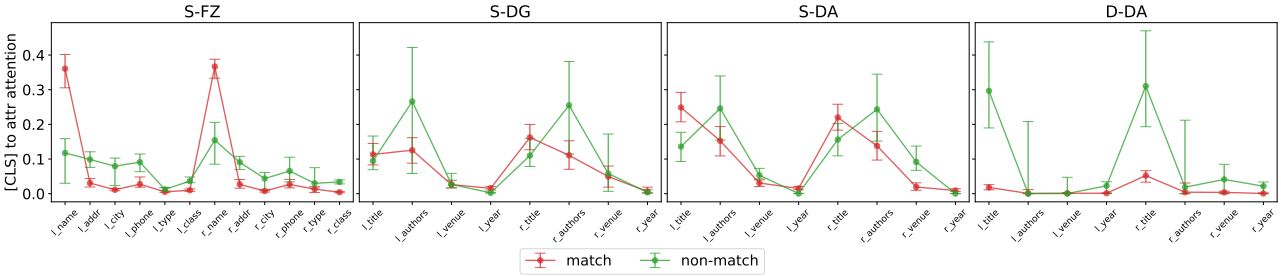
This Section aims to investigate if BERT can recognize that not all the attributes have the same importance in performing the EM task.

Analysis of the attention

Implementation. Through this experiment, the attributes of the dataset on which BERT relies when performing the EM task are evaluated by inspecting the attention modules. To answer this question, the attention given by the special [CLS] token to the other tokens constituting the attributes of the EM dataset is analyzed. The [CLS] token is a special token that is added by BERT to each input and is typically used to compute the prediction of any classification task. For this reason, the embedding associated to the [CLS] token is considered as a summary



(a) Pre-trained vs fine-tuned models, $setting = (SP, MA, PT/FT)$.



(b) Matching vs. non-matching entities, $setting = (SP, MA, FT)$.

Fig. 4.9 Attention given by the token [CLS] to the dataset attributes. The diagram shows the average value computed on all dataset entries of the attention registered in the last layer of the attention heads.

of the input sentence. The experiment considers only the last layer of the BERT architecture. The values on the attribute attention heads are averaged, and the attention values of the [CLS] token towards the attributes are selected. Then the aggregated values by averaging the attention scores for all the dataset entries are computed. The evaluation is performed with the $settings = (SP, MA, PT/FT)$ and Figure 4.9a reports the results for a selection of the datasets. Similar results are obtained in all settings and datasets [88].

Discussion. The pre-trained and fine-tuned models generate similar scores. This represents an unexpected result as the embeddings associated to the [CLS] tokens of fine-tuned models should be different from the ones of pre-trained models, since encoding task-specific information. The coarse-grained aggregation applied to attention scores that refer to the same attribute could be the reason for such a result. Nevertheless, the importance scores are consistently assigned to the attributes that, according to our domain knowledge, better allow users to identify matching entities. Moreover, in all datasets the attention generated towards the entity descriptions is symmetric (i.e., the attention is not focused on (attributes of) one of the two entities). Finally, it can be observed that the attention scores for the structured and dirty version of the DA dataset are diversified: on the dirty dataset the attention is exclusively towards one attribute; while on the structured version to multiple attributes. To elaborate on

the analysis, Figure 4.9b shows the attention scores of the fine-tuned model differentiated between matching and non-matching entities. The scores are diversified and it is not possible to observe if there is more attention on records referring to matching/non-matching entities. However, in some cases, the attributes receiving more attention change when considering matching/non-matching entities. This is the case of the S-DG and S-DA datasets, where non-matching records give high importance to the attribute describing the authors of the publication, and matching records to the title.

Gradient analysis

Implementation. The previous experiment showed how the attention scores associated with the attributes are consistent with the human evaluation. However, the experiment does not reveal whether the knowledge of the attribute importance is actually used in the inference of the matching decision. This experiment provides an answer to this question by analyzing the gradient of the attributes with respect to the predictions of the fine-tuned BERT model. The gradient of a function output with respect to its input variable provides a measure of its contribution to the result. It represents the typical attribution method applied on neural architectures, which do not provide explicit features of importance (such as the coefficients of a linear model), to determine the impact of single components on model predictions. In the experiment, firstly a balanced sample of 50 records is selected for the matching and non-matching classes. Then, the integrated gradient [126] of all tokens is computed. The gradient of each dataset attribute is considered as the maximum gradient measured among its constituent tokens. The results of the experiment are shown in Figure 4.10.

Discussion. The observed results are consistent with those obtained in the previous experiment: the majority of the datasets rely on the tokens belonging to the first attribute to generate the prediction. It should be noted that, by construction, the first attribute of each dataset contains the most discriminative information for the entities (e.g., the attribute title is the first attribute in dataset S-DG).

4.4 Exploitation of the semantic similarity knowledge

The experiments in this Section allow us to understand if BERT introduces some semantic knowledge in the attention heads and embeddings to be used for identifying similar tokens thus supporting the EM task. For performing the experiments, semantically similar pairs of terms are identified by exploiting the cosine similarity of the token embeddings generated

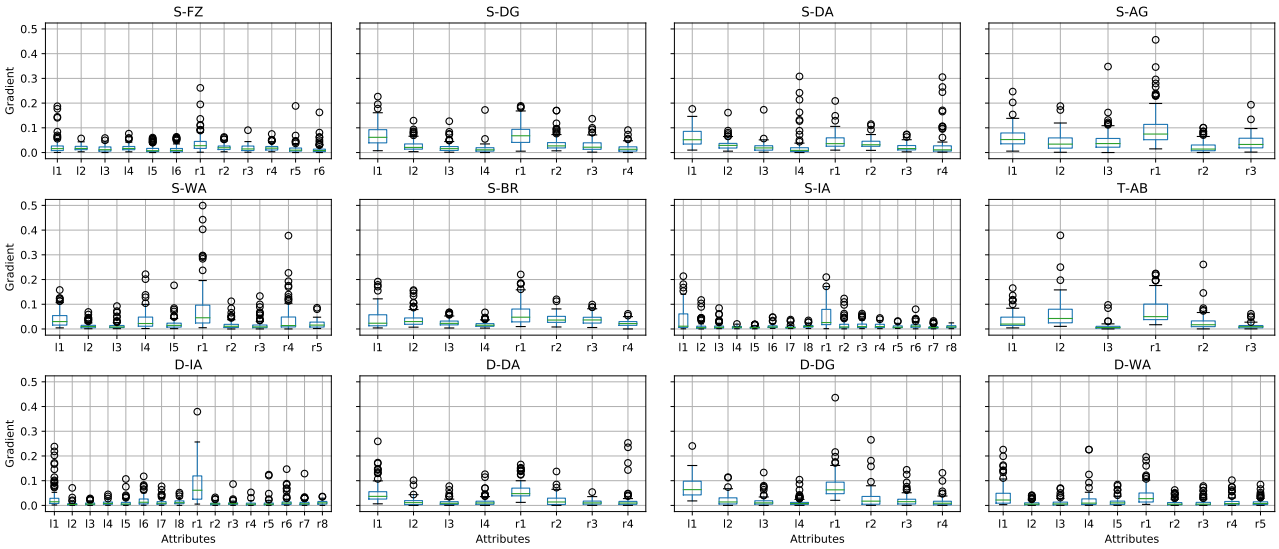


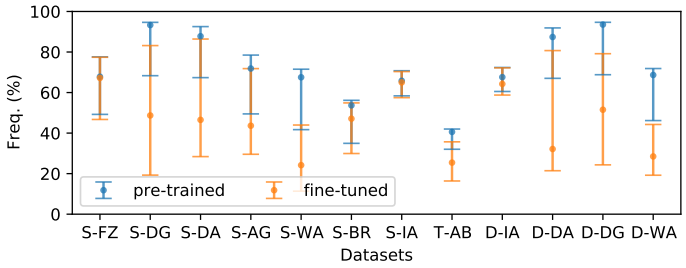
Fig. 4.10 Importance given to the attribute computed with the gradient analysis. The highest the value, the highest the importance of the attribute for the prediction.

with the fastText model [21] and an analysis is conducted on how BERT processes these inputs. Section 4.4.1 examines if BERT exploits semantic knowledge by evaluating the percentage of similar pairs found in the token pairs with the highest attention and how this amount changes with the fine-tuning. Section 4.4.2 analyzes the correlation between the cosine similarity of the embeddings generated with the fastText model with the ones generated with BERT (pre-trained and fine-tuned). Finally, Section 4.4.3 performs a gradient analysis to evaluate the contribution of the semantic relationships on the inferences.

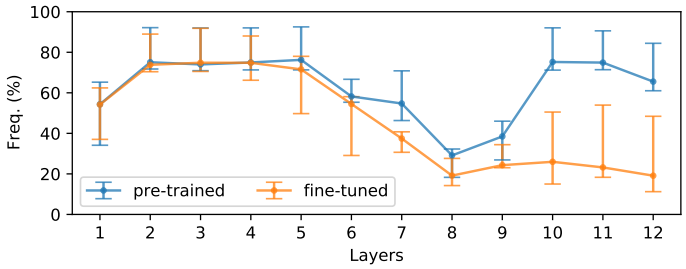
4.4.1 Attention and semantic similarity

Implementation. The goal of this experiment is to evaluate the extent of the attention that BERT gives to words with high similarity in solving the EM task. For each dataset record, two sets are created: one including the pairs of words with the highest attention score, i.e. the ones in the last quartile according to the values computed on an average attention head obtained by averaging all heads referring to the same layer; and the second with the most semantically similar pairs of words, i.e. the ones in the last quartile computed measuring the cosine similarity of their fastText embeddings. The experiment measures the percentage of shared pairs in the sets.

Figures 4.11a and 4.11b show the results of the experiment, aggregated per dataset and per layer, respectively on the $setting = (SP, -, PT/FT)$.



(a) Frequency across the datasets.



(b) Frequency across the layers.

Fig. 4.11 Attention to word pairs with high semantic similarity. The y-axis shows the percentage of word pairs with the highest attention scores that are also highly semantically similar, according to their fastText embeddings.

Discussion. Figure 4.11a shows that generally the pre-trained models generate a percentage of shared pairs of words higher than the fine-tuned. Figure 4.11b shows that fine-tuning process largely decreases in the last layers the attention to pairs of highly similar tokens. The results of this experiment are somewhat unexpected since other experiments made on NLP tasks [83, 64] demonstrate that the semantic knowledge (1) is located in the last layers and (2) is largely exploited by transformers. It is hypothesized that BERT focuses on another kind of knowledge where the pragmatics complements the semantics and with a higher granularity than the one offered by the one-to-one token similarity. This assumption is confirmed by the fact that the fine-tuning improves the effectiveness of the results (see the experiments in Section 4.2.1) and that the deletion of the heads with the highest presence of the MAA pattern largely decreases the results (see the experiments in Section 4.3.2).

4.4.2 Embeddings and semantic similarity

Implementation. This experiment complements the one reported in the previous Section by analyzing the relationship between highly similar tokens, as resulting with the fastText embeddings, and the BERT embeddings. In particular, the goal is to evaluate if (1) semantically similar pairs of terms, according to fastText, give rise to close BERT embeddings and (2)

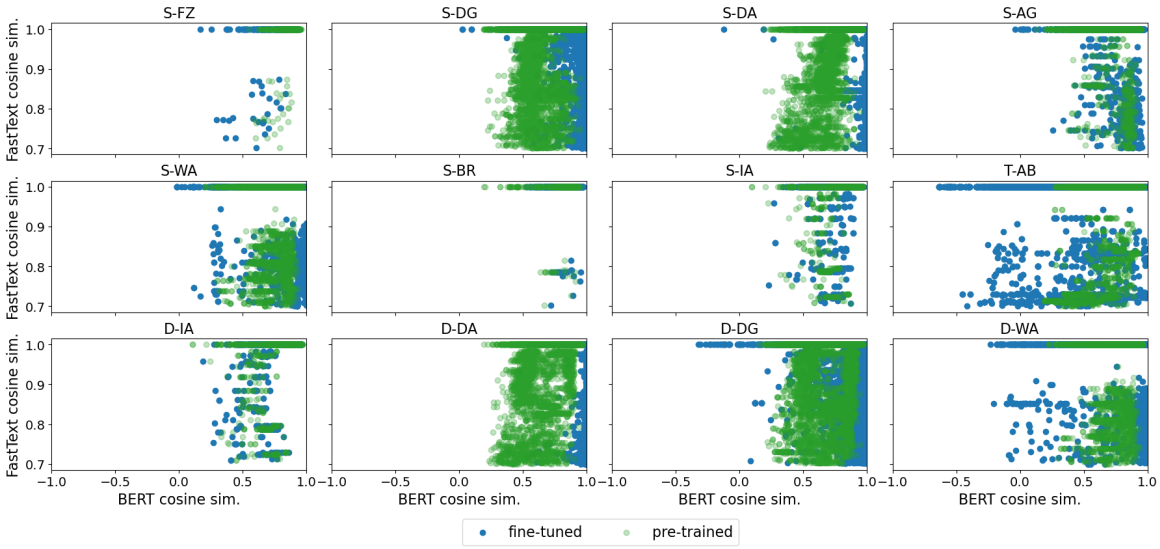


Fig. 4.12 Comparison between BERT and fastText embeddings. The diagrams show the cosine similarity of the embeddings generated by BERT for word pairs with cosine similarity greater than 0.7 according to the fastText encodings.

if and how the fine-tuning changes the process. Note that this experiment differs from the one in Section 4.2.3, since it affects pairs of semantically similar terms instead of random tokens from matching and non-matching entity descriptions. The results of the experiment are shown in Figure 4.12 for the $settings = (SP, -, PT/FT)$, where, for sake of simplicity, only the pairs with a cosine similarity greater than 0.7 according to the fastText encodings have been reported.

Discussion. The visual inspection of the distributions does not show any correlation between semantic similarity and the BERT embeddings. This result confirms the findings of the previous experiment: there is no correlation between the similarity of the BERT embeddings and the semantic similarity of the tokens. This happens even for tokens with the highest semantic similarity, which correspond in some cases to pairs of tokens with a similarity of the embeddings close to zero or negative. Finally, the fine-tuning process has significantly modified the embeddings space: in many datasets (with the exception of T-AB and D-WA and S-AG) the similarity of BERT's embeddings has grown considerably.

4.4.3 Gradient and semantic similarity

Implementation. The analysis of the gradient allows us to evaluate the actual contribution of the semantically similar pairs of tokens on the inferences performed by the EM classification model. As in the experiment in Section 4.3.3, the technique described in [126] is used to

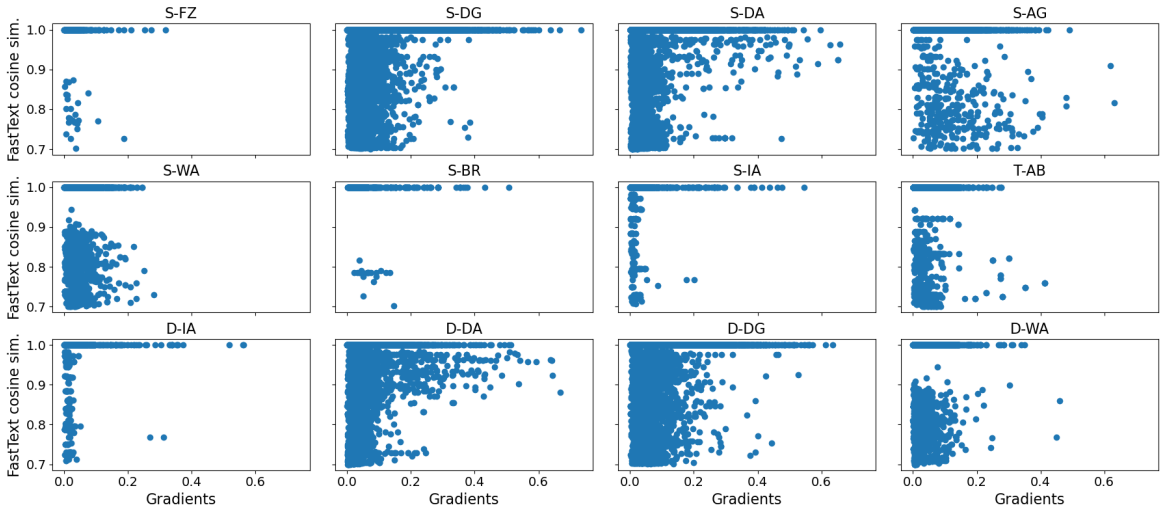


Fig. 4.13 Comparison between gradient and semantic similarity. The gradients generated by a fine-tuned BERT model are compared with the cosine similarity of the fastText embeddings of pairs of words (only similarities greater than 0.7 are shown).

calculate the gradients associated with all the tokens belonging to the EM records. Then exclusively the gradients associated with the pairs of words with a cosine similarity of the relative fastText embeddings greater than 0.7 are selected and these values are summed to obtain a gradient for each pair of terms. In Figure 4.13 the distribution of gradients is compared to the similarity of the fastText embeddings related to the pairs of words in the *setting* $(SP, -, FT)$.

Discussion. The experiment shows that there is a low correlation between the semantic similarity between the tokens and a high value for the gradient. This confirms the findings of the previous experiments: the semantic similarity of the tokens is generally not taken into account by the BERT model and is not exploited for supporting the EM task.

4.5 Lessons learned

Summarizing the results obtained from the experiments, it can be observed that even if BERT-based architectures represent a breakthrough in performing EM (Section 4.2.1), the reasons why they largely support the process can be only partially explained. Answering the three questions introduced in Section 4.1 allows us to observe:

(1) The fine-tuning process is crucial for improving the effectiveness of the BERT-based EM models. The experiments in Section 4.2.2 show that EM-specific knowledge is mainly encoded in the last layers of the architecture (as already observed in analyzing the BERT's

behavior in performing other NLP tasks [77, 61, 60, 112]) and the embedding space changes with the fine-tuning. This leads to an increase in the number of semantically similar pairs of words found in different entity descriptions (Section 4.2.3).
(2) The specific structure of the EM datasets as composed of descriptions of pairs of matching / non-matching entities is recognized and exploited for performing the EM task. The experiments clearly show that not only the attention is given to tokens in the same EM record and belonging to different entity descriptions (Section 4.3.1) but also that matching attributes are recognized (Section 4.3.2). The analysis of the matching attribute attention pattern let emerge an unexpected result: a pattern, the MAA, identifying the matching attributes is found and despite its frequency in the datasets decreases with the fine-tuning, this knowledge is observed to represent a pillar for the effectiveness of the EM process (Section 4.3.2). Moreover, BERT can see that not all attributes are equally important in the EM process and that the importance of the attribute varies when considering matching and non-matching entity descriptions (Section 4.3.3).
(3) The semantic similarity of the tokens is not a key knowledge for the EM process. This is another unexpected result: the attention to semantically similar tokens decreases with the fine-tuning (Section 4.4.1), and no correlation between the semantically similar embeddings computed with the fastText and the BERT approaches was found. Finally, the EM model does not rely on the pair-wise semantic similarity relationships between tokens (Section 4.4.3). The model seems to focus on a more contextualized kind of knowledge where pragmatic knowledge (discovered by BERT) complements the semantic knowledge.

4.6 Conclusion

This Chapter analyzed the BERT's behavior in performing Entity Matching with the aim of understanding the reasons for its high performance. In this Chapter, it has been discovered that BERT can recognize the structure of EM datasets and extracts from the entity descriptions semantic knowledge that goes beyond the pair-wise association between tokens. Moreover, through the fine-tuning process, BERT learned to distribute the attention depending on whether the descriptions refer to matching or non-matching entities.

Future work will be devoted to further clarifying the reasons that make this architecture so performing on the EM task by 1) identifying the components that contribute most to the matching predictions, 2) experimenting with different fine-tuning approaches and 3) correlating the attention weights with the effectiveness of the prediction.

Chapter 5

An Intrinsically Interpretable Entity Matching System

In the previous Chapter we analyzed BERT's internal mechanisms for performing Entity Matching, revealing insights into its performance. However, despite these advances, transformer-based models like BERT remain inherently opaque due to their complex architectures, which limits their interpretability. While post-hoc explanations can shed some light on their decisions, fully understanding their reasoning remains challenging and incomplete.

To address this limitation, this chapter shifts the focus toward designing an inherently interpretable Entity Matching system. Intrinsically interpretable models inherently incorporate transparency, enabling direct comprehension of their decision-making processes without requiring external explanation methods.

This chapter introduces a novel interpretable approach to Entity Matching designed specifically to address these interpretability and usability issues. The proposed solution aims to complement previous findings through an intrinsically interpretable architecture by offering transparent and intuitive explanations alongside accurate matching predictions.

Explainable entity matching is a crucial aspect of working with textual databases, where records can be composed of dozens of features that describe pairs of entities. In such scenarios, generating usable and interpretable explanations for the model's decisions can be particularly challenging. An explanation typically consists of weights associated with the input of the model. The weights measure the impacts of the features in the decision made by the model [92]. A feature-based representation of the explanations can lead to usability problems in the case of textual databases where records can be composed of dozens of features[1]. The scenario is even worse if we consider the case of textual databases used

[1]We adopt the term feature to refer to the tokenized words extracted from the entity descriptions.

Table 5.1 A fragment of an EM dataset

Entity 1			Entity 2		
Name	**Manufac.**	**Price**	**Name**	**Manufac.**	**Price**
exch srvr external sa eng 39400416	microsoft licenses	42166	39400416 exch svr external l/sa	microsoft licenses	22575
digital camera with lens kit dslra200w	sony	37.63	digital camera leather case 5811	nikon	36.11

for performing EM tasks. Here, records describe pairs of entities. They can be composed of a large number of features, thus generating very large, difficult to manage and understand explanations[132, 14]. Moreover, records can contain a large number of duplicated elements. This happens especially in records representing matching entities, which consist of pairs of similar entity descriptions. Such a duplication leads to explanations that are difficult to read (it is necessary to specify which is the entity description to which they belong– the first in the record, a.k.a., the *left entity*, or the second, a.k.a., the *right entity*) and to be interpreted (the same features may have different weights depending on the description to which they belong, and this can generate confusion and uncertainty).

Example 1. Table 5.1 shows two records extracted from an EM dataset. Each record represents two entities, each one describing an object with the same attributes (*Name*, *Manufacturer*, and *Price*). A label is added to the record to indicate that the entity descriptions match (i.e., they refer to the same real-world entity) or do not match (i.e., they refer to different entities). An explanation assigns a score to each feature (i.e., to each tokenized word) in the descriptions, e.g., the approach assigns a score to the feature `external` belonging to the attribute *Name* in the first entity description that can be different from the score assigned to the same feature used in the second entity description.

It is therefore evident that the feature-based granularity of the explanations does not allow users to effectively interpret the results of the EM models. More intuitive information units that consider records as composed of a pair of entity descriptions are needed. We call these information units "*decision units*". We conceive them either as pairs of semantically similar features found in the descriptions of different entities (i.e., the *paired decision units*) or isolated features (i.e., the *unpaired decision units*), which do not have a correspondence in the second entity.

Example 2. The feature `external` from the description of the first entity in Table 5.1 can be associated with the same feature from the second description to form a paired decision unit.

For the feature `lens` from the description of the first entity in the second row, we cannot find any correspondence in the second entity. `Lens` will be then considered as an unpaired decision unit.

In this chapter, we propose an architecture template for intrinsically explainable EM models based on decision units. The idea is that they can constitute the feature space where to train an intrinsically explainable EM model. The architecture template includes three main components (see Section 5.1.1): the decision unit generator, in charge of computing the decision units from the dataset records; the decision unit relevance scorer and the explainable matcher. The *Decision unit relevance scorer* is in charge of the computation of weights (the *relevance scores*) to assign to the decision units. For example, in the case of entities representing people, the decision units associated with the attribute `Name` can have more importance than the ones associated to the attribute `home or work addresses` in deciding if they are referring to the same person. In general, the weights lead paired decision units to push toward match decisions, unpaired decision units toward non-match decisions. Nevertheless, we can find paired decision units in descriptions of non-matching entities and vice-versa. For instance, the description of two different people can share the same address. The specific contribution for each feature has to be therefore computed by taking into account the context for each unit provided by the other units in the record and the other records in the dataset. The *Explainable Matcher* is the architectural component in charge of computing predictions and explanations. It consists of an intrinsically explainable binary classifier trained with the scored decision units. As usual for these kinds of explainers, an *impact score* can be computed for each feature by multiplying its score with the corresponding weight in the trained classifier. The impact score measures the importance of the feature in the prediction. To improve the accuracy, our approach does not directly use the relevant scores to train the model, but applies further feature engineering to inject some structural knowledge in the training set, i.e., the different importance that the dataset attributes assume in the matching decision.

WYM (Why do You Match?)[2] is the implementation of the architecture template for EM textual datasets we propose in this chapter (see Section 5.2). In WYM, the entity descriptions are encoded with word embeddings generated through the BERT language model [39]. We addressed the assignment problem [150] concerning the generation of the decision units with a special implementation of the stable marriage (SM) algorithm [55] maximizing the cosine similarity of the embeddings. Then, we used a deep fully connected layer to infer the relevance scores given the word embeddings. The prediction and the impact scores

[2]WYM implementation is available in the project github at https://github.com/softlab-unimore/WYM.

are computed through an explainable binary classifier. The model is trained on features generated by specific transformations of the relevance scores that allow it to learn structural and pragmatic knowledge.

The application of inverse transformation functions on the coefficients from the trained model generates a set of scores for each decision unit providing an overall weight of its importance for the model. The impact scores are then computed by aggregating the scores per unit and combining them with the relevance scores previously assigned. The impact scores allow the users to interpret the predictions, by identifying the crucial decision units in the evaluation of an EM record. In particular, decision units with positive impact scores are the ones pushing toward an entity matching decision; the ones with negative scores provide evidence of no match. The experimental evaluation shows that not only WYM reaches high accuracy performance, but also a high interpretability level.

The main contributions of this chapter are:

- the use of decision units to represent the basic, atomic information content of a record of an EM dataset;
- the introduction of a novel *interpretable architecture template for EM* able to compute the terms that mainly contribute to taking the matching / non-matching decision;
- an implementation of the architecture relying on the *innovative use of embeddings*, for finding correspondences between terms in the entity descriptions and for scoring the importance of these correspondences. These scores help both the interpretability and the achievement of high performance, thus making our proposal an effective yet interpretable approach;
- a deep evaluation of the effectiveness of our approach in finding EMs and providing useful explanations.

The rest of the chapter is organized as follows. Section 5.1 introduces the main features of the architecture template; Section 5.2 describes its implementation in WYM; Section 5.3 includes the experimental evaluation; Finally, in Section 5.4, we sketch out some conclusions and future work.

5.1 The design of an interpretable entity matching system

EM is a complex task even for people. It requires analyzing the meaning of a pair of textual entity descriptions to evaluate if they refer to the same real-world entity. Recent approaches

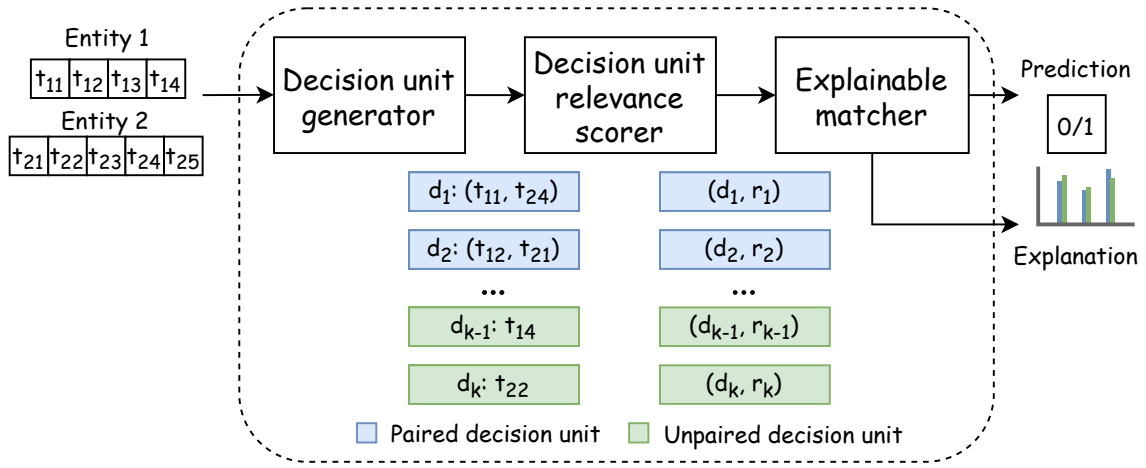


Fig. 5.1 EM Architecture Template

consider EM as a binary classification problem and exploit DL techniques to deal with it. Nevertheless, the design of an approach to be deployed in real business scenarios cannot only consist of a classifier with the associated probability score. It has to provide the reasons for that decision. The usual interpretable ML techniques assign a weight to each feature, i.e. in textual datasets to each token in the record. Our approach introduces the concept of decision unit, that can provide more compact explanations to EM records.

5.1.1 A Reference Architecture Template for Interpretable EM

From the user perspective, an interpretable EM system is a black-box application that receives a pair of textual descriptions as input, infers if they refer to the same real-world entity, and explains the prediction.

More formally, let us consider a bag of features $e = \{t_1, t_2, ..., t_N\}$, where t_i, $i = 1, 2, ..., N$, obtained by the application of a featurization technique to an entity description, Given a pair of entities $r = (e_x, e_y)$, an interpretable EM model returns (1) a prediction $P(r) \in \{0, 1\}$, where 1 means that the entities are matching and 0 non-matching, (2) an explanation $EX(r) = \{(d_r, i_r)\ \forall d_r \in D_r, i_r \in \mathbb{R}\}$, where D_r represents the set of decision units extracted from r. A decision unit can be a single feature from the entity description (an unpaired decision unit) or it can be a pair of features, one for each description (a paired decision unit). The score i_r represents the impact of d_r in the decision. We recall that the decision units with positive i_r push the prediction towards the match, negative i_r towards no-match.

We design an architecture template composed of three components to generate interpretable EM predictions, as shown in Figure 5.1: 1) The *Decision unit generator*, responsible for the extraction of the decision units D_r from each pair of entity descriptions r, 2) The

Decision unit relevance scorer, to determine the relevance of the decision units for the matching task, and 3) The *Explainable Matcher* which composes the final match prediction and generates the explanations $EX(r)$ by attributing an impact score i_r to each decision union d_r.

Decision unit generator

Given an EM record r, this component extracts both the paired and the unpaired decision units. We call $\mathscr{M}_r$ and $\mathscr{N}_r$ the sets of the paired and unpaired decision units extracted by the component. Note that $D_r = \mathscr{M}_r \cup \mathscr{N}_r$. A decision unit $d \in D_r$ can be formalized as:

$$d = \begin{cases} (t_i, t_j), & t_i \in e_x, t_j \in e_y \text{ paired unit} \\ t_i \vee t_j, & t_i \in e_x, t_j \in e_y \text{ unpaired unit} \end{cases} \tag{5.1}$$

Decision units have to respect the following constraints: 1) each feature of an entity description belongs at least to a decision unit, and 2) a feature belonging to an unpaired decision unit cannot belong to a paired decision unit.

The search of the paired decision units can be conceived as a relaxed Stable Marriage (SM) assignment problem [69], where each feature $t \in e_x$ (or e_y) is associated with a preference list of features of the other entity of the record r based on their syntactic/semantic similarities. The goal of this task is to assign to each token $t \in e_x$ one or more tokens $t \in e_y$ such that the syntactic/semantic preferences are maximized. Note that the formulation of the SM problem in the EM domain needs to relax some constraints of the original formulation, by allowing each term to participate in multiple assignments, and the definition of preference lists of variable length. The set of paired and unpaired (i.e. the remaining terms after the assignment) decision units constitute the decision units D_r of the EM explanation related to the record r.

Decision unit relevance scorer

Decision units contribute in determining if the descriptions they belong to refer to the same real-world entity. In general, paired decision units push towards the decision of entity match and the unpaired towards entity non-match, but each decision unit contributes with a different level of importance. The relevance score provides a measure for this importance, by evaluating the strength with which a decision unit pushes in isolation the decision towards a match or a non-match prediction. The relevance scores are not related to the semantic similarity that we used to form the paired decision units (see the experimental evaluation in Section 5.3.1).

The component responsible for assigning a relevance score to each unit is the *Decision unit relevance scorer*. The component uses a relevance scorer function $rs : D_r \rightarrow [-1,1]$ to compute a score for each decision unit $d \in D_r$. A score close to -1 implies that the decision unit pushes the prediction towards a non-match decision, while a value close to 1 mainly contributes to a match decision. Furthermore, we impose that this function is symmetric for paired decision units, i.e. given a decision unit $(t_x,t_y) \in \mathscr{M}_r$, $rs((t_x,t_y)) = rs((t_y,t_x))$. In this way, the assignment of the relevance score is invariant with the provenance of the term (i.e., whether it is part of the left or of the right entity).

The output of the component is, therefore, a set of relevance scores associated with the decision units of the record r: $R_r = \{(d, rs(d)), \forall d \in D_r\}$.

Explainable Matcher

The decision units and the relevance scores of a pair of entities are the input of the *Explainable Matcher* that establishes if they refer to matching or non-matching entity descriptions. The component aims at integrating the "local" knowledge provided by the relevance scores of the isolated decision units with a "contextualized" knowledge provided by the other units in the entity descriptions. This knowledge includes structural knowledge provided by the attributes in the dataset schema S to which the decision units belong to, and pragmatic knowledge through the evaluation of the context of the decision units in the same record.

Our idea is to encode this knowledge through a feature engineering process in a new dataset built upon the relevance scores. The dataset is then used to train a binary interpretable classifier to solve the EM task. The impact scores are then computed by applying inverse transformation functions on the trained coefficients of the model and combining the results obtained with the relevance scores. Formally, given the set of decision units D_r, their relevance scores R_r and the dataset schema S, the Explainable Matcher generates a set of features $F_r = f(D_r, R_r, S)$ using a featurization function f. These features are then used as input to an interpretable (linear) binary classifier C, defined over a set of parameters W, which returns the matching prediction p, i.e., $p = C_W(F_r)$. Given the inherently explainable nature of such a classifier, its fitted parameters W can be used to derive an importance score for each decision unit by applying the expression $I_r = F_r W$.

5.1.2 Challenges

We list some challenges that an implementation of the architecture template has to address. In Section 5.2, we show how these challenges are addressed by our implementation WYM.

(R1) **Mismatch in labels.** This happens when paired decision units, treated as evidence for the match, are found in entries labeled as no-match entity (e.g., the entities are different products, but they share the same brand). An automatic approach could learn that those decision units lead to a non-matching prediction, thus introducing possible mistakes in other inferences. The same happens symmetrically for unpaired decision units in matching entities.

(R2) **Matching search space size and multiplicity**. Although in structured datasets, the attributes define boundaries within which to preferably search for the definition of paired decision units, they should not be considered hard boundaries. Real datasets are frequently "dirty" and contain misaligned data. Moreover, terms describing similar concepts can occur multiple times within an entity description, thus leading to the definition of possible one-to-many and many-to-many relationships between tokens from two entity descriptions.

(R3) **Permutation invariance**. The relevance measure computed for decision units composed of paired tokens has to be symmetric, i.e. the score does not have to change when reversing the order of the tokens.

(R4) **Context-awareness**. The same decision unit can generate a differentiated impact depending on the descriptions and the attribute where it appears.

(R5) **Cardinality invariance**. The relevance measure has to be defined and normalized to manage uniformly decision units composed of paired and unpaired tokens.

5.1.3 Running example

Figures 5.3c and 5.3d show two examples of explanations (i.e. the impact scores) obtained by the application of our implementation of the architecture to the data in Table 5.1. Figure 5.3c refers to matching descriptions: a number of paired decision units are found and the green bars show their impact on the decision taken by the EM system. The product code is the decision unit that provides the largest contribution towards the matching. Figure 5.3d refers to non-matching entity descriptions. Red bars show the impact towards the non-matching decision, mostly supported by unpaired decision units. We observe that as usual for non-matching descriptions, the decision units supporting the decision have a similar impact.

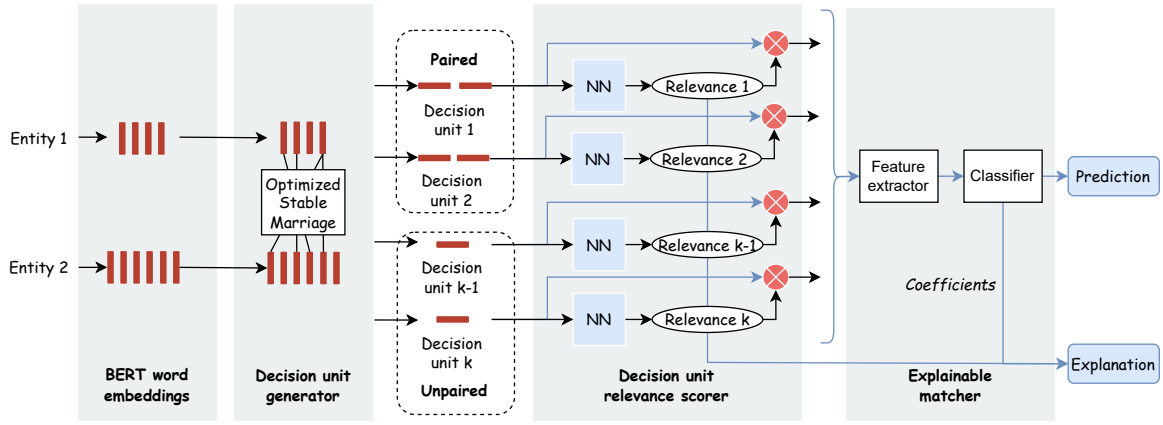


Fig. 5.2 The Functional Architecture Template implemented in WYM.

5.2 The WYM Explainable Matcher

WYM is an implementation of the architecture template for textual databases. The main components are represented in Figure 5.2 and described in Section 5.2.1 (the Decision unit generator), Section 5.2.2 (the Relevance scorer), and Section 5.2.3 (the Explainable matcher). In the following, we suppose that entity descriptions have the same schema, and we call *matching attribute* the attribute in the second entity description corresponding to one selected in the first description.

5.2.1 Decision unit generator

This component implements three main functionalities. Firstly, a tokenizer is applied to transform the textual values into tokens. Then a word embedding technique allows us to assign semantic encodings to the tokens. Finally, tokens from an entity description are possibly paired with the ones of the second description, thus forming paired decision units. For this last operation, we leverage the schema of the dataset, if any, to reduce the alignment space.

Text tokenization and encoding

We apply the BERT language model [39] to encode the meaning of the entity descriptions into contextualized embeddings. The attribute values of the dataset records are concatenated and word-piece tokenization with stop word removal is applied. In Section 5.3.1, the approach is experimented with the pre-trained model and different fine-tuning techniques. In the simplest approach, the embeddings are generated by averaging the hidden states (from the second to the last layer) of the BERT network. Averaging the values of the hidden states

of more layers allows our implementation to have a good trade-off in representing in the embeddings the target feature and its context, which is provided by the other features in the same entity description. Among the fine-tuning technique, the one which obtains the best performance on average on all datasets in the experiments is Sentence BERT [109], obtained by a modification of the pre-trained BERT network that uses siamese and triplet network structures.

Pairing

We exploit the contextualized and semantically rich embedding space provided by BERT to discover the paired decision units. To foster the creation of strong units and to manage the heterogeneity of the real datasets (challenge R2 in Section 5.1.1), we define three search spaces where to evaluate the possible correspondences: (1) The reduced search space formed by the matching attributes only: the dataset structure guarantees that the found intra-attribute correspondences describe the same entity property. (2) The larger search space formed by all attribute values: inter-attributes correspondences allows us to manage dirty and misaligned dataset content. The features paired in the first search space are not considered in this one. (3) The search space formed by the remaining unpaired features of one entity description with the already paired features of the other: this fosters the creation of one-to-many relationships for managing repetitions, periphrasis, and other forms of heterogeneity existing in real datasets.

WYM relies on the cosine similarity to identify close embeddings. As usual for this kind of approach, an experimentally determined threshold is established, beyond which two embeddings are considered similar. To make the approach flexible, WYM allows the users to select three thresholds: θ is the threshold used for computing the intra-attribute correspondences in the first search space; η for the inter-attribute correspondences in the second search space; ε for the evaluation of the possible correspondences involving unpaired features with already paired features. Even if the optimal choice for the threshold depends on the dataset and can only be experimentally determined, the best results are obtained with increasing values for the thresholds from θ to ε. The reason is that θ is used to evaluate pairs of embeddings that belong to the specific and related context provided by the matching attribute. The narrow context supplies the selection of a less selective threshold. Conversely, the value for ε should be the highest, since the search space where this threshold is used is the broadest.

Algorithm 1 provides a detailed description of the process for discovering the decision units. It takes as input a pair of entity descriptions (e_x, e_y) that are part of the same record r and are defined over the common schema $S = \{A_1, A_2, ..., A_M\}$, and the thresholds $\theta, \eta, \varepsilon$ to evaluate the similarity of the token embeddings in the search spaces. It returns the paired $\mathscr{M}$

and unpaired $\mathscr{N}$ decision units found. Firstly, the sets of paired and unpaired decision units are initialized (lines 1-3) and separated according to their provenance (i.e., the description of the left (x) or right (y) entity). We then apply the first phase of decision unit discovery, where the embeddings of the tokens belonging to the matching attributes are pair-wise evaluated (lines 4-8). We denote with $e_x[A]$ (or $e_y[A]$) the tokens of the attribute A in a target entity. The function `GetSMPairs` computes pairs of token embeddings and the ones with a cosine similarity higher than the threshold θ are selected as (intra-attribute) paired decision units and are stored in $\mathscr{M}$. The remaining ones are temporarily labeled as unpaired decision units (lines 7,8) and evaluated as part of inter-attribute correspondences. This test is performed in lines 9-12, where the unpaired tokens from an entity are evaluated with the ones of the other entity. The second threshold η is applied to evaluate the similarity of the embeddings. Finally, the last step (lines 13-18) used threshold ε to find matches between the remaining tokens with already formed decision units. This allows us to generate chains of decision units logically representing one-to-many relationships among the terms in the descriptions (challenge R2).

The operation for pairing the embeddings is provided by the `GetSMPairs` function, which relies on the Gale–Shapley implementation of the Stable Marriage (SM) algorithm[3] [69]. SM is the problem to find one-to-one matches between two equally sized sets of elements, considering preference lists in which each element in a set expresses its preference over the members of the opposite set. The goal is to guarantee that for each element the best connection in its preference list has been selected We re-adapt the algorithm to identify embeddings referring to semantically related terms. In more detail, each embedding is associated with a preference list defined by the closest embeddings in the BERT embedding space (according to a threshold applied to their cosine similarity). With respect to the original problem, in this case, we refer to preference lists of variable length and in which the preference is expressed in continuous (and not boolean) values. The application of this algorithm allows us to more efficiently examine the search space composed of all the possible embedding pairs and to generate correspondences between embeddings in a holistic perspective. In addition, the algorithm creates decision units that respect the required constraints.

Finally, we observe that basing the approach on fine-tuned BERT embeddings allows us to address challenge R4 of Section 5.1.2, since the encodings rely on the contextualized knowledge provided by the other decision units in the entity descriptions.

[3]The Gale-Shapley algorithm finds a stable matching in time $O(n^2)$. The algorithm is executed three times per dataset as shown in Algorithm 1.

Algorithm 1: `DecisionUnitDiscovery`

Input : (e_x, e_y), a pair of entity descriptions in the record r having the same set of aligned attributes $S = \{A_1, A_2, ..., A_M\}$;
θ, threshold similarity intra-attribute;
η, threshold similarity inter-attribute;
ε, threshold similarity for one-to-many relations.

Output : $\mathscr{M}_r$, a list of paired decision units for the record r;
$\mathscr{N}_r$, a list of unpaired decision units for the record r.

$\mathscr{M}_r \leftarrow \emptyset$;
$\mathscr{N}_x \leftarrow \emptyset$;
$\mathscr{N}_y \leftarrow \emptyset$;
`# Intra-attribute correspondences`
foreach $A \in S$ **do**
  $M_A \leftarrow$ `GetSMPairs(`$e_x[A], e_y[A], \theta$`)`;
  $\mathscr{M}_r \leftarrow \mathscr{M}_r \cup M_A$;
  $\mathscr{N}_x \leftarrow \mathscr{N}_x \cup (e_x[A] \setminus \{t_x | (t_x, t_y) \in M_A\})$;
  $\mathscr{N}_y \leftarrow \mathscr{N}_y \cup (e_y[A] \setminus \{t_y | (t_x, t_y) \in M_A\})$;
`# Inter-attribute correspondences`
$M^{\ulcorner} \leftarrow$ `GetSMPairs(`$\mathscr{N}_x, \mathscr{N}_y, \eta$`)`;
$\mathscr{M}_r \leftarrow \mathscr{M}_r \cup M^{\ulcorner}$;
$\mathscr{N}_x \leftarrow \mathscr{N}_x \setminus \{t_x | (t_x, t_y) \in M^{\ulcorner}\}$;
$\mathscr{N}_y \leftarrow \mathscr{N}_y \setminus \{t_y | (t_x, t_y) \in M^{\ulcorner}\}$;
`# One-to-many correspondences`
$M^x_{\ulcorner} \leftarrow$ `GetSMPairs(`$\mathscr{N}_x, \{t_y | (t_x, t_y) \in \mathscr{M}_r\}, \varepsilon$`)`;
$M^y_{\ulcorner} \leftarrow$ `GetSMPairs(`$\mathscr{N}_y, \{t_x | (t_x, t_y) \in \mathscr{M}_r\}, \varepsilon$`)`;
$\mathscr{M}_r \leftarrow \mathscr{M}_r \cup (M^x_{\ulcorner} \cup M^y_{\ulcorner})$;
$\mathscr{N}_x \leftarrow \mathscr{N}_x \setminus \{t_x | (t_x, t_y) \in M^x_{\ulcorner}\}$;
$\mathscr{N}_y \leftarrow \mathscr{N}_y \setminus \{t_y | (t_x, t_y) \in M^y_{\ulcorner}\}$;
$\mathscr{N}_r \leftarrow \mathscr{N}_x \cup \mathscr{N}_y$;
return $\mathscr{M}_r, \mathscr{N}_r$;

5.2.2 Decision unit relevance scorer

The relevance scores are assigned to the decision units to provide a measure of their importance in the matching decision. WYM assigns the score via a supervised regression model (a dense fully connected neural network in the experimented implementation), built over a dataset where each entry represents a decision unit. The dataset items and the target scores are built according to heuristic rules that take into account the challenges introduced in Section 5.1.1. In particular, we consider unpaired decision units as paired with the special element `[UNP]`. This null element is associated with a zero embedding (challenge R5). Moreover, the dataset describes paired units with two main features: the mean and the absolute difference of the embeddings associated with the pairs. This representation is symmetric (challenge R3) and manages small differences in the embeddings (challenge R4). A special procedure has been designed to correctly address the mismatch of the labels (challenge R1) in the computation of the target score for each entry. The values 1 and -1 are assigned to decision

units with a similarity "consistent" with the target label y, i.e., the value 1 when the paired decision units are similar (i.e., the cosine similarity of their embeddings is greater than a predefined threshold α) and the label represents matching entities; when the label represents non-matching entity descriptions we assign the value -1 to unpaired decision units and to dissimilar paired decision units (i.e., their cosine similarity is lower than a predefined threshold β). We associate the neutral value 0 to decision units that definitely represent the same concepts (the ones with high cosine similarity) when they are associated with non-matching entities (this moves their target label before the average computation from -1 to 0). Similarly, the labels of tokens with unmatched concepts in records representing matching entities are moved from 1 to 0. Equations 5.2 summarizes the rules introduced for paired decision units (l and r are the tokens composing the unit); a similar formula allows the management of unpaired units.

$$\hat{y}(l,r,y) = \begin{cases} 0 & \text{if y = 1 } \bigwedge sim(l,r) < \alpha \\ 1 & \text{if y = 1 } \bigwedge sim(l,r) >= \alpha \\ -1 & \text{if y = 0 } \bigwedge sim(l,r) < \beta \\ 0 & \text{if y = 0 } \bigwedge sim(l,r) >= \beta \end{cases} \tag{5.2}$$

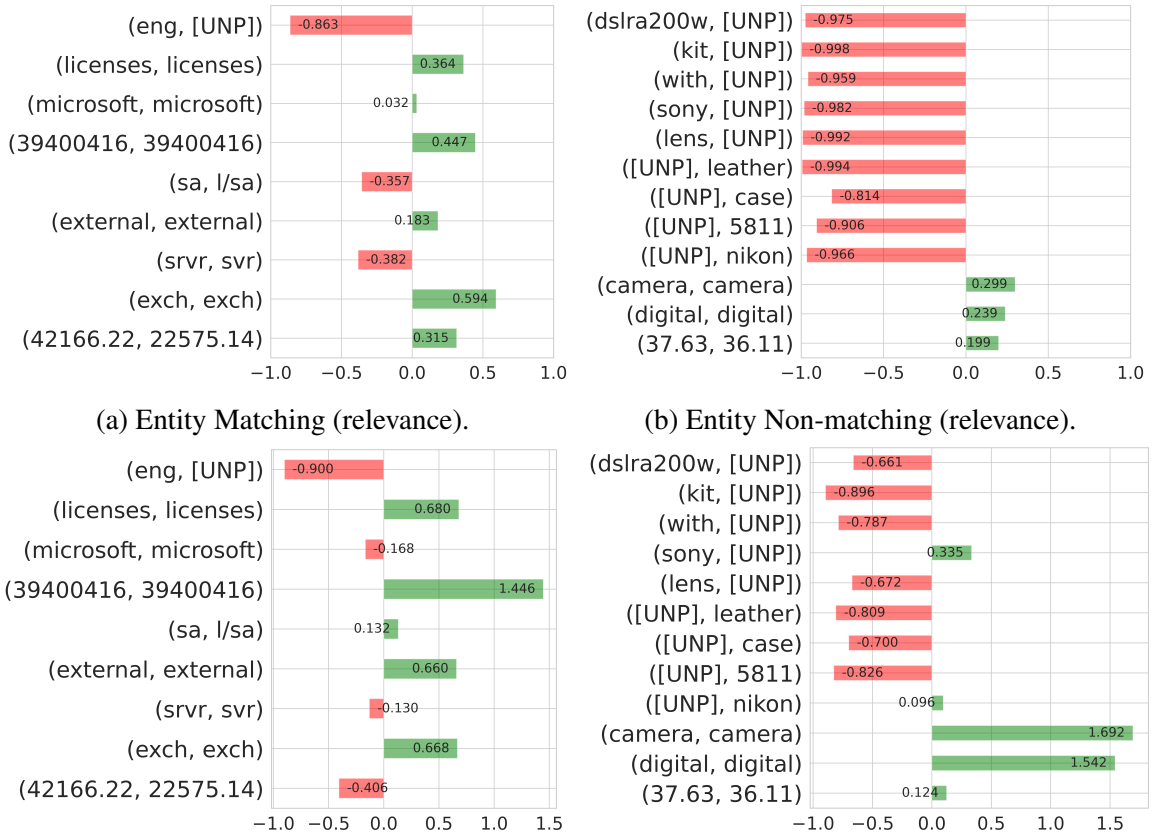
Then, for each decision unit, a unique value y^*_{lr} is computed by averaging the values associated to all its occurrences, as shown in Equation 5.3, where $|E_{lr}|$ is the number of entities in the dataset containing the decision units.

$$y^*_{lr} = \frac{1}{|E_{lr}|} \sum \hat{y}(l,r,y) \tag{5.3}$$

The dataset created through this procedure is used to train a fully connected feed-forward neural network that infers the score for each decision unit. The network is composed of 3 hidden layers with 300, 64, and 32 nodes, using the relu as activation functions. The network was trained with 40 epochs, 256 elements per batch, and a learning rate equal to $3 \cdot 10^{-5}$.

Running Example

Figures 5.3a and 5.3b show the relevance scores computed by WYM on the running example in Section 5.1.3. Figure 5.3a reports the scores for the decision units in the matching entity descriptions. We note that not all the paired units contributed to the matching decisions. To some of them, a negative value is assigned by WYM that pushes the decision towards a no-match. The highest contribution towards the match is provided by the unit `(exch,exch)`, i.e. the abbreviation of the name of the software described and to the product code `(39400416,39400416)`. The highest contribution towards the non-match is provided



(a) Entity Matching (relevance).

(b) Entity Non-matching (relevance).

(c) Entity Matching (impact).

(d) Entity Non-matching (impact).

Fig. 5.3 Explanations with relevance and impact scores

by the term `(eng)`, available only in the description of the first entity, which forms an unpaired decision unit. Recall that relevance scores represent the contribution of the decision units in isolation, with the contextual knowledge provided by the embeddings. Figure 5.3b shows the actual relevance scores for non-matching descriptions. We count a small number of paired decision units pushing towards the match decision, and a large number of unpaired units pushing towards the non-match decision.

5.2.3 Explainable matcher

The relevance scores provide an indication of the contribution of each decision unit to the prediction, which we improved by introducing structural and contextual knowledge. The idea is to apply specific featurization functions that introduce such knowledge to the decision units and relevance scores of the entity descriptions. There are three types of contextual and structural knowledge that we can introduce, by aggregating features and scores per attribute, entity description and record. The functions we apply include simple statistical operators (such as max, min, count, sum, mean, median, and the difference between max and min), and are applied to the decision units aggregated for the selected scopes. For example, we introduce a featurization function that aggregates the relevance scores of the paired and unpaired decision units belonging to the same attribute.

The new dataset is used to train a binary classifier that infers if pairs of entity descriptions refer to the same real-world entity. Actually, WYM relies on a pool of ten interpretable classifiers (Logistic Regression - LR, Linear Discriminant Analysis - LDA, KNN, CART, Naive Bayes - NB, Support Vector Machine - SVM, AdaBoost - AB, Gradient Boosting - GBM, Random Forest - RF, and Extra Tree –ET), and the one obtaining the best F1 score is selected. Finally, we exploit the interpretable nature of the selected models to estimate the impact that each decision unit generates on the prediction. Firstly, we extract from the classifiers the coefficients learned. They provide an indication of the importance of each generated feature. Then, we apply an inverse feature engineering transformation to identify the decision units that contribute to each feature and associate them with the impact score. For example, the coefficient of the model associated with the average of the relevance scores of the decision units belonging to a specified attribute, contribute to each decision unit with a weight equal to 1 / N, where N represents the total number of decision units in the attribute. The impact scores are then obtained by splitting the contributions provided by the coefficients of the classification model to the decision units involved. For each decision unit, the related coefficients are then multiplied by the relevance score, and the results averaged.

Table 5.2 The Magellan Benchmark used in the experiments.

Dataset	Type	Datasets	Size	% Match
S-DG	Structured	DBLP-GoogleScholar	28,707	18.63
S-DA		DBLP-ACM	12,363	17.96
S-AG		Amazon-Google	11,460	10.18
S-WA		Walmart-Amazon	10,242	9.39
S-BR		BeerAdvo-RateBeer	450	15.11
S-IA		iTunes-Amazon	539	24.49
S-FZ		Fodors-Zagats	946	11.63
T-AB	Textual	Abt-Buy	9,575	10.74
D-IA	Dirty	iTunes-Amazon	539	24.49
D-DA		DBLP-ACM	12,363	17.96
D-DG		DBLP-GoogleScholar	28,707	18.63
D-WA		Walmart-Amazon	10,242	9.39

Running Example

Figures 5.3c and 5.3d show the actual impact scores computed by WYM on the examples in Section 5.1.3. Figure 5.3c shows that the decision units describing the product code `(39400416,39400416)` and the type of product `(licenses, licenses)` are the ones mainly supporting the matching decision. We observe as the score highly changes for these two units with reference to the relevance score in Figure 5.3a. We observe also as the last unit, the price of the product, `(42166.22,22575.14)` changes polarization from supporting the match as relevance score (in isolation, the model understands that these are semantically related terms) to supporting the non-match as impact score (the price is important in establishing if we are referring to the same entity). Figure 5.3d shows a example of a non-match entity. As it frequently happens for this kind of descriptions, there is not a single unit that supports the decision, but there are many units that contribute in predicting the descriptions as different. We note that with reference to the relevance score in Figure 5.3b, the importance of the paired units increased. Both the products are related to digital cameras, but the codes are unpaired `(dsira200w)`, `(5811)`, as well as many other product features. Moreover, we observe that the unpaired unit `(sony)` pushes towards a match. This is probably due to the dirty dataset, where the brand is not usually part of both entity descriptions.

5.3 Experiments

The experimental evaluation aims at answering four main questions: 1) How effective is WYM in solving EM tasks (Section 5.3.1); 2) If the impact scores provide a reliable

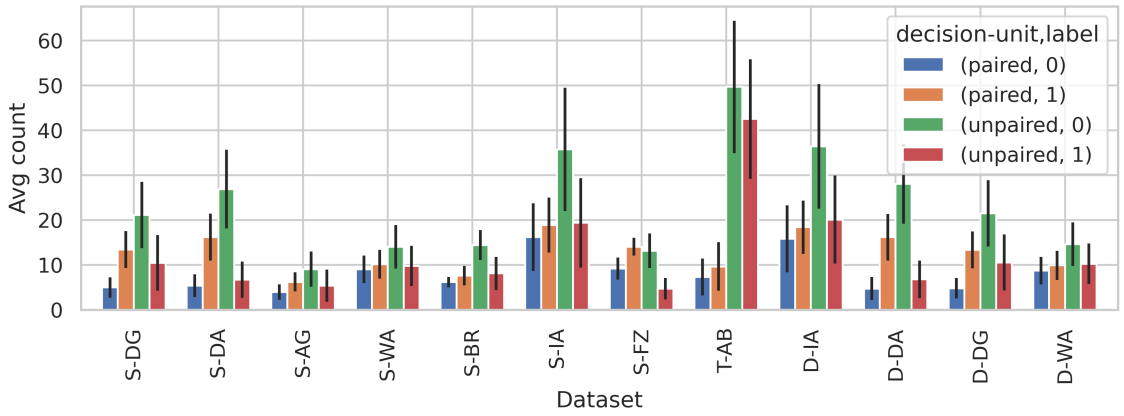


Fig. 5.4 Average distribution of the decision units in the datasets.

interpretation of the EM predictions (Section 5.3.2); 3) If the time required to train the system and to compute the explanations makes it usable in real-world scenarios (Section 5.3.3); 4) If the decision-based explanations are effective for the users (Section 5.3.4). For questions 3 and 4 we report the results of the experiments only. Interested readers can refer to our technical report published in the project github.

Datasets. The experiments are performed against 12 datasets provided by the Magellan library[4] which are usually considered the reference benchmark for the evaluation of EM tasks. In Table 5.2, we show some of their descriptive statistics: the total number of records representing matching entities in the fourth column and the percentage of records associated with a matching label in the last column. Figure 5.4 shows the average distribution of paired and unpaired decision units in the datasets. As expected, the overall number of units associated with non-matching descriptions is greater than the one of matching descriptions. Among the former, we see more unpaired than paired decision units. The *T-AB* dataset shows a different distribution, with a large number of unpaired units. The reason is that this is a database of large textual descriptions where the presence of periphrasis makes difficult the creation of paired units. For the purposes of the experimental evaluation, each dataset is divided into training, validation, and test set which were created with 60-20-20 proportions. The implementation of WYM used in the experiments is available in the project github at https://github.com/softlab-unimore/WYM.

Settings. We run the experiments on a machine with 16 GB of RAM, Tesla T4, and 24 Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz. We adopted the following thresholds for the generation of the decision units: $\theta = 0.6$, $\eta = 0.65$ and $\varepsilon = 0.7$.

[4]https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

Table 5.3 Effectiveness measured with the F1 score, and, in brackets, the rank of each model for each dataset. The comparison between WYM and the other approaches is shown in the right part of the table. Values are in bold (underlined) when they differ from WYM more than 3% (less than -3%).

Dataset	WYM	**DM+**	**AutoML**	**CorDEL**	**DITTO**	Δ DM+ (%)	Δ AutoML (%)	Δ CorDEL (%)	Δ *DITTO* (%)
S-DG	0.936 (5)	0.947 (2)	0.940 (3)	0.940 (3)	0.956 (1)	-0.8	-0.1	-0.1	-1.7
S-DA	0.990 (3)	0.985 (4)	0.970 (5)	0.992 (2)	0.99 (1)	0.5	2	-0.02	0
S-AG	0.625 (5)	0.707 (3)	0.673 (4)	0.700 (2)	0.756 (1)	-8.2	-4.8	-7.5	-13.
S-WA	0.726 (4)	0.736 (3)	0.649 (5)	0.940 (1)	0.857 (2)	-0.1	**7.7**	-21.4	-12.0
S-BR	0.839 (3)	0.788 (5)	0.805 (4)	0.889 (2)	0.944 (1)	**5.1**	**3.4**	-5.0	-10 .5
S-IA	1 (1)	0.912 (5)	0.922 (4)	1 (1)	0.971 (3)	**8.8**	**7.8**	0. 0	2.9
S-FZ	1 (1)	1 (1)	0.969 (5)	1 (1)	1 (1)	0.0	**3.1**	0.0	0.0
T-AB	0.645 (4)	0.628 (5)	0.769 (2)	0.649 (3)	0.893 (1)	1.7	-12.4	-0.4	-24.8
D-IA	0.963 (1)	0.794 (5)	0.870 (3)	0.824 (4)	0.957 (2)	**16.9**	**9.3**	**13.9**	0.6
D-DA	0.972 (3)	0.981 (2)	0.969 (5)	0.970 (4)	0.99 (1)	-0.9	0.3	0.2	-1.8
D-DG	0.923 (4)	0.938 (2)	0.934 (3)	0.915 (5)	0.958 (1)	-1.5	-1.1	0.8	-3.5
D-WA	0.603 (3)	0.538 (4)	0.652 (2)	0.512 (5)	0.857 (1)	**6.5**	-4.9	**9.1**	-25.4
AVG	**0.852 (3.1)**	**0.830 (3.4)**	**0.843 (3.8)**	**0.861 (2.8)**	**0.927 (1.1)**				

5.3.1 Effectiveness of the EM Model

The effectiveness of WYM is evaluated according to three main perspectives: in Section 5.3.1, the performance is compared with competing systems; in Section 5.3.1, we evaluate how WYM behaves by varying the size of training sets, and in Section 5.3.1 we introduce a study of the components of the WYM architecture, evaluating different settings and implementation choices.

Comparison with competing approaches

The effectiveness of WYM against the datasets in the benchmark in terms of F1 score is computed. The results are compared with the results achieved by DeepMatcher+[5] (DM+) , one of the pioneering EM systems based on deep learning, AutoML[6] [101], an approach that provides the automatic application of ML models to the EM problem by pipelining AutoML systems with transformer-based encoders, CorDEL [140] and *DITTO* [82], a contrastive DL approach and a BERT-based approach currently representing the state-of-the-art systems for solving the EM tasks. The results are shown in Table 5.3.

[5]DM+ is the combination of experiments / implementations as defined in [82]

[6]We average the results related to the best configuration (Hybrid-EM-Adapter) for AutoSklearn, AutoGluon and H20AutoML

Discussion. The overall WYM performance is slightly better than DM+, similar to AutoML and CorDEL, and worse than *DITTO*. The average F1 score measured on the overall benchmark is 0.852 (WYM), 0.83 (DM+), 0.843 (AutoML), 0.861 (CorDEL) and 0.927 (*DITTO*). If we consider a threshold of $\pm$ 3% from the result achieved by our approach, where we consider the results to be similar, we observe that WYM performs better than DM+ in 4 datasets, worst in 1 dataset, and within the threshold in the remaining 7 datasets; it performs better than AutoML in 4 datasets, worst in 3 datasets, and within the threshold in the remaining 5 datasets; better than CorDEL in 2 datasets, worst in 3 and within the threshold in the remaining 7 datasets; finally, it performs worse than *DITTO* in 7 datasets, and within the threshold in the remaining 5 datasets. The detailed error analysis showed that WYM makes a large number of errors in recognizing product codes in the entity descriptions. In many cases, they form a decision unit even if they are not the same. This is mainly due to the tokenization mechanism introduced by BERT. Heuristics can be applied to address the problem. In particular, we verified an improvement of the F1 score in the T-AB dataset (from 0.645 to 0.754) after the insertion of domain knowledge that allows only equal product codes to belong to the same paired decision units. The table shows in brackets the rank of each model for each dataset. The analysis of the average rank shows that *DITTO* can obtain outstanding results and the other approaches are close to each other.

Take away. As already pointed out in the literature [41], providing interpretability to the predictions comes with the price of decreasing the effectiveness of the approach. We consider WYM as a good compromise between the quality of the predictions and the capability of interpreting them. *DITTO*, which definitely achieves the best performance, acts for the users as an oracle that does not provide any support for understanding the reasons for its decisions. WYM obtains high quality results and provides the decision units with the impact scores that explain the predictions.

Learning Curves

The learning curves provide insight into the dependence of the EM model on the size of the training set. We select samples of increasing size from the training set and we evaluate the F1 score of the predictions. The dimensions of the experimented samples are 500, 1K, 2K records, and the entire dataset. Figure 5.5 shows the learning curves obtained. For sake of simplicity, the experiments were performed by using the encodings obtained with a pre-trained version of the BERT model, but the shape of the curve does not change by using fine-tuning. The curves for the datasets *S-BR*, *S-IA*, *S-FZ*, *D-IA* are not shown: the size of the training (270 records in *S-BR*) and test sets (90 elements in *S-BR*) are too small for a reliable evaluation.

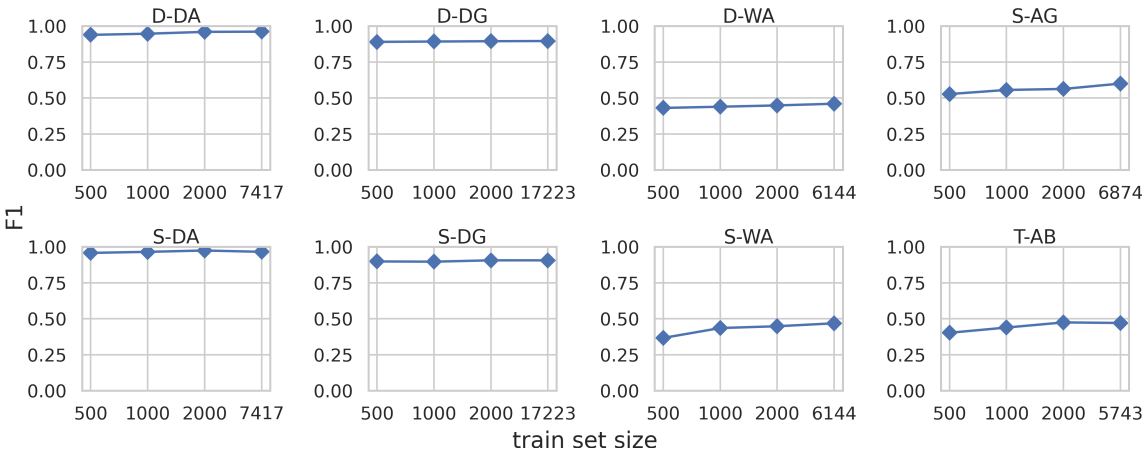


Fig. 5.5 Learning Curves. The training set size is shown in the x-axes, the accuracy of the model (F1 score) in the y-axes.

Discussion. The experiment shows that WYM is not sensitive to the size of the training set. Only in three datasets (S-AG, S-WA, and T-AB) the score increases more than 10% as the dimension of the training set increases.

Take away. The results show that WYM provides good results in scenarios affected by limited availability of data. The experiments show good results even with 500 records. The creation of labels for such an amount of data is feasible in a small amount of time for a human.

Analysis of the WYM components

In this Section, we aim to understand the contribution of each WYM component to the overall performance of the system.

The Decision Unit Generator. The experiment shows the contribution of the word embeddings on the creation of the decision units measured through the overall performance of the approach. The current implementation relies on the SBERT embeddings [109]. We show the performance obtained with two other kinds of embeddings: the pre-trained BERT model, and the BERT model fine-tuned on the EM task. Finally, the performance obtained with decision units computed on the basis of the Jaro-Winkler distance [142], i.e., an edit distance-based similarity measure, offers a simple baseline for the problem. The results of the experiment are shown in Section "Decision Unit Generator" of Table 5.4.

Discussion. The effectiveness does not largely vary if we consider the SBERT fine-tuning adopted in WYM and the other BERT based architectures experimented, as confirmed by the average F1 score and the average rank computed by considering all datasets. In many datasets, fine-tuning was not able to improve the results. Since the goal of the paper is to

Table 5.4 Effectiveness (F1 score) varying the component implementations. In brackets the rank of the model for each datataset.

	WYM	Decision Unit Generator			Scorer			Matcher
		j-w dist.	BERT-pt	BERT-ft	bin. scr.	cos. sim.	bin j-w	smp. feat.
S-DG	0.936 (1)	0.923 (6)	0.930 (4)	0.930 (4)	0.932 (3)	0.936 (1)	0.834 (8)	0.904 (7)
S-DA	0.990 (1)	0.980 (5)	0.989 (2)	0.986 (3)	0.976 (6)	0.983 (4)	0.965 (7)	0.952 (8)
S-AG	0.625 (2)	0.542 (6)	0.647 (1)	0.624 (3)	0.532 (7)	0.550 (5)	0.312 (8)	0.607 (4)
S-WA	0.726 (2)	0.710 (3)	0.670 (4)	0.592 (6)	0.458 (7)	0.748 (1)	0.281 (8)	0.611 (5)
S-BR	0.839 (8)	0.848 (7)	0.903 (3)	0.963 (1)	0.933 (2)	0.903 (3)	0.875 (6)	0.848 (5)
S-IA	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	0.981 (7)	0.982 (6)	0.898 (5)	0.836 (8)
S-FZ	1.000 (1)	1.000 (1)	0.977 (4)	0.955 (8)	0.977 (4)	1.000 (1)	0.957 (3)	0.977 (4)
T-AB	0.645 (1)	0.546 (4)	0.599 (2)	0.527 (5)	0.396 (7)	0.515 (6)	0.272 (8)	0.581 (3)
D-IA	0.963 (2)	0.653 (7)	0.982 (1)	0.929 (3)	0.828 (4)	0.821 (5)	0.513 (7)	0.755 (8)
D-DA	0.972 (1)	0.913 (7)	0.960 (2)	0.940 (6)	0.958 (5)	0.957 (4)	0.752 (8)	0.953 (5)
D-DG	0.923 (3)	0.850 (7)	0.925 (1)	0.924 (2)	0.897 (6)	0.911 (4)	0.585 (8)	0.907 (5)
D-WA	0.603 (1)	0.505 (6)	0.554 (4)	0.557 (3)	0.356 (7)	0.545 (5)	0.269 (8)	0.578 (2)
AVG	0.85 (2)	0.79 (5)	0.85 (2.4)	0.82 (3.8)	0.77 (5.4)	0.82 (3.8)	0.63 (7)	0.79 (5.3)

demonstrate that we can obtain explainability with limited reduction of the performance, we did not evaluate advanced fine-tuning techniques and we limited ourselves to using a model fine-tuned on the EM task for the computation, which is a different task from the computation on the relevance scores. We are therefore convinced, and in this, we are supported by the literature (see for example the experiments in [130]), that there is room for improvement by perfecting the fine-tuning technique. The comparison with the results obtained with the Jaro-Winkler[7] syntactic similarity measure [142] shows the strength of the architecture: using a syntactic approach the performance decreases, but it does not happen as dramatically as you might expect.

The Decision Unit Scorer. This component is based on a fully connected neural network to provide a relevance score to the decision units. We calculate the effectiveness of the overall model after the substitution of the neural network with (1) a binary score that assigns 1 to the paired decision units and 0 to the unpaired; and (2) a simple scorer based on the cosine similarity of the embeddings of the tokens. The binary approach is also applied to the decision units created with the Jaro-Winkler distance, providing a baseline for the approach. The results are reported in the "Scorer" Section of Table 5.4.

[7]Similar results are obtained with other syntactic string similarity measures. We selected to show the results obtained with the Jaro-Winkler measure since this is a well known measure, performing well on many benchmark problems [20] and not so simple as the edit distance.

Table 5.5 Classifiers used as Explainable Matchers (F1 score). In bold, the best performance per dataset.

Dataset	LR	LDA	KNN	DT	NB	SVM	AB	GBM	RF	ET	Avg.	S.D.
S-DG	0.925	0.927	0.927	0.894	0.890	**0.936**	0.912	0.921	0.918	0.936	0.919	0.015
S-DA	0.982	0.971	0.984	0.949	0.963	**0.990**	0.977	0.968	0.977	0.985	0.975	0.012
S-AG	0.578	0.622	**0.625**	0.554	0.597	0.583	0.595	0.604	0.602	0.621	0.598	0.021
S-WA	0.721	0.709	0.632	0.639	0.572	0.720	0.686	**0.726**	0.716	0.719	0.684	0.050
S-BR	0.788	0.690	0.788	0.828	0.718	0.788	**0.839**	0.765	0.765	0.765	0.773	0.041
S-IA	**1.000**	0.852	0.906	0.912	0.787	**1.000**	0.947	0.947	**1.000**	0.982	0.933	0.067
S-FZ	0.977	0.977	0.977	0.977	0.977	0.977	**1.000**	0.955	**1.000**	**1.000**	0.982	0.014
T-AB	0.631	**0.645**	0.560	0.564	0.533	0.627	0.609	0.622	0.580	0.607	0.598	0.035
D-IA	0.909	0.760	0.760	0.871	0.696	0.830	**0.963**	0.931	0.881	0.877	0.848	0.077
D-DA	**0.972**	0.963	0.962	0.955	0.948	**0.972**	0.955	0.955	0.961	0.968	0.961	0.008
D-DG	**0.923**	0.922	0.902	0.893	0.887	0.918	0.898	0.913	0.919	0.921	0.910	0.013
D-WA	0.570	**0.603**	0.424	0.497	0.524	0.556	0.566	0.567	0.559	0.584	0.545	0.049
Avg.	0.831	0.803	0.787	0.794	0.758	0.825	0.829	0.823	0.823	0.830	–	–
S.D.	0.159	0.141	0.180	0.170	0.166	0.160	0.159	0.149	0.163	0.155	–	–

Discussion. We observe that the WYM relevance scores computed with the DL model perform better than the ones generated via the binary score. The same happens if we compare the results obtained with the DL model and the binary score of the Jaro-Winkler decision units. This suggests that a DL-based technique can effectively support the EM process. Moreover, the comparison between the results achieved with the Jaro-Winkler measure shows that the syntactic similarity does not convey enough semantics to assure a high level of effectiveness. Finally, we note that the results obtained in small datasets are not consistent with the ones in the other datasets. One of the reasons is the size of the training set, which is not enough large for training the model (e.g., the dataset *S-BR* contains 270 tuples in the training set).

The Explainable Matcher. This component generates the datasets to which the interpretable binary classifiers are applied. The datasets are created via feature engineering processes on the relevance scores. To evaluate the effectiveness of feature engineering, we perform an experiment with datasets composed of a limited number of features. In particular, the datasets contain 6 features, obtained by applying the count and the average operators on all the relevance scores, on the positive (the one pushing the decision toward the match), and on the negative scores (the one pushing towards the non-match). The F1 score achieved is shown in the "simplified feature" column of the "Matcher" Section of Table 5.4. Moreover, we perform a second experiment to compare the F1 scores achieved by all classifiers evaluated. The results of this experiment are reported in Table 5.5 (the last two rows and columns represent the average and the standard deviations on the datasets and classifier, respectively).

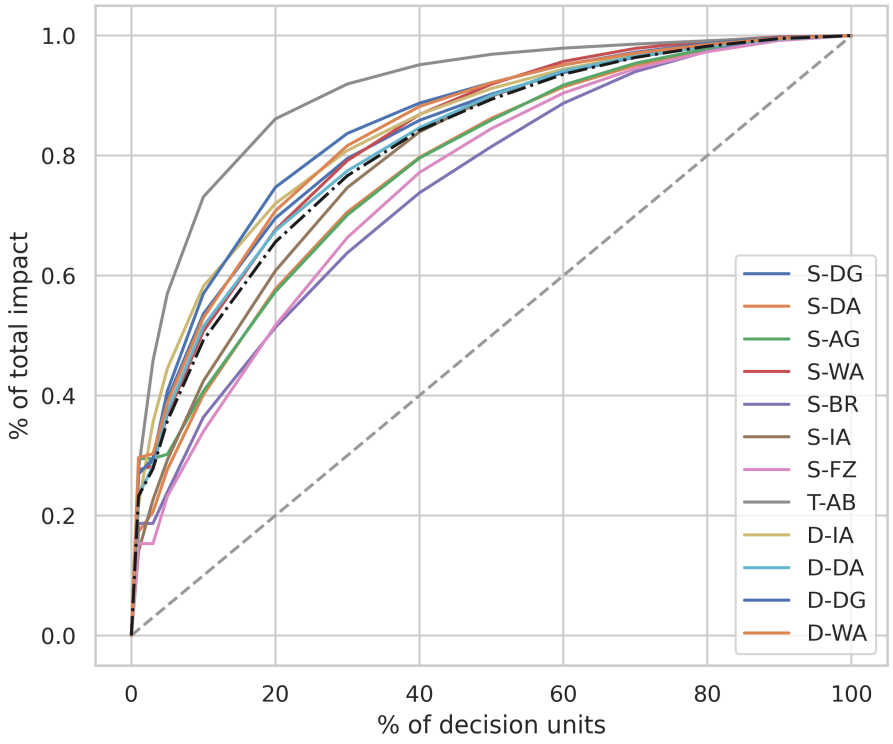


Fig. 5.6 Conciseness of the explanations.

Discussion. The first experiment shows the effectiveness of the WYM feature engineering process. Only in the dataset *S-BR*, the performance of the simplified is better than the one of the implemented component. The motivation is the same as in the previous experiment: when the dataset is small, the results suffer from variance. The second experiment shows that all classifiers obtain similar results (the standard deviation is generally low) and that the winner classifier depends on the dataset.

Take away. The overall ablation and robustness study presented in the Section shows that the components synergistically contribute to the generation of accurate results.

5.3.2 Interpretability of the EM Model

In this second set of experiments we investigate the interpretability of WYM by analyzing whether the impact scores can reveal the rationale behind the decisions. We evaluate this aspect in quantitative terms by analyzing the conciseness (Section 5.3.2), the faithfulless (Section 5.3.2), the contribution (Section 5.3.2) of the explanations and by comparing the WYM impacts with the explanations generated by a different tool (the Landmark EM explanation system – Section 5.3.2).

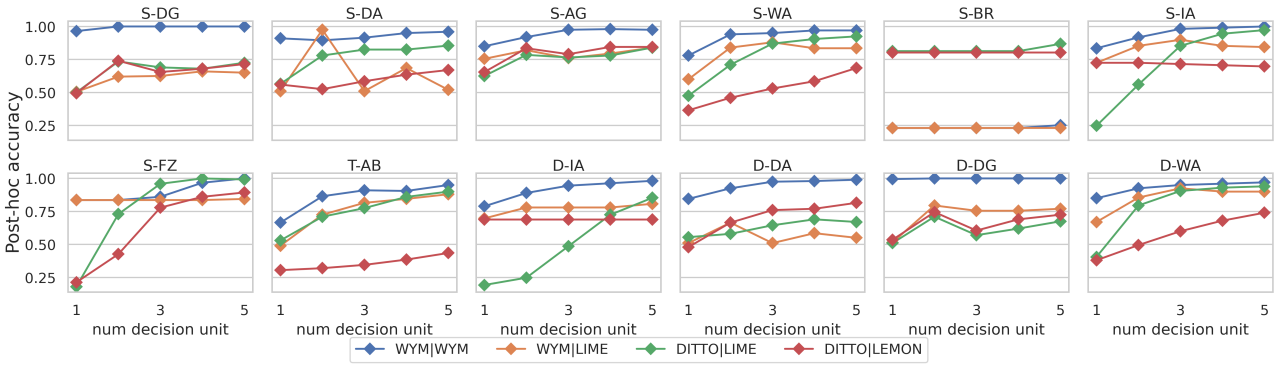


Fig. 5.7 Sufficiency evaluated with the post-hoc accuracy. To high values correspond accurate explanations.

Conciseness

The conciseness can make the explanations consumable for humans, who cannot analyze the dozens of decision units part of a dataset record. Therefore, an explanation is usable if it can describe the prediction with few elements. In Figure 5.6, we show the results of the Pareto analysis performed for each record in every dataset by ordering the decision units per impact in descending order and plotting the cumulative values.

Discussion. The Figure clearly shows the conciseness of the explanations. Even if we limit the analysis to a few decision units we can understand the behavior of the model. If the user analyzes 3% of the decision units, s/he get typically insight into the elements providing between 18% and 40% of the impact. 20% of the decision units provide between 50 and 83% of the impact.

Take away. The impact is concentrated in a limited number of tokens, making it easy for the user to discern among the decision units those that are most significant for prediction.

Faithfulness

We evaluate the faithfulness of the explanation to the EM Model via the notion of sufficiency, i.e., the ability of the top-impact elements to model a prediction [41, 6, 52, 80]. We adopt the post-hoc accuracy [31] as sufficiency measure. For each test data, we select the top v important units based on the impact attributions for the model to make a prediction and compare it with the original prediction made on the whole input text. We compute the post-hoc accuracy on M examples,

$$post-hoc-acc(v) = \frac{1}{M}\sum_{m=1}^{M} 1[y_v^{(m)} = y^{(m)}] \tag{5.4}$$

where $y^{(m)}$ is the predicted label on the m-th test data, and $y_v^{(m)}$ is the predicted label based on the top v important words. Higher post-hoc accuracy indicates better explanations. Figure 5.7 shows the results of the experiments by using up to the top 5 decision units. We compared the values obtained in four settings: we consider WYM evaluated as EM model and explainer, WYM evaluated as EM model and explained with LIME, *DITTO* explained with LIME and *DITTO* explained with LEMON using the single token granularity.

Discussion. The experiment shows that WYM used as an explainer provides more accurate results than the post-hoc LIME and LEMON systems in almost all evaluations. In the dataset S-BR, WYM performs significantly worse than *DITTO*[8]; in S-FZ WYM performs as *DITTO* , apart from some configurations.

Take away. The overall analysis of the sufficiency shows that the post-hoc explainer LIME cannot accurately explain WYM and *DITTO*, As a consequence, we observe that even if *DITTO* can generate more accurate predictions than WYM, the quality of the explanations generated by LIME is lower than the ones generated by WYM. Similar results are obtained by coupling DITTO with LEMON. Despite the accuracy obtained, in scenarios where the explanation is crucial, *DITTO* seems not to be the best solution.

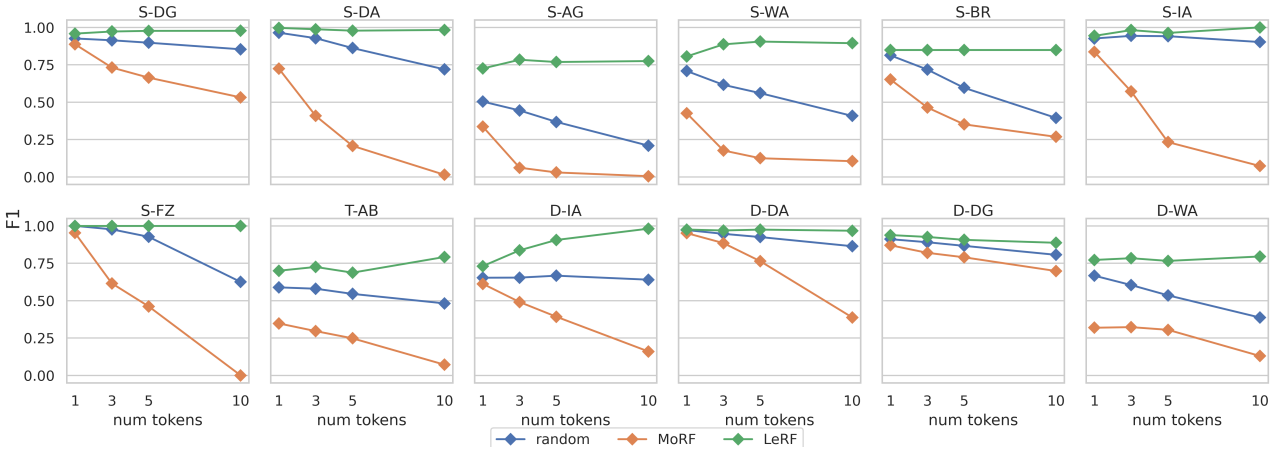


Fig. 5.8 F1 score obtained in EM Models after the removal of the most relevant decision units (MoRF), less relevant decision units (LeRF) and random units.

Contribution of the decision units to the overall accuracy

To evaluate the contribution of the impact scores assigned to the decision units to the overall accuracy of WYM, this experiment perturbs the dataset records by removing selected

[8]This is the smallest dataset, with a test set containing 91 records.

decision units and analyzing the performance variations on these synthetic datasets. This is an extension of the post-hoc evaluation presented in the previous Section.

We experiment with three techniques for the removal of the decision units applied to the datasets: 1) *MoRF*, where we eliminate for each record the k decision units that contribute most to the prediction (i.e. units with high positive impact in records describing matching entities and units with high negative impact for non-matching), 2) *LeRF*, where the k decision units that contribute less to the prediction are removed (i.e. high negative impact in case of entity matches and high positive impact / in case of non-matches), and 3) *Random*, where k random decision units are removed. We expect that when we remove the most relevant decision units (MoRF) from records describing matching entities, the effectiveness (F1 score) will decrease; on the other hand, the model should not be affected by the removal of the least relevant units first (LeRF). The results of the experiment are shown in Figure 5.8, where, for each dataset, the F1 score generated by WYM as the removal technique varies, is reported.

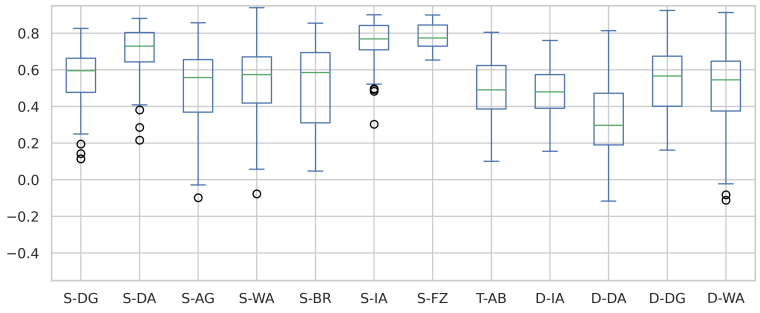
Discussion. Analyzing the results we observe how impact scores assigned by WYM reflect the real importance of each token on the prediction. By perturbing the data with the *MoRF* strategy, WYM performance drops drastically (up to 60% in some datasets). The phenomenon is mostly marked as the number of removed units increases, however, in some datasets (such as Abt-Buy, Amazon-Google, and the two versions of Walmart-Amazon) the performance drops after the removal of a single unit. Moreover, the *LeRF* perturbation does not produce substantial variations in the performance, which in most of the datasets slightly improves.

Take away. The experiment confirms the high quality of the explanations generated. The MoRF removal strategy confirms that decision units with a high impact score are crucial for model accuracy. The LeRF removal strategy shows that units with a low impact score do not contribute to generating accurate results.

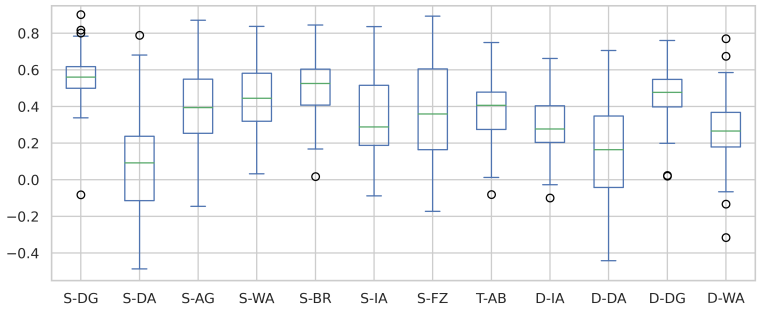
Correlation with Landmark

The WYM explanations are compared with the ones generated by *Landmark Explanation*[13, 12], a framework that extends the capabilities of a post-hoc perturbation-based explainer to the EM scenario. The experiment is performed by selecting a balanced sample of 100 elements from each benchmark dataset and using *Landmark* to compute the explanations[9]. Since it provides scores to tokens (and not to decision units), the explanations are post-processed by merging semantically similar tokens and averaging their scores. The outputs are then compared with the ones of WYM through the Pearson correlation measure. Figure 5.9

[9]We configure Landmark Explanation to generate 100 perturbations for each entity of an EM record



(a) Impact score match records



(b) Impact score non-match records

Fig. 5.9 Pearson correlation between the explanations generated by WYM and Landmark.

shows the results of the experiment, where the distribution of the correlation scores across all the records of each dataset is reported.

Discussion. The results show that concerning the descriptions of matching entities there is a moderate positive correlation between the scores provided by the approaches (the average Pearson correlation on all datasets is 0.577). The correlation is less marked if we consider the non-matching entities, where the average correlation is 0.348.

Take away. The low correlation in the case of non-matching predictions confirms that if two entity descriptions have many elements of diversity it is difficult to determine which ones mostly contribute to the decision. This is because we can find many subsets of them which could suffice for the model to determine a non-match prediction.

5.3.3 Time performance

We evaluated the time performance through two experiments. In the first experiment, we compute the overall time required to train the system and to compute the explanations. In

the second experiment, we analyze the pipeline time breakdown to evaluate the components where most of the time is spent.

Take away. The experiments show that the training time is similar to the one of DITTO (the average throughput is around 9 records per second). The average throughput in the generation of the explanations is around 20 records per second. This time is one order of magnitude lower than DITTO (around 130 records per second). Nevertheless, DITTO does not compute the explanations. The analysis of the pipeline breakdown shows that the time spent on making the explanations is around 40% of the overall time. The overall analysis shows that WYM can be used for computing the predictions and explanations in typical real-world scenarios (on average it can generate 70k+ explanations per hour).

5.3.4 Users evaluation

A small scale users evaluation is performed to understand the quality of the decision unit-based explanations and to assess if this type of explanation is useful to understand the behavior of the EM system. A questionnaire is administered to 15 users (colleagues and students from the ICT doctorate course at the University of Modena and Reggio Emilia). The participants are required to evaluate the feature-based explanations generated with DITTO and LIME and the decision unit-based explanations generated with WYM. Three pairs of entity descriptions were shown. The first pair of entity descriptions is referring to the same real entity; the second pair refers to different entities; the third pair is composed of the same description copied twice.

Take away. The experiment shows that the users usually prefer decision unit-based explanations. Only when the entity descriptions are really close (as in the third pair proposed in the questionnaire) users were satisfied also by the feature-based explanations. Even the small scale of users involved, the Fleiss' kappa was 0.787, thus showing good agreement among the participants.

5.4 Conclusion

In the paper, we presented an architecture template for performing interpretable entity matches. The architecture is based on three components, predicts if a pair of entity descriptions refer to the same real-world entity, and provides the terms (i.e., the decision units) that mainly led to the decision. We described our implementation WYM which has experimented with the datasets usually adopted for evaluating EM approaches. The results show that WYM

has performance similar to the ones of other state-of-the-art techniques, but the predictions can be explained.

We have identified two main directions towards which to focus our future work. From one side, we will investigate the introduction of external knowledge in the approach (in the form of synthetic sentences automatically generated and rules on decision units) to improve the effectiveness and the quality of the explanations. From the other side, we will experiment if decision units can be effectively used to train DL-based EM systems coupled with post-hoc explainers.

Chapter 6

Explaining The Role of Evidence in Data-Driven Fact Checking

In Chapter 5, we presented an intrinsically interpretable Entity Matching architecture, emphasizing the importance of transparency and interpretability in machine learning models applied to textual data. While interpretability is crucial in Entity Matching, it becomes even more critical in sensitive tasks such as computational fact-checking, where the decisions made by automated systems have significant implications for public trust and societal well-being.

Automated fact-checking methods have been proposed to address the scalability challenge of verifying the rapid content influx on social media, offering cost-effective alternatives to human-based methods. Computational fact-checking involves establishing the relationship between a textual claim and a body of evidence to determine if the evidence supports, refutes, or is insufficient [114]. Evidence can be structured tabular information, unstructured text, or both [59]. Evidence-based solutions are the most common and effective approaches to this problem [117].

Such computational solutions rely on a ML pipeline, where, given a claim to check, a first model (*retriever*) identifies relevant evidence from a corpus of documents and a second model (*verifier*) evaluates the veracity of the claim using such evidence. Table 6.1 illustrates an annotated example from a fact-checking dataset. The retriever returns both evidence that supports the claim (two sentences in green) and refuting evidence (the one in red, stating that the production rights and certification are held by *Wei Hang*, not *Windecker Industries*). It also retrieves noisy content that is not useful to the verifier's final decision (in gray).

Correct labeling is crucial but insufficient for combating misinformation. Most solutions act as black-boxes: they provide accurate inferences but do not explain the reasoning behind the decisions. However, fact-checkers reject solutions that do not expose the evidence

CLAIM	The Eagle AC-7 Eagle 1 is a military aircraft that was manufactured by Windecker Industries.
EVIDENCE	The Eagle AC-7 Eagle 1 (USAF designation YE-5) is an aircraft. Windecker Industries was an American aircraft manufacturer founded in 1962. It was manufactured by Windecker Industries. [...] Wei Hang holds the rights and the type certificate to produce the aircraft.
LABEL	SUPPORTS

Table 6.1 Annotated example from the FEVEROUS dataset with a claim, its retrieved evidence (supporting in green, noise in gray, and refuting in red), and the corresponding ground truth label.

justifying the model's output [94]. Transparency is essential for users to verify the evidence and decide if they agree with the model [59].

Our goal is not just to identify relevant evidence for the given claim, as done by re-rankers in the retrieval stage [115], but to determine how pieces of evidence influence the verifier's claim classification. Our solution aims at labeling evidence similarly to the colored annotations in Table 6.1. Inference is not enough: it is essential to explain the reasoning behind a verifier by determining the key piece of evidence in supporting, refuting, or stating that a claim lacks sufficient information for verification. While some fact-checking models can provide useful or noisy labels for evidence, none are capable of indicating which specific evidence led the model to lean towards a *Refutes* decision. In this direction, we propose a framework that explains the workings of a fact-checking model by scoring individual pieces of evidence based on their contribution to the decision-making process.

Our framework enables users to better understand the results of a fact checking tool. In particular, we derive explanations of the examples that allow us to address the following three questions:

- Q1: What are the evidence pieces that steer the decision? Can the verifier discern noise, identifying pieces of evidence that are important in the claim evaluation?
- Q2: To what extent do the verifiers rely on the evidence and the claim to perform the decision? Is this reliance determined by whether a claim is supported, refuted, or deemed to have insufficient evidence?
- Q3: How is distributed the impact of the different pieces of evidence on the verifier? Do verifiers use all the evidence?

This work explores these questions using post-hoc methods explanations methods (Section 2) over four popular datasets and three verifier models, two of which have their own

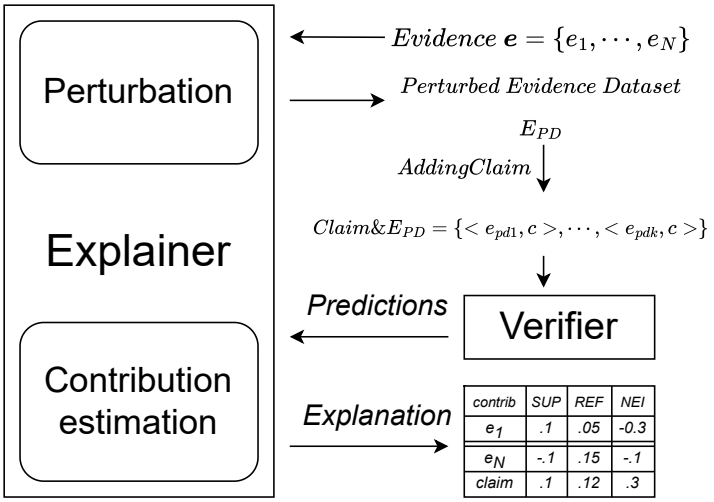


Fig. 6.1 The explanation framework.

intrinsic explainer (Section 3). Our experimental campaign demonstrates that post-hoc methods enable analysis of every verifier, including those based on fine-tuning pre-trained language models (PLMs), producing evidence-based justifications of similar quality to those generated by intrinsic explainers (Section 4).

6.1 The Framework

The Fact-checking Problem. We define evidence $\mathbf{e} = \{\mathbf{e}_s, \mathbf{e}_t\}$ as a non-empty subset of sentences $\mathbf{e}_s = \{s_1, \ldots, s_{|\mathbf{e}_s|<n}\}$, where n is the total text count, and cell values $\mathbf{e}_t = \{t_1, \ldots, t_{|\mathbf{e}_t|<p}\}$ arising from a table, where p is the total cell count. A supervised verifier model $\mathscr{F}$ assesses whether a textual claim c finds support or contradiction w.r.t. a given evidence $\mathbf{e}$. Formally, $\mathscr{F}$ takes as input a data pair $<\mathbf{e}, c>$ and outputs a verdict in the space $\mathscr{L}$. We name *gold evidence* the evidence selection from the fact-checking dataset. Those evidence pieces are the clean ground truth of evidence to be retrieved. They are required to deduce $\mathscr{L}$, in contrast with *noisy* evidence. In this work, we consider two settings. In the binary (2-label) setting $\mathscr{L} = \{Supports, Refutes\}$, while in the full (3-label) setting $\mathscr{L} = \{Supports, Refutes, Not\ Enough\ Information\}$. The *Not Enough Information* (NEI) label is associated with cases where in the corpus there is not enough evidence to state if the claim is supported or refuted. Some verifiers focus on the binary case because datasets with three labels are rare, e.g., NEI examples are only 3% in FEVEROUS.

Our Proposal. Figure 6.1 shows our framework[1] to understand the *contribution* of each evidence in predicting the label for a claim. It adapts a local post-hoc explainer (LIME and SHAP in our study) to evidence and claim. Local post-hoc explainers rely on "variations" of the input to compute the contributions of the features (every example's evidence pieces) in the decision (for a claim). In our framework, a perturbed dataset is generated for the example by removing pieces of evidence. The explainers use the predictions of the verifier over the perturbed dataset coupled with the original claim to build the explanation. It assigns for each example a contribution to each feature (evidence) and to an *intercept*, defined as the average difference between the overall prediction and the total contribution of all individual evidence in the perturbed dataset. This can be interpreted as the contribution that the explainer attributes to the claim within the context provided by the evidence. The prediction is thus approximated expressed as a function of the contributions defined by the explanation:

$$\text{Pred} \approx \sum_{i=1}^{N} \text{contribution}(e_i) + \text{Intercept} \tag{6.1}$$

where contribution(e_i) (contribution of evidence i) is the contribution over the model prediction detected by an explainer for an evidence.

6.2 Datasets and Models

We first introduce the datasets in our study and how we inject noise into the evidence set for each example. We then present the different models in our study and the configuration of the explainers.

6.2.1 Datasets

We select four fact-checking datasets. All examples consist of a claim written by a human, a label, and the golden evidence used to classify the claim. Statistics about the datasets after pre-processing are in Table 6.2. When the original test set is private, we use the original validation set as test set.

Feverous [3] is an extension of Fever [133] with more complex claims. Claims are crafted by humans from textual and tabular evidence from Wikipedia. We linearize tabular evidence in the format: $Cell_{Value}$ <*context*>$Cell_{Header}$</*context*>. We create variants of the dataset with and without the label *'Not Enough Information'*, namely **Fev.3L** and **Fev.2L**.

[1]The code is available on the project GitHub https://anonymous.4open.science/r/explainable-fact-checking-268D/

Dataset	#Claim	T.	|Claim|	#Ev.	#Noise
Fev.3L	71.2k(27.2k/2.2k/41.8k)	train	25.3	1.6	16.8
	7.9k(3.5k/0.5k/3.9k)	test	24.9	1.4	17.6
Fev.2L	69k(27.2k/0/41.8k)	train	25.3	1.6	16.8
	7.4k(3.5k/0/3.9k)	test	24.9	1.4	17.5
SciFact	0.8k(0.2k/0.3k/0.3k)	train	12.3	1.3	9.0
	0.3k(0.1k/0.1k/0.1k)	test	12.5	1.3	8.7
AVTC	2.8k(1.7k/0.3k/0.8k)	train	17.1	2.6	5.6
	0.5k(0.3k/0.04k/0.1k)	test	14.4	2.6	5.5
FM2	10.4k(5.3k/0/5.1k)	train	13.7	1.3	9.1
	1.4k(0.7k/0/0.7k)	test	13.8	1.3	9.1

Table 6.2 Statistics of datasets used for training and testing the fact-checking models. The number of claim details stand from left to right for *'Supports'*, *'Not Enough Information'*, and *'Refutes'*.

Dataset	Precision	Recall	F1	Sufficient
Fev. 2L	0.07	0.36	0.12	✗
Fev. 3L	0.07	0.36	0.12	✗
SciFact	0.13	1	0.23	✓
AVeriTeC	0.32	1	0.48	✓
FM2	0.12	1	0.22	✓

Table 6.3 Performance metrics for the retrievers used to build every dataset (Precision, Recall, F1).

SciFact [137] contains expert-written claims paired with evidence from scientific papers. The evidence is from textual sources only.

FM2 [48] is obtained from an online multiplayer game where users write claims from a list of evidence from Wikipedia. To gain points, claims must be hard to fact-check by other players. Evidence is only textual and the *'Not Enough Information'* label is not present.

AVeriTeC (**AVTC**) [116] contains real-world claims to verify with Web evidence. Each claim has evidence in the form of question-answer pairs supported by online content. We treat each pair as one textual evidence. This dataset has a fourth label, *'Conflicting Evidence/Cherry-picking'*, we do not report it in our results as no verifier returns it.

Controlled Noisy Evidence. Datasets come with golden evidence for every claim. However, we must include noisy evidence in the examples to enable our experiments. Whenever possible, we obtain the evidence with retrievers, as this is how they arise in practice. When

no corpus is available, we mix the gold evidence with noise. We show an overview of retrieval performances in Table 6.3. We next detail how we add noise to every data set.

For Feverous, the corpus of Wikipedia pages and the retriever are provided in the pipeline, so we run it on the train and test datasets to obtain their noisy versions. We retrieve 5 documents per claim, and in each document 5 sentences and 3 tables. The retriever selects an arbitrary number of cells per table. Crucially, gold evidence is missing after the retrieval step (0.36 Recall) and a lot of noise is selected in the dataset (0.07 Precision). In the Feverous datasets, recall is lower than 1, i.e., some golden evidence are not picked by the retriever. This may lead the labels of the verifier to be less accurate, as it may lack sufficient information to verify the claim.

SciFact authors included for each example five "distractor abstracts" that cover topics mentioned in the original article. We append sentences from these abstracts to the original evidence, up to a total of 20 sentences.

To write a claim, FM2's players use one to two sentences out of ten sentences from Wikipedia, the remaining sentences are used as noise.

In AVeriTeC, gold evidence are human-created question-answer pairs. The authors provide a question-answer generator to obtain retrieved evidence. From the generator, we pick the least relevant pairs measured by BM25 against the claim. We add an average of 5.5 pairs per example.

6.2.2 Models and their training

For the verifier models, the objective is to infer a label from a claim and the retrieved evidence. Before testing models on the test set, we fine-tune them on the corresponding train set.

RoBERTa [85] is a transformer-based language model. We fine-tune the model on relevant NLI datasets [95] as in Feverous, with learning rate $1e^{-5}$ for the dataset with 3 labels, and $1e^{-7}$ for the datasets with 2 labels. We run the training for 1 epoch on Feverous and AVeriTeC, and 3 epochs on FM2 and Scifact.

GFCE [7] jointly trains veracity prediction and explanation generation using a fine-tuned version of DistilBERT. It leverages both claim texts and supporting evidence to produce justifications. We use a learning rate of $1e^{-5}$ for every dataset. We train FM2, Feverous and SciFact on 3 epochs, and AVeriTeC on 2.

We report an LLM-prompting verifier based on LLaMa 3.1 70B [42]. The prompt includes the description of the task and the claim with its evidence. The model outputs a veracity label as well as a list of the evidence used to make the decision. The inference takes up to 2 hours for a thousand predictions. To enable analysis with SHAP and LIME, we also use an alternative prompt without the evidence-labeling task, reducing both input and output

lengths, and consequently the running time. We observe a comparable behavior with this prompt in claim labeling, but with inference time reduced by 75%.

6.2.3 Configurations of the Explainers

We use LIME and SHAP to analyze the verifier outputs. For the GFCE and RoBERTa models, we set 500 perturbation samples per explanation, while for LLaMa, we reduced it to 100 because of its computational cost. We explain a subset of the datasets: we exclude examples exceeding 512 tokens (too long for RoBERTa) and sample up to 1,000 examples from each dataset.

6.3 Experiments

We address the research questions presented in the introduction. For Q1, we start by investigating the ability of SHAP/LIME-based explanations to distinguish Noise from Useful Evidence by contrasting them against intrinsic explainer models. Then, for Q2, we analyze how every model uses the input evidence and the claim in outputting a label. Finally, to address Q3, we investigate the importance given by verifiers to the useful evidence for each label individually.

6.3.1 SHAP and LIME based Noise Detection

We measure the ability of SHAP and LIME to act as an evidence labeling system when used to observe a verifier model. To assess the quality of our proposal, we measure it against explanations provided by intrinsic explainer models (GFCE and LLaMa), that output both a label for the claim and for each input evidence piece.

Evidence Labeling GFCE and LLaMa output a binary label for each evidence piece, together with an *importance in decision* score. However, this information does not label the evidence piece in terms of their role in the decision, i.e., if they steer the verifier towards a refutes or supports decision.

For SHAP and LIME, we model the noise detection problem as a binary classification task using the sum of absolute contributions of the explanations as score. We hypothesize that any noisy evidence should leave the prediction unaltered, i.e., the explainer recognizes that a noisy evidence does not change the state of the claim. From the methods, we obtain scores for every evidence piece relative to its importance in a claim label decision. To label each evidence piece as noise or useful, we infer a threshold that separates them. To get its best

values for each dataset and observed model, we use the explanation scores for 100 evidence pieces from the train set executed on the current model. The training set includes gold labels indicating the usefulness of each evidence. We choose the threshold that maximizes the sum of the F1 scores for useful and noisy evidence. We then label each evidence as Useful or Noise using such threshold.

Table 6.4 Performance comparison across different models and datasets in terms of claim verification (left), explanations from the intrinsic explainers (middle), and explanations from the post-hoc explainers (right). Best explanation result for every dataset and observed model pair in **bold**.

DS	Model	Verifier				Intrinsic Explainer			SHAP/LIME		
		Acc.	F1 Sup	F1 Nei	F1 Ref	Acc.	F1 Useful	F1 Noise	Acc.	F1 Useful	F1 Noise
Fev.2l	Roberta	0.61	0.73	-	0.35	-	-	-	**0.88**/0.83	0.24/**0.30**	**0.93**/0.90
	GFCE	0.48	0.03	-	0.64	0.78	0.31	0.87	0.56/**0.83**	0.27/**0.40**	0.69/**0.90**
	LLaMA	0.69	0.70	-	0.68	**0.88**	**0.47**	0.93	**0.88**/0.86	0.21/0.20	**0.94**/0.92
Fev.3l	Roberta	0.60	0.71	0	0.41	-	-	-	**0.89**/0.88	0.35/**0.40**	**0.94**/0.93
	GFCE	0.59	0.69	0	0.46	0.79	0.33	0.88	0.80/**0.82**	**0.36**/0.34	0.88/**0.89**
	LLaMA	0.57	0.70	0.17	0.44	**0.88**	**0.46**	**0.93**	0.83/0.84	0.33/0.26	0.90/0.91
SciFact	Roberta	0.67	0.69	0.75	0.49	-	-	-	0.83/**0.86**	**0.28**/0.23	0.91/**0.92**
	GFCE	0.37	0.52	0	0.23	0.38	0.20	0.50	**0.66**/0.58	**0.21**/0.20	**0.79**/0.72
	LLaMA	0.76	0.80	0.71	0.72	0.80	**0.51**	0.88	**0.87**/0.85	0.29/0.26	**0.93**/0.92
FM2	Roberta	0.70	0.72	-	0.68	-	-	-	0.87/**0.88**	**0.27**/0.16	**0.93/0.93**
	GFCE	0.58	0.59	-	0.56	0.60	**0.32**	0.71	**0.87**/0.83	0.09/0.20	**0.93**/0.90
	LLaMA	0.87	0.88	-	0.87	0.85	**0.58**	0.91	**0.88**/0.85	0.40/0.38	**0.93**/0.92
AVTC	Roberta	0.79	0.86	0.52	0.78	-	-	-	**0.78/0.78**	**0.49/0.49**	**0.86/0.86**
	GFCE	0.59	0.50	0.50	0.69	0.37	**0.49**	0.17	**0.70**/0.69	0.48/0.44	**0.79/0.79**
	LLaMA	0.76	0.77	0.44	0.85	**0.88**	**0.81**	**0.92**	0.77/0.75	0.63/0.54	0.84/0.83

Useful Evidence Identification Table 6.4 presents the results of the experiments. Each row represents a dataset and the verifier applied to it, representing the observed model for our approach. The columns under *Verifier* show the associated performance of each verifier on claim labeling. The columns under *Intrinsic Explainer* report the Accuracy, F1 Useful, and F1 Noise for labeling evidence for the verifiers that can justify their decisions, i.e., GFCE and LLaMa. The columns under *SHAP/LIME* report the same metrics using the post-hoc methods. All scores for a single dataset are computed by running the models on the associated test set - both for claim and evidence labeling.

Overall, SHAP provides more accurate results than LIME, independently of the metric analyzed. SHAP wins in 62%, 69%, and 54% of the cases when looking at Accuracy, F1 Useful, and F1 Noise, respectively. For most dataset/model pairs, the SHAP and LIME-based

explainers perform on par with the intrinsic explainer - GFCE or LLaMa - regardless of the metric. Post-hoc explanation methods applied on GFCE verifier outperform its intrinsic explainer in 13 out of 15 cases. If we compare across all explainers for a single dataset (both intrinsic and post-hoc), the accuracy of evidence labeling metric for SHAP/LIME is the highest on 4 out of 5 datasets. In noise detection and accuracy, post-hoc methods outperform the intrinsic ones in all cases except two, while LLaMA shines in terms of identification of useful evidence. The results show that post-hoc explainers compete with models specifically trained on this task. Finally, our proposal is the only one able to explain a Roberta model, which lacks an intrinsic explainer.

Restraining to explanations based on SHAP and LIME, in all cases, the worst accuracy for evidence labeling is obtained when the observed model is the GFCE's verifier. This is consistent with the performance of the verifier, that is, on average for all datasets, 14 absolute points under the worst scores among the two other verifiers.

When examining the results for the same model and dataset, we observe that GFCE's explanations have scores that are not consistent with the verifier's scores. The origins of this behavior rely on the separation of the explainer and verifier in this model. Indeed, even if they are trained with a joint loss, the GFCE's explainer predictions are not directly linked to the verifier's decisions.

Being the performance of SHAP and LIME highly dependent on the verifier's performance, we would expect post-hoc models to struggle on evidence labeling for GFCE. However, most results for post-hoc explainers for GFCE are even better than the intrinsic alternatives. Such results show that the GFCE's verifier is capable of accurately detecting noise, but the verifier is not using the evidence in its decision-making process. The analysis in Section 6.3.2 supports our reasoning.

The experiments show that post-hoc methods are able to discern noise. In some situations, they identify noise better than intrinsic explainers tailored for this task. By extension, we can derive that fact-checking models effectively distinguish noisy evidence. However, the results are significantly lower in the detection of useful evidence. LLaMA intrinsic outperforms competitors in F1 useful while the best post-hoc counterpart is 19.4 points below on average on this metric. This highlights that the LLaMa intrinsic explainer is not directly connected to its veracity prediction. The explanation generated intrinsically may not represent how evidence pieces are used for veracity prediction, thus not being faithful.

Q1 Take-away: Post-hoc methods effectively discern noise, often better than intrinsic explainers, indicating that fact-checking models can identify irrelevant evidence, yet struggle with recognizing useful evidence.

6.3.2 Model Reliance on Claim and Evidence

We analyze how the verifier model depends on the (gold) evidence, noise and claim to make decisions over (1) whether the model relies on the claim and the evidence for predictions, (2) how contributions are distributed among the evidences.

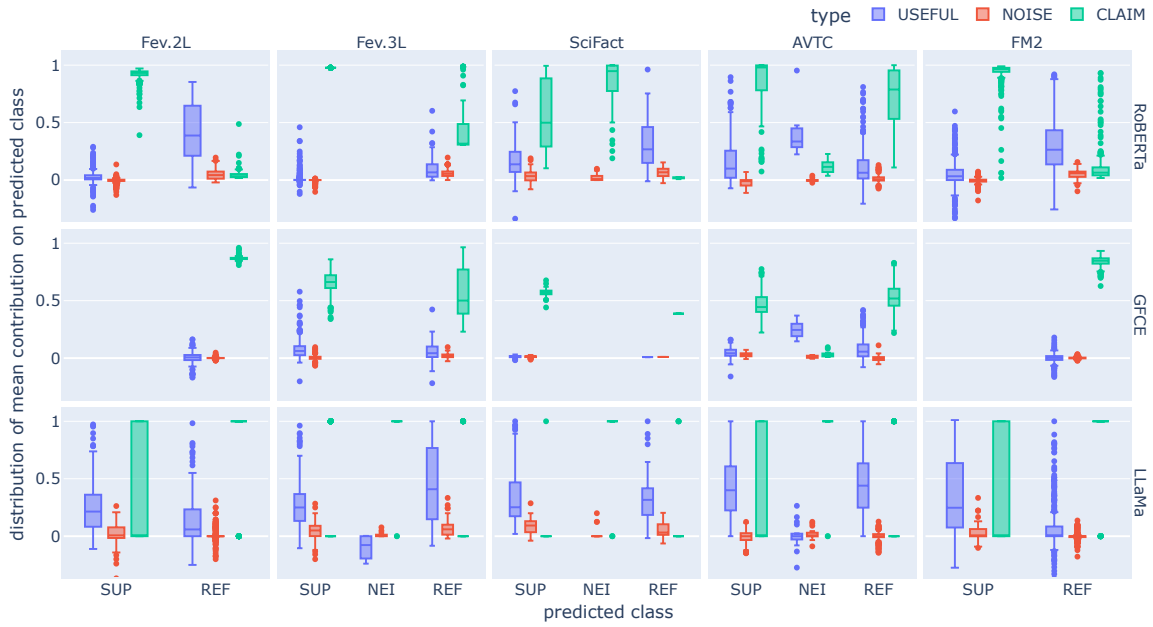


Fig. 6.2 Distribution of contribution scores obtained with SHAP of the useful evidence, noisy evidence, and claim on the predicted classes (Supports, Refutes, NEI). We spread horizontally the five datasets and vertically the three observed models. Every plot reports the predicted class over the x and the prediction contributions for useful evidence, noise, and the claim itself.

Importance of claim and evidence. We analyze separately the examples labeled by a model as *Supports*, *Refutes* or *Not Enough Information*. For each example, we compute three contributions: the one from the useful evidence (obtained averaging the contributions for all useful evidence pieces of an example), the one from the noisy evidence (by averaging them as well), and the one from the intercept (the contribution from the claim). The distribution of these contributions for all examples in the test set is shown in Figure 6.2 as USEFUL (evidence), NOISE (evidence), and CLAIM. Computing the average contribution for useful and noise lets us analyze their contribution independently from their unbalance in quantity. Even though models may assign greater contributions to useful evidence than over noise, in some experiments the higher number of noisy evidence makes the sum of their contribution greater than the contribution of the useful evidence.

We report the analysis on every dataset and every verifier. The behavior is consistent between SHAP and LIME, so we report results for the former, as it performs slightly better in the previous study and we include the overall results in the Appendix. The plots show the values obtained on the subset of the test sets where the verifiers are correct in the label prediction. This ensures a fair comparison independent of model accuracy disparities. The behavior is consistent when plotting the entire test set, as reported in the Appendix.

Figure 6.2 shows that in all experiments the average contribution of useful evidence in claim labeling is greater than that from noisy evidence. The exception is for label *NEI*, where noise contributes as much as useful evidence. For this label, verifiers predominantly exploit the claim. This behavior indicates a lack of sufficient useful evidence, coherently with the results in Table 6.4.

In RoBERTa, for Fev. 2L most of the contribution to the *Refutes* label comes from the useful evidence. In the *Supports* case, the median of the contributions of the claim is the only score deviating from zero, i.e., the claim is the only factor that contributes to the claim labeling. This may suggest that the model labels a claim as *Supports* when it cannot find any evidence to refute it. On the Fev. 3L, the model displays a similar behavior as on Fev. 2L. In the AVeriTec dataset, RoBERTa predominantly bases its decision on the claim across all labels. We observe the same behavior for GFCE. This reliance on the claim, rather than on useful evidence, undermines the verifier's trustworthiness, as reflected in the non-excellent results in claim labeling in Table 6.3.

For LLaMa, the post-hoc methods show that the majority of the contributions always come from the evidence. However, among the three verifiers, it is the only one that does not rely on evidence to label AVeriTec *Not Enough Information*.

Comparing LLaMa's behavior on datasets with 2 and 3 labels reveals a noteworthy shift. The introduction of the *NEI* label enhances the contribution of the evidence for the *Refutes* class. This indicates a transition from an implicit *Refutes* logic to a more nuanced implicit *NEI* reasoning, which appears appropriate and logically sound.

Take-away Q2: Using post-hoc analysis, we uncover that verifiers rely differently on evidence and claims to make decisions. This approach reveals significant differences in how models actually weigh and utilize these elements in their decision-making process.

6.3.3 Contribution Spread on Evidence

We measure whether the contributions are evenly distributed across evidence pieces. We rank the evidence by decreasing contribution in all examples. Then, we compute the mean for all the evidence with the same rank.

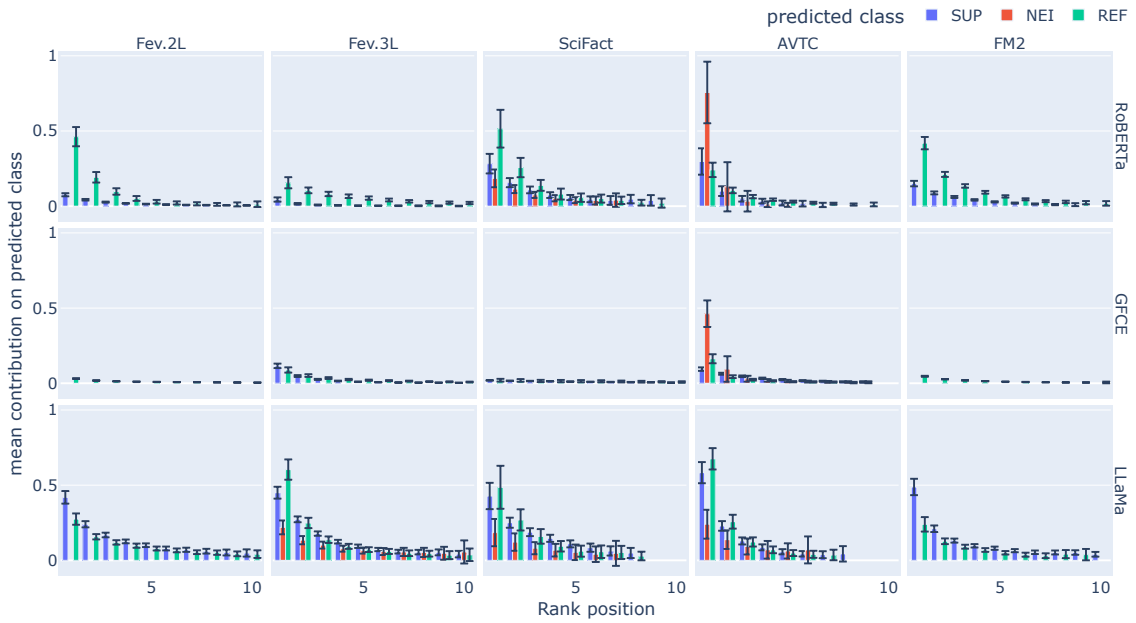


Fig. 6.3 Mean contribution score on the predicted class for the evidence pieces in an example. Scores are grouped by predicted class and in descending order, removing evidence pieces contributing negatively to the predicted class. Experiment run on each verifier (rows) and each dataset (columns), scores obtained with SHAP.

Results in Figure 6.3 confirm the observations for Q2: the verifiers rely on evidence very differently. GFCE is lightly influenced by the evidence, with a maximum contribution of the useful evidence of 0.2 across every dataset on the *Supports* and *Refutes* labels. In cases where the predicted label is *Not Enough Information*, GFCE and RoBERTa on AVeriTeC show a high contribution score for the most useful evidence, indicating that the models switch their decision to *NEI* based on a single piece of evidence. It is surprising that adding evidence causes the model to switch its prediction toward a lack of information. This may suggest that the verifier did not correctly learn the implications of the NEI label, likely due to the small percentage of such claims in the training data

Next, we look at contribution ranking for evidence across models and datasets. LLaMa seems to use evidence most effectively, as it distributes the contribution scores more evenly across the available evidence for both *Supports* and *Refutes* than the other verifiers. LLaMa is also the only one using the evidence in every dataset – with the first contribution always around 0.5. This is surprising for two reasons. First, LLaMa has not been fine-tuned for the fact-checking task, in contrast with RoBERTa and GFCE. Second, the LLM's extensive knowledge base could enable it to disregard the provided evidence. Our explanation is that

this behavior comes from the large size of the model (70B), as the same model with only 8B shows lower results (Table 6.6 in Appendix).

Finally, looking at the results by label, the evidence play a key role for RoBERTa when it predicts *Refutes*, while this behavior is less evident for *Supports*. On the other hand, LLaMa exhibits a more balanced performance between the two labels. In GFCE, the low contribution of the evidence involves all labels.

Take-away Q3: Post-hoc analysis reveals that evidence impact is unevenly distributed across verifiers, with models primarily focusing on select evidence pieces rather than utilizing all available information equally.

6.4 Experimental Setup

The experiments were conducted on a system running Ubuntu 20.04.6 LTS with a kernel version of 5.4.0-177-generic. The hardware configuration consisted of an AMD EPYC 7272 12-Core Processor with 128GB of RAM, and a NVIDIA GeForce RTX 3090 GPU with 24GB of memory. The software setup included Python 3.8.10, LIME 0.2.0.1, SHAP 0.45.0. To run inference on LLaMa 3.1 70B, we used a more powerful server running Ubuntu 20.04.6 LTS with a kernel version of 5.15.0-122-generic. The hardware configuration consisted of an AMD EPYC 7742 64-Core Processor with 512GB of RAM, and a NVIDIA A100-SXM with 80GB of memory. The software setup included Docker 23.0.3 where we ran in a container LLaMa 3.1 using Python 3.9.

6.4.1 Configurations of the Explainers

To analyze the outputs of the verifier models, we utilized two explainability methods: LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations). These methods were employed with consistent configurations across different models, except for LLaMa 3.1 70B, which required adjustments due to computational constraints. For both the GFCE and RoBERTa models, we used 500 perturbation samples to generate each explanation. This number of samples ensures explanation stability and fidelity to the model's behavior. Given that the number of explanation units (evidence) is not greater than 10 it follows that we are always covering at least half of all the possible combinations or evidence. Considering that explainers weigh more samples close to the original item the chosen number of perturbations guarantees fidelity. However, for LLaMa 3.1 70B, generating explanations was more time-consuming due to the model's larger size and complexity. To accommodate this, we reduced the number of perturbation samples to 100. This reduction allowed us to obtain results in a reasonable time frame without sacrificing too much on the quality of explanations. Another important adjustment was related to the number of examples we explained from each dataset. We first filtered out examples that were too long for RoBERTa (over 512 tokens), as these would exceed the model's input length limit and make the explanations less comparable. From the remaining examples, we selected up to 1,000 examples to explain for each dataset. Some datasets had fewer than 1,000 examples available for explanation after filtering, in which case we explained all the available examples. This approach ensured that the explainers were applied to a manageable and representative subset of the data.

To provide a clearer overview of the time performance of the explainers across different models and datasets, we summarizes the mean time required per explanation in Table 6.5.

This Table highlights the variations in explanation time based on the model and dataset combinations, with particular attention to the computational differences observed for LLaMa 3.1 70B due to its larger size and complexity, as compared to GFCE and RoBERTa.

Table 6.5 Performance comparison across different models and datasets in terms time per explanations. mean (std)

exp	Dataset Model	AVTC	FM2	Fev.2L	Fev.3L	SciFact
LIME	GFCE	60.96 (24.7)	106.61 (1769.2)	44.58 (126.0)	44.17 (10.3)	19.75 (2.3)
	LLaMa	40.65 (10.0)	45.55 (8.1)	29.80 (19.5)	42.63 (16.5)	33.29 (13.1)
	RoBERTa	9.74 (3.3)	7.05 (1.6)	8.96 (3.0)	8.05 (3.0)	6.34 (1.9)
SHAP	GFCE	65.72 (22.8)	66.76 (13.9)	51.33 (14.1)	51.32 (13.9)	55.49 (17.3)
	LLaMa	45.37 (11.2)	56.04 (13.7)	51.93 (18.3)	53.28 (23.5)	31.70 (17.4)
	RoBERTa	5.87 (3.5)	6.37 (1.7)	5.91 (3.6)	6.11 (3.8)	3.48 (3.0)

6.5 Results and Discussion

6.5.1 Importance of claim and evidence

As promised in the experiments, we provide comprehensive results on the contribution of evidence and claims across the entire test set. Figure 6.4 presents these findings. Additionally, we report results using the LIME explainer in Figure 6.5 to offer an alternative perspective on feature importance. The new figures corroborate our initial interpretation of the contribution distribution, reinforcing our conclusions.

6.5.2 More details on GFCE

We worked to find the best settings for each verifier to make a fair comparison. This sometimes meant changing how we used the training data. For the GFCE model, we had to figure out the best training settings on our own, since the creators of GFCE didn't give specific instructions for this. In particular, we reduced the imbalance in AVeriTeC at train time by equalizing the number of *'Supported'* and *'Refuted'* claims. On the other hand, GFCE's training is not impacted by the skew in SciFact: we leave it as is.

6.5.3 More details on Feverous

The pipeline implemented to verify a claim in [4] is the following. First, starting from a claim, relevant pages are retrieved using TF-IDF. Then, inside each of those pages, the application of TF-IDF allows to find the relevant sentences. Concurrently, a retriever model,

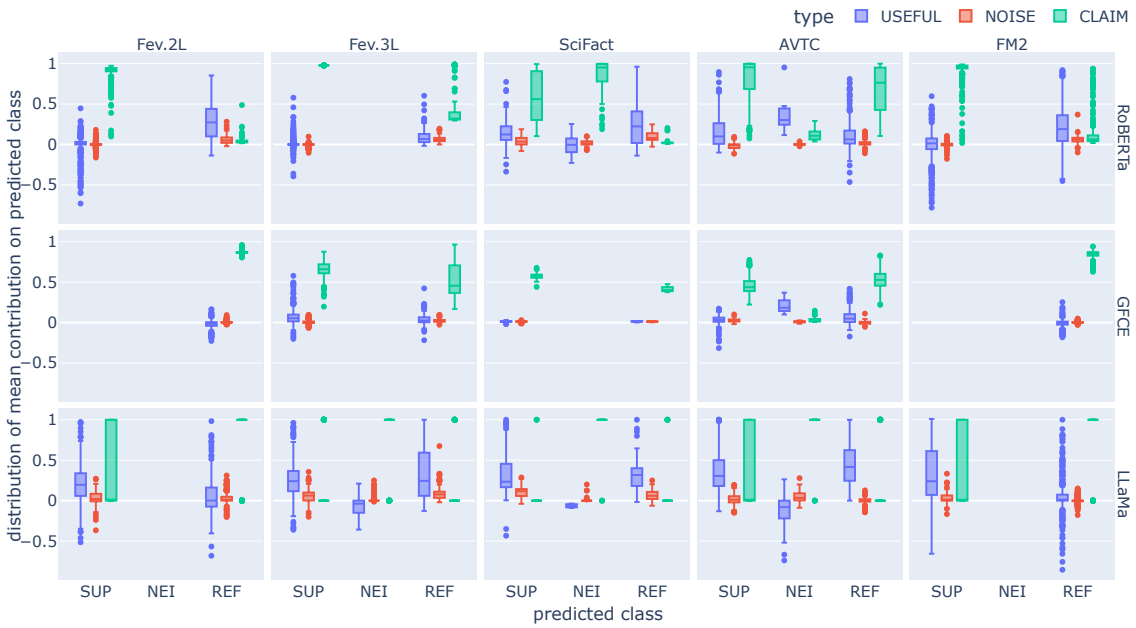


Fig. 6.4 Distribution of contribution scores obtained with SHAP of the useful evidence, noisy evidence, and claim on the predicted classes (Supports, Refutes, NEI). We spread horizontally the five datasets and vertically the three observed models. Every plot reports the predicted class over the x and the prediction contributions for useful evidence, noise and the claim itself. We run the explanation on the whole test set, including the incorrect prediction.

based on Roberta Large, selects the relevant cells for the claim. Finally, a predictor model, still based on Roberta Large, takes the claim as well as the combined modalities of evidence, text and table, as input, to output a verdict for a claim, *Supports*, *Refutes* or *Not Enough Information* in some contexts.

At label prediction step, for each example, the evidence is appended to the claim as input to the Roberta. Figure 6.6 presents how a claim from the dataset is fed to the predictor model along the retrieved evidence. This setting allows us to add or remove evidence and see the impact of such operation on the final result of label prediction.

6.5.4 LLaMa in low-resources constraints

LLaMa 3.1 70B requires to have a high-end GPU such as NVIDIA A100 to run it on. In a low-resource constraint setting, an user might want to be able to use an 8B variant of the model. We show in Table 6.6 the impact of this decision. The clear impact of the model switch, with more than 20 points lost in some cases, make this smaller model not an ideal choice for the user; both in term of evidence and claim labelling.

Fig. 6.5 Distribution of contribution scores obtained with LIME of the useful evidence, noisy evidence, and claim on the predicted classes (Supports, Refutes, NEI). We spread horizontally the five datasets and vertically the three observed models. Every plot reports the predicted class over the x and the prediction contributions for useful evidence, noise and the claim itself. We run the explanation on the whole test set, including the incorrect prediction.

6.5.5 Comparative Analysis of RoBERTa without noise

We investigate the role of presenting noise to the model at train time

This appendix presents a comparative analysis of two RoBERTa models: one trained with a mixture of noise and gold evidence, and another trained exclusively with gold evidence. Figure 6.8 illustrates the key differences between these models.

Training Data Composition Effects

Our analysis reveals distinct behaviors between the two models:

- Gold Evidence Only Model:
  - Exhibits enhanced ability to recognize useful evidence
  - Demonstrates an 'implicit refutes' behavior
  - Predicts *'Supports'* when useful supporting evidence is identified
  - More effectively utilizes evidence for predicting *'Supports'* claims
- Noise-Inclusive Model:

A village in China, in 2011 Kolhan had 693 people with around half (390) being workers. </s> Kolhan is a village in the Palghar district of [[Maharashtra|Maharashtra]], [[India|India]]. </s> Kolhan </s> Country is [[India|India]]. </s> Kolhan is [[India|India]].

Fig. 6.6 An example of input of the Feverous model. In green the original claim, in purple the corresponding evidence

You are an expert fact-checker. I will provide you a claim and a list of evidence. Based on them, you should answer in two steps.
In the first step, you should repeat every evidence that is useful to predict a label for the claim. Repeat only those evidence pieces, one evidence piece per line starting with their index, as they are presented to you. Start this step by the text: `Useful evidence:`
The second step should contain your predicted label. The label `[PREDICTED LABEL]` can be `$LABELS_TAG$`. Answer even if you are not sure. The format of the second step should be `Label:[PREDICTED LABEL]`

Fig. 6.7 Prompt for LLaMa. `$LABELS_TAG$` is a place holder for the possible labels of the given dataset.

- Displays an 'implicit supports' behavior
- Predicts *'Refutes'* when refuting evidence is detected

Concerning evidence utilization, the model trained exclusively on gold evidence appears to exploit the available evidence more effectively, particularly when predicting *'Supports'* claims. This suggests that the inclusion of noise in the training data may impact the model's ability to fully leverage the evidence provided.

These findings indicate that the composition of training data significantly influences the model's behavior and its approach to evidence evaluation. The gold-evidence-only model seems to adopt a more conservative stance, requiring strong supporting evidence to predict *'Supports'*, while the noise-inclusive model appears more inclined to predict *'Supports'* unless confronted with compelling refuting evidence.

This difference in behavior has important implications for the application of these models in fact-checking tasks and highlights the need for careful consideration of training data composition in model development.

Claim labelling				
Dataset	**Accuracy**	**F1 Sup**	**F1 NEI**	**F1 Ref**
Fev. 3L	0,57 (0,54)	0,70 (0,61)	0,17 (0,12)	0,44 (0,53)
Fev. 2L	0,69 (0,60)	0,70 (0,64)	-	0,68 (0,53)
SciFact	0,76 (0,50)	0,80 (0,69)	0,71 (0,18)	0,72 (0,41)
FM2	0,87 (0,65)	0,88 (0,52)	-	0,87 (0,73)
AVeriTeC	0,76 (0,63)	0,77 (0,49)	0,44 (0,27)	0,85 (0,75)
Evidence labelling				
Dataset		**Accuracy**	**F1 Ev.**	**F1 Noise**
Fev. 3L		0,88 (0,75)	0,46 (0,28)	0,93 (0,85)
Fev. 2L		0,88 (0,76)	0,47 (0,29)	0,93 (0,85)
SciFact		0,80 (0,53)	0,51 (0,30)	0,88 (0,64)
FM2		0,85 (0,59)	0,58 (0,30)	0,91 (0,71)
AVeriTeC		0,88 (0,60)	0,81 (0,54)	0,92 (0,64)

Table 6.6 Performance metrics of LLaMa 3.1 70B versus 8B (scores in parenthesis) for Claim labelling and Evidence labelling across datasets.

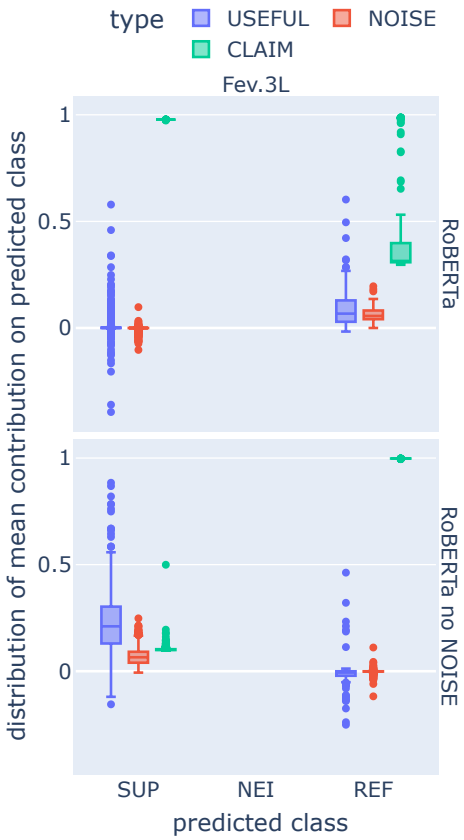


Fig. 6.8 Distribution of contribution scores obtained with SHAP of the useful evidence, noisy evidence, and claim on the predicted classes (Supports, Refutes, NEI). In depth analysis for Roberta trained with and without noisy evidence.

Chapter 7

Fair Classification with a Scalable Reduction Approach

The mitigation techniques proposed in the literature and partially reviewed in Section 2 suffer from limitations that make them difficult to apply to real-world scenarios. Mitigation techniques based on data pre-processing adjust the training data before it is input into the algorithm. They are generally model-agnostic, but they can result in suboptimal accuracy and fairness. In the hospital example, in section 1, this could result in a model that exhibits substantial allocative harms and whose accuracy is too low to be useful in the application. Post-processing techniques optimize fairness by adjusting the model's outputs after training, offering flexibility in model choice but requiring access to sensitive attributes (the attribute describing the race group in the example) at deployment, which can be legally restricted in certain domains. In-training techniques are typically too rigid, because they tend to incorporate only one specific type of fairness constraint in one specific training algorithm (and one specific model family). This restricts their practicality in real-world scenarios that require evaluating fairness using various metrics and testing different machine learning models.

EXPGRAD [1, 2] (EXPonentiated GRADient) has been developed to overcome the aforementioned state-of-the-art limitations. It is conceived as a reduction approach that works as a wrapper around any standard classification algorithm, and supports a wide range of fairness definitions. This approach provides the flexibility to enhance the fairness of existing AI systems without necessitating a re-architecture of deployed systems. However, its application to large datasets is limited by the substantial training time required, i.e., it is up to a factor of 100 slower than unmitigated techniques. The algorithmic approach proposed in the paper, which we call EXPGRAD^{++}, can speed up the reduction approach by an order of magnitude using two algorithmic innovations.

ExpGrad *in a nutshell.* ExpGrad deals with standard classification algorithms that take as input a binary classification dataset and return a classifier h that minimizes the classification error among all classifiers from some family $\mathscr{H}$. We refer to these standard classification algorithms as *oracles*. For example, the logistic regression algorithm is an oracle for the family $\mathscr{H}$ consisting of linear classifiers. ExpGrad addresses the challenge as a constrained optimization problem of finding the most accurate randomized classifier (over the family $\mathscr{H}$) subject to fairness constraints. Randomized classifiers over $\mathscr{H}$ correspond to probability distributions over $\mathscr{H}$, and so they are represented as vectors in $|\mathscr{H}|$ dimensions, which are intractably large. ExpGrad is able to solve such a large dimensional problem by performing a coordinate descent. In each iteration, it selects one coordinate out of $|\mathscr{H}|$ (a specific classifier $h_t \in \mathscr{H}$), whose inclusion in the randomized solution helps decrease the current violation of fairness constraints, while still preserving good accuracy. So, although the dimensionality of the solution space is very large, after t iterations, only t coordinates are non-zero. The key question is how to pick the next coordinate along which to update. This is done by creating a weighted version of the training data and calling the oracle. The classifier h_t returned by the oracle in the t-th iteration of ExpGrad corresponds to the coordinate which is now being included in the randomized solution. The weights on the training data in the oracle call are chosen in such a way as to exaggerate those examples that need to be classified correctly to decrease the violation of the fairness constraints.

Limitations of ExpGrad. In practice, ExpGrad takes around 100 iterations to converge to the randomized classifier that optimizes accuracy under the specified fairness constraint. The most expensive operation in each iteration is the oracle call, which means solving a standard classification problem on a weighted data set. In principle, scaling ExpGrad performance requires solving two challenges: (1) reducing the number of oracle calls, and (2) making each call cheaper (e.g., LightGBM oracle calls in ExpGrad take on average 84.12% of the total training time). These are non-trivial challenges. For example, solving (1) with early stopping could have detrimental consequences on both accuracy and fairness, while solving (2) through uniformly at random sampling does not work (see Section 7.3.4). Therefore the novelty of this work lies precisely in solving these challenges.

Our contribution. Our first innovation is to insert an additional low-dimensional optimization step, which speeds up the convergence of the algorithm, so the solution is reached in 10 iterations (and thus 10 oracle calls) rather than 100. Our second innovation is to decrease the cost of each oracle call by passing it a smaller dataset obtained by a proper subsampling. To understand the first innovation, recall that ExpGrad seeks to solve a constrained optimization problem in $|\mathscr{H}|$ dimensions, and in each iteration, it performs a coordinate descent,

which adds a new non-zero coordinate to the current iterate. We insert an additional step that seeks to solve the same constrained optimization problem, but only over the classifiers $h_1, \ldots, h_t$ returned so far, so in the t-th iteration, this is only a t-dimensional problem. This restricted problem can be solved fast using linear programming. We then use one oracle call to check whether the restricted solution actually solves the full $|\mathscr{H}|$-dimensional problem, and if it does, we terminate. Conceptually, this allows us to shortcut many iterations of coordinate descent, and thus reach convergence faster. This approach is called *column generation* [47, 58], because the classifiers generated by the oracle calls are included as columns in the matrix that represents the restricted problem as a linear program. In practice, column generation decreases the number of EXPGRAD iterations from around 100 to around 10, as we show in our experiments (see Section 7.3).

While the column generation decreases the number of oracle calls, the goal of our second innovation is to decrease the cost of each oracle call. One simple approach, which we call *static sampling*, would be to subsample the training data uniformly at random and solve the constrained optimization problem for the smaller dataset; this improves the running time but leads to some loss in accuracy. We improve upon this naïve strategy by noting that datasets passed to the oracle are weighted, with a different weighting in each iteration. By sampling the data adaptively, according to the weights, we sacrifice less accuracy than we would if we used static sampling. For instance, if the dataset weights generated in a given iteration of EXPGRAD put 90% of probability mass on 10% of examples, then picking the 10% examples with the largest weights results in a much smaller loss in accuracy than picking an arbitrary 10% of examples. In our experiments, we show that our adaptive sampling approach outperforms static sampling, and we do not see major losses in accuracy, even when the sample size is only 25% or even less of the original dataset size (see Section 7.3).

Our experiments evaluate the efficacy of EXPGRAD^{++} and compare it both with EXPGRAD and other baselines. We show that our two innovations (column generation and adaptive sampling) speed up EXPGRAD by an order of magnitude. The resulting approach still enjoys the flexibility of EXPGRAD, while substantially improving its scalability. In fact, the column generation improvement, which leads to a speed-up without any sacrifice to accuracy, has been already incorporated into the *fairlearn* library.

In summary, the main contributions of this paper are: (1) A novel method that interleaves exponentiated-gradient updates with column-generation updates, significantly reducing the number of calls to the base algorithm; (2) An adaptive sampling technique, which decreases the sizes of the datasets used for training the models, thereby enhancing the efficiency of the training process; (3) A detailed experimental evaluation, including on large datasets,

demonstrating the efficiency and effectiveness of the proposal making it able to work in real-world scenarios.

In Section 7.1 we formalize the problem setting and describe the reduction approach in more detail. In Section 7.2 we introduce EXPGRAD^{++}, which is then extensively evaluated in Section 7.3. Finally, in Section 7.4, we conclude and discuss future directions.

7.1 Preliminaries

7.1.1 Problem definition

We consider binary classification tasks, where the input is a dataset consisting of labeled examples $(X_1, A_1, Y_1), \ldots, (X_n, A_n, Y_n)$, where $X_i \in \mathbb{R}^d$ is a feature vector describing the i-th example, $A_i \in \mathscr{A}$ is a categorical sensitive attribute, and $Y_i \in \{0,1\}$ is a binary label.

For example, consider training an AI system that refers patients to a post-discharge care program based on the risk of a 30-day readmission. In this case, X_i contains clinical information about a patient; A_i is a categorical variable encoding the patient's race and ethnicity; and Y_i indicates whether the patient was readmitted within 30 days of discharge.

The goal of a classification algorithm is to produce a classifier $h : \mathbb{R}^d \to \{0,1\}$ that accurately predicts label Y on a new example represented by a feature vector $X \in \mathbb{R}^d$. We assume that h is chosen from some family $\mathscr{H}$ (like linear classifiers, decision trees, or neural nets). For example, logistic regression considers linear classifiers of the form $h_\beta(X) = 1\{\beta^T X \geq 0\}$ where $\beta \in \mathbb{R}^d$ and $1\{\cdot\}$ is an indicator function.

The sensitive attribute A is only required during training, but not during inference. However, we make no assumption on the relationship between X and A, and so it is possible for X to contain some information about A. In this way, our task definition encompasses both the settings when the sensitive attribute is available at inference time (in that case it is included as part of X) as well as the settings when the sensitive attribute is not available (in that case it is not included as part of X).

Standard classification algorithms seek $h \in \mathscr{H}$ that minimizes training classification error $err(h)$:[1]

$$\min_{h \in \mathscr{H}} err(h), \quad \text{where} \quad err(h) = \frac{1}{n} \sum_{i=1}^{n} 1\{h(X_i) \neq Y_i\}. \tag{7.1}$$

[1]Standard algorithms also include various mechanisms to control overfitting, such as regularization or early stopping. Similarly to Agarwal et al. [1], we model regularization (and other similar mechanisms) by assuming that the family $\mathscr{H}$ has been appropriately restricted. For example, for regularized logistic regression, $\mathscr{H} = \{h_\beta : \beta \in \mathbb{R}^d, \|\beta\| \leq C\}$, where the value of C controls overfitting.

Algorithmic approaches to unfairness mitigation instead optimize training classification error under a fairness constraint, typically specified using the sensitive attribute A.

There are many notions of (un)fairness, appropriate in different applications (e.g., [68] lists 34 measures collected from a review of the literature). In this paper, for simplicity, we focus on two standard quantitative definitions of fairness, but our technique encompasses many other notions (see [1] for further details). Specifically, we consider *demographic parity* and *equalized odds* [62, 16]:

Definition 1 (Demographic parity: DP)**.** *We say that a classifier h satisfies* demographic parity *with respect to a distribution over triples (X,A,Y) if its decision is statistically independent of A, that is, if $\mathbb{E}[h(X) \mid A = a] = \mathbb{E}[h(X)]$ for all $a \in \mathscr{A}$.*

Definition 2 (Equalized odds: EO)**.** *We say that a classifier h satisfies* equalized odds *with respect to a distribution over triples (X,A,Y) if its decision is statistically independent of A, conditional on Y, that is, if $\mathbb{E}[h(X) \mid A = a, Y = y] = \mathbb{E}[h(X) \mid Y = y]$ for all $a \in \mathscr{A}$ and $y \in \{0,1\}$.*

The degree to which a classifier h satisfies demographic parity on a data set $(X_1,A_1,Y_1),\ldots,(X_n,A_n,Y_n)$ can be quantified using the quantity

$$\begin{aligned}\Delta_{\mathrm{DP}}(h) &= \max_{a\in\mathscr{A}} \left|\hat{\mathbb{E}}[h(X) \mid A = a] - \hat{\mathbb{E}}[h(X)]\right| \\ &= \max_{a\in\mathscr{A}} \left|\frac{1}{n_a}\sum_{i:A_i=a} h(X_i) - \frac{1}{n}\sum_{i=1}^{n} h(X_i)\right|\end{aligned} \tag{7.2}$$

where n_a is the number of examples with $A_i = a$. Compared with Definition 1, in Eq. (7.2) we replace true expectations $\mathbb{E}[\cdot]$ with empirical averages $\hat{\mathbb{E}}[\cdot]$, evaluated on the training data. The value of Δ_{DP} is equal to the largest deviation of $\hat{\mathbb{E}}[h(X) \mid A = a]$ from $\hat{\mathbb{E}}[h(X)]$, across all a. Definition 1 is satisfied (on the empirical distribution) exactly when the deviations across all a are equal to zero. Otherwise, Δ_{DP} quantifies an (additive) violation of fairness constraints. We refer to Δ_{DP} as the *DP difference*.

For equalized odds, we can quantify the violation of fairness constraints using an analogous quantity, called the *EO difference*:

$$\begin{aligned}\Delta_{\mathrm{EO}}(h) &= \max_{a\in\mathscr{A},y\in\{0,1\}} \left|\hat{\mathbb{E}}[h(X) \mid A = a, Y = y] - \hat{\mathbb{E}}[h(X) \mid Y = y]\right| \\ &= \max_{a\in\mathscr{A},y\in\{0,1\}} \left|\frac{1}{n_{a,y}}\sum_{i:A_i=a,Y_i=y} h(X_i) - \frac{1}{n_y}\sum_{i:Y_i=y} h(X_i)\right|\end{aligned}$$

where n_y and $n_{a,y}$ refer to the number of examples with $Y_i = y$ and $A_i = a$, $Y_i = y$, respectively.

Continuing with the post-discharge care example, demographic parity states that the patients from all race/ethnicity groups should be referred to the post-discharge care program at equal rates. The fairness constraint $\Delta_{\mathrm{DP}}(h) \leq \varepsilon$ states that the referral rate of every race/ethnicity group is only allowed to differ from the overall referral rate by at most ε.

The equalized odds condition states that false positive rates and false negative rates for all the race/ethnicity groups should be the same. The fairness constraints $\Delta_{\mathrm{EO}}(h) \leq \varepsilon$ states that false positive rates and false negative rates of every race/ethnicity group are allowed to differ from the overall false positive rates and overall false negative rates, respectively, by at most ε.

7.1.2 Reduction approach

Reduction approach to fair classification [1] seeks to solve the problems of the form:

$$\min_{h \in \mathscr{H}} err(h) \quad \text{such that} \quad \Delta(h) \leq \varepsilon, \tag{7.3}$$

where Δ formalizes fairness constraints and $\varepsilon \geq 0$ is the degree to which we allow the fairness constraint to be violated.

Reduction approach works with a broad family of fairness constraints including Δ_{DP}, Δ_{EO} and many others (see [1] for details). It also works with any family of ML models; it only requires the ability to solve weighted (but unconstrained) classification problems, meaning problems that minimize weighted classification error

$$\min_{h \in \mathscr{H}} \frac{1}{n} \sum_{i=1}^{n} w_i 1\{h(X_i) \neq Y_i\} \tag{7.4}$$

for any set of weights $w_i \geq 0$. Eq. (7.4) can be solved by standard fairness-unaware classification algorithms, like those that fit logistic regression models, decision trees, or neural nets. The algorithms that solve Eq. (7.4) are viewed as *oracles* from the perspective of the reduction, and in practice they are implemented by library calls.

In order to leverage the strength of an oracle, we cast the constrained optimization problem in Eq. (7.3) as a linear optimization. For a classifier $h \in \mathscr{H}$, let $\mathbf{v}_h \in \mathbb{R}^n$ denote the vector with entries $v_{h,i} = h(X_i)$ for $i = 1, \ldots, n$. Then Eq. (7.3) can be rewritten as

$$\min_{h \in \mathscr{H}} \mathbf{c}^T \mathbf{v}_h + c_0 \quad \text{such that} \quad \mathbf{A}\mathbf{v}_h \leq \mathbf{b}, \tag{7.5}$$

where the vector $\mathbf{c} \in \mathbb{R}^n$ and the scalar $c_0 \in \mathbb{R}$ are determined by the choice of an error metric, in our case $err(h)$; further details in Agarwal et al. [1]. The matrix $\mathbf{A} \in \mathbb{R}^{k \times n}$ and vector

$\mathbf{b} \in \mathbb{R}^k$ are based on the choice of the fairness metric Δ and bound ε from Eq. (7.3). The dimension k is determined by the cardinality of $\mathscr{A}$ and the choice of the fairness metric Δ.

For example, using Eq. (7.2), the constraint $\Delta_{\mathrm{DP}}(h) \leq \varepsilon$ can be written as $2 \mid \mathscr{A} \mid$ constraints of the form

$$\begin{aligned} \hat{\mathbb{E}}[h(X) \mid A = a] - \hat{\mathbb{E}}[h(X)] &\leq \varepsilon \\ -\hat{\mathbb{E}}[h(X) \mid A = a] + \hat{\mathbb{E}}[h(X)] &\leq \varepsilon, \end{aligned}$$

for all $a \in \mathscr{A}$, which can be expressed using a suitable matrix $\mathbf{A}$, with the vector $\mathbf{b}$ having all entries equal to ε. The constraint $\Delta_{\mathrm{EO}}(h) \leq \varepsilon$ can be similarly expressed using $k = 4 \mid \mathscr{A} \mid$ constraints (two constraints for each combination of $a \in \mathscr{A}$, $y \in \{0,1\}$). Full derivation of $\mathbf{A}$ for DP, EO (as well as for more general fairness constraints) is provided in [1]. We view k as a problem-specific constant. In much prior work, the sensitive attribute is binary ($\mid \mathscr{A} \mid = 2$), yielding $k = 4$ for DP and $k = 8$ for EO.

Instead of working with classifiers $h \in \mathscr{H}$, the reduction approach considers a larger set consisting of randomized classifiers that randomize over a finite subset of $\mathscr{H}$. We write $\mathscr{P}(\mathscr{H})$ for the set of such randomized classifiers and identify them with the corresponding probability distributions over $\mathscr{H}$. A randomized classifier $p \in \mathscr{P}(\mathscr{H})$ makes a prediction on an example X by first sampling $h \sim p$ and then predicting $h(X)$. For example, if p puts 0.5 probability mass on h_1 and 0.5 probability mass on h_2, then the randomized classifier specified by p predicts 1 on points X where $h_1(X) = h_2(X) = 1$, predicts 0 on points X where $h_1(X) = h_2(X) = 0$, and flips a coin on points X where $h_1(X) \neq h_2(X)$.

The linear optimization from Eq. (7.5) can be generalized to randomized classifiers. For any $p \in \mathscr{P}(\mathscr{H})$, we set $\mathbf{v}_p = \sum_{h \in \mathscr{H}} p(h)\mathbf{v}_h$, and the resulting optimization problem is

$$\min_{p \in \mathscr{P}(\mathscr{H})} \mathbf{c}^T \mathbf{v}_p \quad \text{such that} \quad \mathbf{A}\mathbf{v}_p - \mathbf{b} \leq 0. \tag{7.6}$$

This constrained optimization problem can be algorithmically solved by finding a solution to the min-max problem

$$\min_{p \in \mathscr{P}(\mathscr{H})} \max_{\lambda \in \mathbb{R}^k_+, \|\lambda\|_1 \leq B} L(\mathbf{v}_p, \lambda) \tag{7.7}$$

where $L(\mathbf{v}, \lambda)$ is the Lagrangian

$$L(\mathbf{v}, \lambda) = \mathbf{c}^T \mathbf{v} + \lambda^T (\mathbf{A}\mathbf{v} - \mathbf{b}), \tag{7.8}$$

and $\lambda \in \mathbb{R}^k_+$ is the vector of non-negative Lagrange multipliers.

Algorithm 2: EXPGRAD

Input : Lagrangian specified by $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{k \times n}$, $\mathbf{b} \in \mathbb{R}^k$
bound B, learning rate η, convergence threshold ν
maximum iterations T

Set $\theta_1 \leftarrow 0 \in \mathbb{R}^k$
for $t = 1, 2, \ldots, T$ **do**
  Set $\lambda_{t,j} = B \frac{\exp\{\theta_{t,j}\}}{1+\sum_{j'=1}^{k} \exp\{\theta_{t,j'}\}}$ for $j = 1, \ldots, k$
  $h_t \leftarrow \text{BEST}_h(\lambda_t)$, and let $\mathbf{v}_t = \mathbf{v}_{h_t}$
  $\mathbf{v}_{\text{EG}} \leftarrow \frac{1}{t}\sum_{t'=1}^{t} \mathbf{v}_{t'}$, $\lambda_{\text{EG}} \leftarrow \frac{1}{t}\sum_{t'=1}^{t} \lambda_{t'}$
  $\nu_{\text{EG}} \leftarrow \texttt{EvaluateDualityGap}(\mathbf{v}_{\text{EG}}, \lambda_{\text{EG}})$
  if $\nu_{\text{EG}} \leq \nu$ ***or*** $t = T$ **then**
    return *p that randomizes uniformly over* $h_1, \ldots, h_t$
  Set $\theta_{t+1} \leftarrow \theta_t + \eta(\mathbf{A}\mathbf{v}_t - \mathbf{b})$
Function `EvaluateDualityGap(`$\mathbf{v}, \lambda$`)`**:**
  $\overline{L} \leftarrow L(\mathbf{v}, \lambda^*)$ where $\lambda^* = \arg\max_{\lambda' \in \mathbb{R}^k_+, \|\lambda'\|_1 \leq B} L(\mathbf{v}, \lambda')$;
  $\underline{L} \leftarrow L(\mathbf{v}_{h^*}, \lambda)$ where $h^* = \text{BEST}_h(\lambda)$;
  return $\max\{L(\mathbf{v}, \lambda) - \underline{L},\ \overline{L} - L(\mathbf{v}, \lambda)\}$;
Function BEST_h`(`λ`)`**:**
  `//` `returns` $\arg\min_{h \in \mathscr{H}} L(\mathbf{v}_h, \lambda)$
  Let $\mathbf{w} = \mathbf{c} + \mathbf{A}^T\lambda$;
  Find $h^* = \arg\min_{h \in \mathscr{H}} \mathbf{w}^T\mathbf{v}_h$; `// by calling a standard classification algorithm for` $\mathscr{H}$ `on the data set reweighted according to` $\mathbf{w}$
  return h^*;

Conceptually, the Lagrangian *scalarizes* the original constrained optimization problem in Eq. (7.6) by summing up the objective and individual constraint violations, multiplied by the Lagrange multipliers. The Lagrange multipliers λ_j, for $j = 1, \ldots, k$, specify the importance of not violating each of the constraints. The possibility of using sufficiently large λ_j in the inner maximization forces the outer minimization to choose the point p that (approximately) satisfies the constraints. It can be shown formally that with an appropriate choice of B, the solution of Eq. (7.7) approximately solves Eq. (7.6) (see [1]).

The reduction approach solves the min-max problem from Eq. (7.7) by an iterative algorithmic scheme developed by Freund and Schapire [53] for finding an equilibrium in a zero-sum game. The min-max problem in Eq. (7.7) can be viewed as a game between a λ-player seeking to maximize the Lagrangian and $\mathbf{v}$-player seeking to minimize the Lagrangian. Freund and Schapire [53] propose an iterative protocol where in each round

t, the λ-player plays the action λ_t according to a suitable online learning algorithm (in our case the exponentiated gradient algorithm of Kivinen and Warmuth [74]) and $\mathbf{v}$-player plays the best response $\mathbf{v}_t$ to the other player's action λ_t. Freund and Schapire [53] show that the averages of the played actions λ_t and $\mathbf{v}_t$ converge to an equilibrium of the game, which coincides with the solution of the min-max problem.

This scheme is implemented in Algorithm 2. The vector $\theta_t \in \mathbb{R}^k$ is used to obtain λ_t (the action of the λ-player) via a soft-max transformation that guarantees that the components of λ_t are non-negative and sum to at most B (step 3). The best response of the $\mathbf{v}$-player is obtained in step 4 by calling the function $\textsc{Best}_h(\lambda_t)$, which is implemented by calling the oracle for the family $\mathscr{H}$. In steps 5 and 6, we consider the current average play of the λ- and $\mathbf{v}$-player and check how close these averages are to the equilibrium of the game by evaluating how much each player can unilaterally improve their objective (see function EvaluateDualityGap). If the possible improvement is below a convergence threshold ν, or if we have reached the maximum number of iterations, we return the current average play. Otherwise, we update θ_t, following the exponentiated-gradient update rule. Conceptually, we form a vector of constraint violations $\mathbf{A}\mathbf{v}_t - \mathbf{b}$ and increase the values of the components of θ_t (and thus also of λ_t) according to how much violation occurs. This means that in the next iteration of the protocol, the constraints that were more violated receive more importance in the Lagrangian.

Agarwal et al. [1] show that for typical families $\mathscr{H}$ (like linear classifiers, neural nets, and boosted trees), given a dataset of size n, we should set $B \propto \sqrt{n}$, $\eta \propto 1/n$, $\nu \propto 1/\sqrt{n}$, and $T \propto n^2$ to guarantee that the solution returned by the algorithm satisfies the fairness constraints and matches the accuracy of the solution of Eq. (7.7) (up to an error of at most $O(1/\sqrt{n})$, which is on the same scale as the difference between the training error and error with respect to the true distribution). This means that for a dataset of size 1 000, the theory requires around 1 million iterations! The most costly operation in each iteration is the call to $\textsc{Best}_h$, which involves calling an oracle, on a weighted dataset of size n. So theory would yield the running time that is 1 million times slower than the running time of a fairness-unaware approach. In practice, on the datasets of size up to 50 000, we find that around 100 iterations suffice to reach the termination condition (see our experiments in Section 7.3). However, a 100-fold slowdown is still prohibitive and presents the main obstacle for applying a reduction approach with larger datasets.

7.2 Scalable Reduction

We introduce two innovations to speed up Algorithm 2. First, we interleave exponentiated gradient with column generation (see, e.g., [58], Section 7.3) to decrease the number of

iterations to around 5–10 (instead of 100). Second, we use sampling to generate smaller training data sets when calling the oracle in BEST_h. The resulting approach is presented in Algorithm 3, with new and revised steps marked with an asterisk.

7.2.1 Column generation

Assume that the family $\mathscr{H}$ is of finite cardinality, $|\mathscr{H}| = N$; this is without loss of generality, because the optimization in Eqs. (7.6) and (7.7) only considers predictions on a fixed dataset of size n, and hence we can assume $|\mathscr{H}| \leq 2^n$.

Let $\mathbf{V} \in \mathbb{R}^{n \times N}$ be the matrix with columns corresponding to vectors $\mathbf{v}_h$ across $h \in \mathscr{H}$. Then a probability distribution $p \in \mathscr{P}(\mathscr{H})$ can be viewed as a vector in $\mathbb{R}^N_+$ with the components $p(h)$ summing to one, $\sum_{h \in \mathscr{H}} p(h) = 1$, and $\mathbf{v}_p$ is obtained by matrix-vector multiplication as $\mathbf{v}_p = \mathbf{V}p$. Thus, Eq. (7.7) can be written as a linear programming (LP) problem:

$$\max_{\lambda \in \mathbb{R}^k_+, \|\lambda\|_1 \leq B} \min_{p \in \mathbb{R}^N_+, \sum_{h \in \mathscr{H}} p(h) = 1} \left[\mathbf{c}^T \mathbf{V} p + \lambda^T (\mathbf{A}\mathbf{V}p - \mathbf{b}) \right]. \tag{7.9}$$

It is intractable to solve this problem with standard LP solvers (since N is exponential in n). To circumvent the dimensionality of N, the column generation (CG) approach considers a small subset of base classifiers $\tilde{\mathscr{H}} \subseteq \mathscr{H}$, with $m = |\tilde{\mathscr{H}}|$, and replaces the large matrix $\mathbf{V} \in \mathbb{R}^{n \times N}$ by a smaller matrix $\tilde{\mathbf{V}} \in R^{n \times m}$ that only contains columns corresponding to the classifiers $h \in \tilde{\mathscr{H}}$. Instead of directly solving Eq. (7.9), CG solves a smaller restricted problem

$$\max_{\lambda \in \mathbb{R}^k_+, \|\lambda\|_1 \leq B} \min_{p \in \mathbb{R}^m_+, \sum_{h \in \tilde{\mathscr{H}}} p(h) = 1} \left[\mathbf{c}^T \tilde{\mathbf{V}} p + \lambda^T (\mathbf{A}\tilde{\mathbf{V}}p - \mathbf{b}) \right]. \tag{7.10}$$

The CG approach starts with $m = 1$ and $\tilde{\mathscr{H}}$ containing an arbitrary base classifier (for example, the classifier obtained by optimizing accuracy without any fairness constraints), and then repeatedly solves the restricted problem in Eq. (7.10). The solution to the restricted problem is checked for optimality using the function EVALUATEDUALITYGAP from Algorithm 2. If the duality gap is non-zero, it means that h^* obtained in the call to EVALUATEDUALITYGAP can be used to reduce the objective in Eq. (7.10), and so h^* is added to $\tilde{\mathscr{H}}$, its corresponding vector $\mathbf{v}_{h^*}$ is added to $\tilde{\mathbf{V}}$, and the LP in Eq. (7.10) is solved again. This is repeated until convergence.

In practice CG can converge in a very small number of iterations, but we are not aware of any non-trivial theoretical upper bounds. Therefore, we interleave column generation (CG) with exponentiated gradient (EG), which allows us to benefit from strong practical performance of CG while retaining the worst-case guarantees of EG. We run the two ap-

proaches independently, except that the classifiers obtained by oracle calls in EG are included as columns in CG, which avoids redundant oracle calls.

More precisely, to combine the two approaches, we introduce a data structure that contains all of the classifiers returned by $\textsc{Best}_h$ so far, which we refer to as *cache*. This *cache* plays a role of $\tilde{\mathscr{H}}$ in CG. The modified algorithm (Algorithm 3) initializes *cache* with the classifier returned by a standard (fairness-unaware) classification algorithm for $\mathscr{H}$ (step 2). In each iteration t, the algorithm first performs the EG update (steps 4–9), and if the convergence condition in step 8 is not satisfied, it runs a single iteration of CG with $\tilde{\mathscr{H}} = \textit{cache}$ (steps 10–14). If the CG convergence condition is not satisfied (in step 13) then it finishes the EG update (in step 15) and proceeds to next iteration. In other words, in each iteration, we perform both the EG update (which involves an oracle call) and also a CG step. Convergence can be reached by either EG or CG. Solving the restricted LP in the CG step is significantly faster than an oracle call, so repeating it at each iteration does not significantly impact running time. However, CG enjoys an advantage over EG in how it assigns probabilities to each classifier in the randomized classifier. In EG, the probabilities are distributed evenly over the set of classifiers selected so far (in step 6). On the other hand, CG finds probabilities by solving Eq. (7.10), so these probabilities are the ones that give the best solution to the problem in Eq. (7.3) when considering the set of classifiers obtained from all the iterations of EG performed so far. Thus, the CG solution in each step is always as good or better than the EG solution in that step. Since the EG iterates are unaffected, we retain the original convergence guarantee. However, by introducing CG steps, it is possible to terminate much earlier if the duality gap of the CG iterates falls below ν. In practice, we observe that this termination condition is reached in around 10 iterations instead of 100 iterations that were required by EG to reach a similar quality of the solution (see Section 7.3).

While CG decreases the number of iterations and therefore the number of oracle calls, the goal of our second innovation is to decrease the cost of each oracle call. We do this by subsampling the data before passing it to the oracle.

Subsampling is a general and widely used strategy that makes the training process faster and more manageable in terms of storage and memory resources, but it can lead to a loss of accuracy in the obtained solution if the sample does not represent the original data sufficiently well. A naïve strategy, which we call *static sampling*, is to subsample the training data uniformly at random and solve the constrained optimization problem for the smaller dataset. We expect this strategy to do well in approximating the overall training error, but it might severely impact the accuracy of fairness metrics, which are calculated on various subgroups of data, and whose sizes might become too small as a result of subsampling.

Algorithm 3: ExpGrad+

Input : Lagrangian specified by $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{k \times n}$, $\mathbf{b} \in \mathbb{R}^k$
bound B, learning rate η, convergence threshold ν
maximum iterations T, sampling ratio ρ

1 Set $\theta_1 \leftarrow 0 \in \mathbb{R}^k$
2 $cache \leftarrow \{h_{\text{init}}\}$ where $h_{\text{init}} = \arg\min_{h \in \mathscr{H}} err(h)$
3 **for** $t = 1, 2, \ldots, T$ **do**
4 Set $\lambda_{t,j} = B \frac{\exp\{\theta_{t,j}\}}{1+\sum_{j'=1}^{k} \exp\{\theta_{t,j'}\}}$ for $j = 1, \ldots, k$
5 $h_t \leftarrow \text{BEST}_h(\lambda_t)$, and let $\mathbf{v}_t = \mathbf{v}_{h_t}$
6 $\mathbf{v}_{\text{EG}} \leftarrow \frac{1}{t}\sum_{t'=1}^{t} \mathbf{v}_{t'}$, $\lambda_{\text{EG}} \leftarrow \frac{1}{t}\sum_{t'=1}^{t} \lambda_{t'}$
7 $\nu_{\text{EG}} \leftarrow$ `EvaluateDualityGap`$(\mathbf{v}_{\text{EG}}, \lambda_{\text{EG}})$; `// see Alg. 2`
8 **if** $\nu_{\text{EG}} \leq \nu$ **then**
9 **return** *p that randomizes uniformly over* $h_1, \ldots, h_t$
10 Let $\tilde{\mathbf{V}}$ be the matrix with columns $\mathbf{v}_h$ across $h \in cache$
11 $(\lambda_{\text{CG}}, p_{\text{CG}}) \leftarrow$ solve Eq. (7.10) with $\tilde{\mathscr{H}} = cache$
12 $\nu_{\text{CG}} \leftarrow$ `EvaluateDualityGap`$(\tilde{\mathbf{V}} p_{\text{CG}}, \lambda_{\text{CG}})$; `// see Alg. 2`
13 **if** $\nu_{\text{CG}} \leq \nu$ ***or*** $t = T$ **then**
14 **return** *p that randomizes over cache according to* p_{CG}
15 Set $\theta_{t+1} \leftarrow \theta_t + \eta(\mathbf{A}\mathbf{v}_t - \mathbf{b})$
16 **Function** $\text{BEST}_h(\lambda)$**:**
`// approximates` $\arg\min_{h \in \mathscr{H}} L(\mathbf{v}_h, \lambda)$
17 Let $\mathbf{w} = \mathbf{c} + \mathbf{A}^T \lambda$
18 Set $q_i = \frac{|w_i|}{\sum_{i=1}^{n} |w_i|}$ for $i = 1, \ldots, n$;
19 $D \leftarrow \{\}$
20 **for** 1 **to** $\lceil \rho n \rceil$ **do**
21 Sample $i \sim \mathbf{q}$
22 Add to D an example (X, Y) with $X = X_i$, $Y = 1\{w_i < 0\}$
23 Find $h^* = \arg\min_{h \in \mathscr{H}} \sum_{(X,Y) \in D} 1\{h(X) \neq Y\}$; `// by calling a standard classification algorithm for` $\mathscr{H}$
24 Add h^* to *cache* and return h^*

We improve upon this naïve strategy by noting that datasets passed to the oracle are weighted, with a different weighting in each iteration determined by the current vector of Lagrange multipliers λ_t (see step 4 and the implementation of the function BEST_h in Algorithm 2). The components of the weight vector $\mathbf{w} \in \mathbb{R}^n$ specify how important each data point is. Operationally, this tends to upweight the groups for which the fairness constraint is most violated. This suggests a natural adaptive sampling strategy, which subsamples the original data according to $\mathbf{w}$. We implement such an adaptive sampling strategy in

the function BEST_h in Algorithm 3. The size of the subsampled dataset is controlled by a sampling ratio $\rho \in (0,1)$, with the subsampled dataset of size $\lceil \rho n \rceil$.

By sampling the data adaptively, according to the weight vector $\mathbf{w}$, we sacrifice less accuracy than we would if we used static sampling. For instance, if the weight vector $\mathbf{w}$ in a given iteration of EXPGRAD^{++} puts 90% of probability mass on 10% of examples, then picking the 10% examples with the largest weights results in a much smaller loss in accuracy than picking an arbitrary 10% of examples. In our experiments, we show that the adaptive sampling approach outperforms static sampling, and we do not see major losses in accuracy even when the sample size is only 25% or even less of the original dataset size (see Section 7.3).

7.3 Experimental evaluation

In our experiments, we will be concerned with several aspects of the evaluated methods. The first is *effectiveness*: Are the methods producing accurate solutions that respect fairness constraints? The second is *efficiency*: What is the running time of the methods? Finally, the third is *flexibility*: Do the methods support a wide range of fairness metrics, accuracy-fairness trade-offs, classifier families, and sensitive attribute types? We will compare EXPGRAD^{++} with baselines along these three axes, and then also evaluate the impact of our two innovations (constraint generation and subsampling) on the performance of EXPGRAD^{++}.

More specifically, we conduct four types of experiments that analyze EXPGRAD^{++} from four perspectives: (1) **End-to-end usage.** The goal of the experiments in Section 7.3.2 is to show that EXPGRAD^{++} is an effective, efficient, and easy-to-tune approach for the development of end-to-end ML pipelines. We compare EXPGRAD^{++} with other baselines in terms of training time (quantifying efficiency) as well as accuracy and fairness (quantifying effectiveness). (2) **Contribution of the column generation.** In Section 7.3.3 we conduct an ablation study to evaluate the impact of the *column generation* on the effectiveness and efficiency of the reduction approach. (3) **Robustness to sampling.** In Section 7.3.4 we evaluate how the sampling ratio ρ impacts the effectiveness and efficiency of EXPGRAD^{++}. We also demonstrate that adaptive sampling leads to a better use of data (meaning more effective solutions) than static sampling. (4) **Support of multi-valued sensitive attributes.** EXPGRAD^{++} supports fairness metrics defined over multi-valued sensitive attributes, but many existing approaches only support binary sensitive attributes, so when the actual sensitive attribute is multi-valued the user needs to artificially binarize its values by partitioning them into two groups. The experiment in Section 7.3.5 shows that this procedure does not effectively mitigate unfairness with respect to the original attribute values.

7.3.1 Experimental Settings

System and implementation. We conducted the experiments on a machine equipped with Intel(R) Xeon(R) Platinum 8370C CPU @ 2.80GHz and 62.8 GB RAM, running on Ubuntu 20.04.5 LTS operating system. The experiments were coded in Python 3.9.12 using the *fairlearn* library version v0.9.0.dev0. All of the experiment code is publicly available in the project github.

Datasets. Our evaluation includes 5 real-world datasets, summarized in Table 7.1. Adult [17], COMPAS [78], and German [65] are small datasets usually adopted as a benchmark in many fairness studies [68, 89, 105]. ACSEmployment and ACSPublicCoverage [40] are large datasets that allow us to perform an evaluation in larger scenarios and to experiment with multi-valued sensitive attributes.

Adult describes demographic and occupational attributes of several thousand individuals extracted from the 1994 US Census database. We predict whether an individual has an income higher than 50K, with gender as the sensitive attribute.

COMPAS contains arrest records, demographic information, and criminal history of defendants arrested in 2013–2014. We predict reoffense within two years, with race as the sensitive attribute.

German contains records of individuals applying for credit or loan to a bank. We predict whether the default risk of an individual is high or low, using sex as the sensitive attribute.

ACSPublicCoverage and *ACSEmployment* have been recently proposed to create new and more challenging machine learning tasks with the aim to establish strong empirical evaluation practices within the algorithmic fairness community. They have been created starting from US Census data collected within the American Community Survey. The datasets include information related to ancestry, citizenship, education, race, employment, language proficiency, income, disability, etc. The two datasets differ in the prediction task: ACSPublicCoverage contains data for predicting whether an individual is covered by public health insurance, ACSEmployment contains data for predicting whether an individual is employed. We select race (RAC1P in the dataset) as the sensitive attribute. Unlike other datasets, in this case, the sensitive attribute is multi-valued.

Models, hyperparameters, and other settings. We perform all experiments with two different base models: logistic regression and gradient-boosted decision trees, implemented using *LogisticRegression* and *HistGradientBoostingClassifier* classes from the *scikit-learn* library. Their hyperparameters are tuned separately for each dataset, without fairness constraints, using 5-fold crossvalidation. In the paper, we only report the results for logistic regression, but results for gradient-boosted decision trees are very similar, and can be found in the

Table 7.1 The datasets used in the experiments. Size is the dimension of the dataset, n is the number of data points (#rows), d is the number of features (#columns), S and |S| are the name and the cardinality of the sensitive attribute.

Dataset	**Size (MB)**	n	d	**S**	**|S|**
ACSEmployment	319.47	3.2×10^6	99	RAC1P	9
ACSPublicCoverage	163.26	1.1×10^6	140	RAC1P	9
Adult	4.67	4.5×10^4	9	Sex	2
COMPAS	0.37	4.2×10^3	3	Race	2
German	0.05	1.0×10^3	9	Sex	2

technical report [9]. We consider two types of fairness constraints: demographic parity and equalized odds. The violation of fairness constraints is quantified using the demographic parity difference and the equalized odds difference (see Section 7.1).

For each experiment configuration, we evaluate the performance of each algorithm using stratified 3-fold cross validation, executed twice with different random seeds, resulting in six replications of each experiment. Stratification is performed by jointly considering the sensitive attribute and the label, ensuring a consistent sampling approach for both the training and testing splits. Each fold constitutes one-third of the entire dataset; consequently, in each iteration, the training set encompasses two-thirds, while the test set comprises one-third of the dataset. We have verified the consistency between the performance of the baseline models in our pipeline with a 70/30 train-test split as performed in other prior works.

Baselines. We compare ExpGrad^{++} with six baselines. CALMON [27] and FELD [51] are selected as representative pre-processing approaches, ZAFAR, which proposes two methods to enforce disparate impact (DI) [145] and equalized odds (EO) [146], as a representative in-processing approach, and HARDT [62] as a representative post-processing approach. Moreover, we include ExpGrad [1], the approach we are extending with column generation and sampling, and UNMITIGATED, i.e., the fairness-unaware approach. Not all the baselines can be used for all the experiments. In particular, ZAFAR EO (with equalized odds does) not support multi-valued sensitive attributes (i.e., it cannot be run on the datasets ACS*), and CALMON requires specifying a problem-specific distortion function, which was not available for ACS* datasets.

7.3.2 End-to-end evaluation

The goal of our first set of experiments is to demonstrate that ExpGrad^{++} effectively and efficiently supports the development of ML pipelines. As its input, it requires a bound on

the allowed amount of fairness violation. As we show, it returns an accurate classifier that satisfies the fairness constraint, while exhibiting satisfactory running time.

Implementation. EXPGRAD^{++} requires specification of the allowed constraint violation in Eq. (7.3). We consider five values for ε, i.e., 0.005, 0.01, 0.02, 0.05, 0.10, and 0.15 for our model. In ACS* datasets we do not evaluate at the value $\varepsilon = 0.15$, because the test error of EXPGRAD^{++} already matches the UNMITIGATED approach at $\varepsilon = 0.10$. In EXPGRAD^{++}, we use the sampling ratio of 0.25 with the datasets ACSPublicCoverage and ACSEmployment, and do not use subsampling on the three smaller datasets.

In Figure 7.1, we compare the performance of EXPGRAD^{++} with baselines across the 5 datasets, for both demographic parity (Figure 7.1a) as well as equalized odds (Figure 7.1b). For each of the datasets and each of the constraint types, we plot the test error and the running time required to train the model as a function of the constraint violation on test data. EXPGRAD^{++} and EXPGRAD are plotted as curves because they allow various fairness-accuracy trade-offs by specifying the bound ε. Other methods optimize a fixed trade-off and are plotted as points. Additional results, including the train error and train constraint violation, are available in [9].

Discussion. In Figure 7.1a we consider demographic parity. The top row shows the test error as a function of the test fairness violation. Points towards lower left generally represent better models, with a lower test error and a lower test constraint violation. Methods differ quite drastically in terms of their fairness violation (see the scale of the x-axis), but tend to be closer to each other in terms of error (see the scale of the y-axis), especially on the three small datasets (Adult, COMPAS, German). On the three small datasets, we see that EXPGRAD^{++} always matches or outperforms other methods. On the two larger datasets (which we simply refer to as ACS*), the post-processing approach HARDT achieves a better fairness-accuracy trade-off than EXPGRAD^{++}. However, recall that HARDT requires access to the sensitive attribute at the inference time, which is not possible in many real-world scenarios. Notice that the unmitigated model always achieves the lowest error for all datasets (apart from German), but suffers from a high violation of the fairness metric.

Comparing EXPGRAD^{++} with EXPGRAD, we observe that column generation and sampling do not have a negative impact on the fairness and accuracy; in fact, it appears that on ACS Employment, EXPGRAD^{++} achieves better trade-offs than EXPGRAD due to a faster convergence within the default maximum number of iterations T. In German, EXPGRAD^{++} appears to perform slightly worse at low constraint violations, but the differences are not statistically significant (see diagrams with the error bars in the technical report [9]). The plots also show that EXPGRAD^{++} effectively navigates fairness-accuracy tradeoff. As the fairness-constraint violation increases, EXPGRAD^{++} matches the test error of the unmitigated model.

In the second row of Figure 7.1a, we show the training runtimes of all the methods. We observe that the unmitigated approach and post-processing are the fastest. The running time of ExpGrad^{++} is similar, or better than the running time of the preprocessing and inprocessing baselines (with the exception of FELD on the two smallest datasets: COMPAS and German). Importantly, ExpGrad^{++} achieves almost an order of magnitude speed-up compared with ExpGrad without any sacrifices to the quality of returned solution.

The evaluation of equalized odds in Figure 7.1b leads to similar conclusions. ExpGrad^{++} achieves generally the best accuracy-fairness tradeoff, with the only exception being HARDT on ACS Employment (but recall that HARDT requires access to sensitive features at inference time). The unmitigated and the post-processing approaches are the fastest approaches, but ExpGrad^{++} matches or outperforms the other approaches in terms of running time (except for FELD on the two smallest datasets). Again, we see that ExpGrad^{++} speeds up over ExpGrad by almost an order of magnitude, without sacrificing the solution quality.

Takeaway. ExpGrad^{++} satisfies the desiderata for a fair ML approach introduced in Section 7: users can select the metric, adjust the fairness constraint, and apply the approach on large datasets with multi-valued sensitive attributes. The experiment confirms (and extends to the time performance) what emerged in the survey [68] about the effectiveness: the approaches generally trade accuracy for fairness: increasing ε decreases the test error (and decreases the execution time in big datasets).

7.3.3 Impact of column generation

We next conduct an ablation study to evaluate the impact of the column generation component in the ExpGrad^{++} algorithm.

Implementation. For the sake of simplicity, we show the results on the Adult, COMPAS, and German datasets only, and we do not perform any subsampling. We compare the performance of ExpGrad^{++} with and without the column generation component; the version without column generation corresponds to the ExpGrad approach in Algorithm 2. In both versions, we set the allowed constraint violation to $\varepsilon = 0.005$. We run ExpGrad^{++} with the default learning rate and default termination condition. For ExpGrad, we consider three different learning rates (specified in the *fairlearn* library via the hyperparameter `eta0` $\in \{0.5, 1.0, 2.0\}$), and for each, we take the best-performing solution among the three solutions (according to the duality gap). Note that we only report the time of one run, but not all three—this gives an advantage to ExpGrad.

In Figure 7.2, we show how well the two algorithms optimize accuracy and fairness as a function of time. We focus on training metrics because these capture the progress of optimization and the convergence condition that stops the iterations is verified over training data. (test metrics were reported in end-to-end evaluation). For EXPGRAD^{++}, we plot the accuracy and fairness after reaching the termination condition based on the duality gap (since the algorithm always reaches the early termination condition), but for EXPGRAD, we plot the performance at multiple different iterates, corresponding to different values of t. This again possibly gives EXPGRAD some advantage, because we do not require it to know when to stop.

Discussion. Figure 7.2 clearly shows that adding the column generation improves the efficiency of the original EXPGRAD. The new algorithm, EXPGRAD^{++}, reaches the solution of the same quality as eventually found by EXPGRAD in substantially fewer iterations, and hence with a substantially lower running time.

Focusing on the second row of both demographic parity and equalized odds experiments, note that the original algorithm EXPGRAD starts with a solution that has a larger than desired constraint violation, and as the constraint violation decreases, the training error slightly decreases (as the first row of the figures shows; in particular on Adult and demographic parity). This is the fairness-accuracy tradeoff that we also observed in the end-to-end evaluation. One exception to this pattern is the results for equalized odds on the Adult dataset. In this case, we achieve the desired fairness violation early in the optimization, and so the optimization focuses on decreasing the training error. In all cases, EXPGRAD^{++} is able to reach the final point of EXPGRAD in many fewer iterations.

Takeaway. The column generation clearly speeds up EXPGRAD^{++}, allowing it to achieve the same quality of solutions as EXPGRAD in many fewer iterations.

7.3.4 Robustness to sampling

In this Section, we investigate how the sampling ratio impacts the running time and the quality of solutions (i.e., test error and fairness violation) produced by EXPGRAD^{++}.

Implementation. In all experiments, we set the allowed constraint violation to $\varepsilon = 0.005$. We evaluate our *adaptive subsampling* approach for the sampling ratios $\rho \in \{0.001, 0.004, 0.016, 0.063, 0.251\}$. As a baseline, we also consider *static subsampling*, where the dataset is subsampled (uniformly at random) once at the beginning and then EXPGRAD^{++} is run on the subsampled dataset (without any further subsampling). In both cases, we run EXPGRAD^{++} with column generation and a default setting of optimization hyperparameters. As before, we consider both demographic parity and equalized odds.

Additionally, we also report performance on unmitigated approach trained on the same subsampled dataset as the static ExpGrad^{++}.

In Figure 7.3 we show how the running time, test error, and test constraint violation vary as a function of sampling ratio.

Discussion. As expected, adaptive sampling generally shows a similar running time as static sampling in all cases except for ACS Employment with demographic parity, where static sampling is substantially slower at low sampling ratios. This is perhaps because static subsampling severely undersamples small groups which leads to harder optimization problems to solve.

Across the board we see that both adaptive and static sampling achieve similar levels of fairness—this level is constant as a function of the sampling ratio, because in all cases we set $\varepsilon = 0.005$. A major exception to this is Adult with equalized odds, where static sampling fails to achieve the desired level of fairness violation at the lowest sampling ratio. As before, this is because severe subsampling makes the underlying optimization problem much harder, so the algorithm fails to find a feasible solution within the default number of iterations $T = 50$. We also see that the adaptive approach achieves slightly worse fairness violations on ACS* datasets with equalized odds; this is because in this case static sampling restricts the optimization space towards trivial solutions (e.g., constant decisions), which generally achieve low fairness violations, but large errors (as is the case here).

Considering the comparison between ExpGrad^{++}, ExpGrad without column generation (CG), with adaptive sampling and static sampling, we observe that for small datasets (Adult, COMPAS, German) ExpGrad without CG and with adaptive sampling performs as well as or slightly better than than ExpGrad^{++} in terms of accuracy and fairness. However, ExpGrad is always slower. For large datasets, ExpGrad without CG and with adaptive sampling fails to achieve the same level of fairness as ExpGrad^{++}. The iterations reach the limit without converging. However, ExpGrad has a lower error and is faster than ExpGrad^{++}. This indicates that CG enhances also the capability of reaching better fairness. Moreover, we observe that also in ExpGrad without CG, the static sampling is generally converging slower in terms of accuracy but is faster than the adaptive sampling. An exception is the ACSPublicCoverage dataset, where ExpGrad with adaptive sampling outperforms ExpGrad^{++} in both accuracy and time, while reaching the same fairness level. We attribute this to a generalization problem between train and test data (see Figure 7.3b where we show the fairness on both the train and the test data). ExpGrad^{++} meets the fairness requirements on train data whereas ExpGrad without CG struggles to satisfy the constraint. The stringent fairness constraint on training data forces ExpGrad^{++} to sacrifice

some accuracy on train data and test failing to maintain the same fairness levels on test data. Consequently ExpGrad appears to be the better performer in this specific context.

Finally, note that adaptive sampling always achieves lower test error than static sampling (up to statistical uncertainty shown by error bars). This points to the fact that it leads to a more informative sample of the data. Importantly, for the adaptive sampling approach, we see that at fairly low sampling rates (0.01 on the two large datasets and 0.1 on the three small datasets), the test error is already close to the test error using the entire dataset (sampling ratio of 1.0). Also, as expected, the unmitigated method is generally faster and achieves a low test error, but the solutions that it produces severely violate the fairness constraints.

Takeaway. ExpGrad^{++} with adaptive sampling is robust to the sampling size. Adaptively selecting a small subsample of the dataset (around 0.01 on larger datasets and 0.1 on smaller ones) does not decrease test accuracy and does not lead to a worse fairness violation. Compared with static sampling, adaptive sampling uses the data better and achieves greater model accuracy, especially for large datasets. For equalized odds, this occurs at the expense of a slightly worse violation of fairness constraints.

7.3.5 Multi-valued sensitive attributes

Here we demonstrate that although many fairness mitigation approaches assume that sensitive attributes are binary, full support of multi-valued sensitive attributes (as done by ExpGrad^{++}) is necessary to mitigate fairness violations.

Implementation. We perform experiments with the ACSEmployment dataset, a large dataset with a multi-valued sensitive attribute (RAC1P). We either consider the multi-valued sensitive attribute as is, or consider its binarized version. In binarized versions, we split 9 race categories into two groups based on the average employment rate in each group, relative to the median employment rate (across all 9 groups).

In Figure 7.4 we show the fairness violations using the *multi-valued* variants of demographic parity difference and equalized odds difference, where for each method we consider training it using either multi-valued or binarized sensitive attribute.

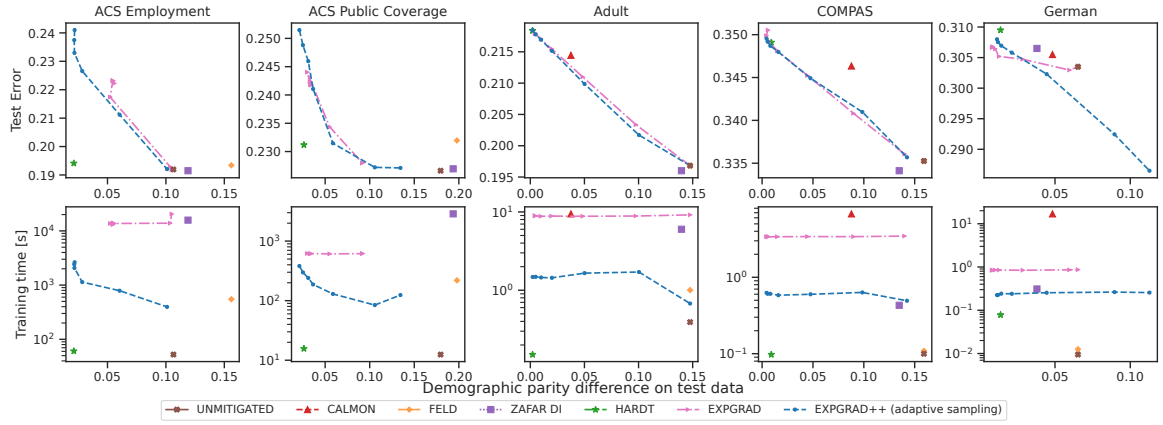
Discussion. None of the methods effectively reduce the fairness with respect to the multi-valued attribute when trained on binarized data. FELD and ZAFAR also do not effectively reduce the fairness violation, when they are trained on multi-valued data (in fact, it seems that the fairness violation gets worse than with binarized training). On the other hand, ExpGrad^{++} and HARDT, both effectively reduce unfairness when trained on multi-valued data. This is to be expected as both methods provably find solutions that satisfy multi-valued fairness.

Takeaway. Real-world scenarios require the support of multi-valued sensitive attributes. EXPGRAD^{++} and HARDT both effectively incorporate fairness constraints with such attributes (albeit HARDT requires access to the sensitive attribute at inference time, which is not the case for EXPGRAD^{++}).

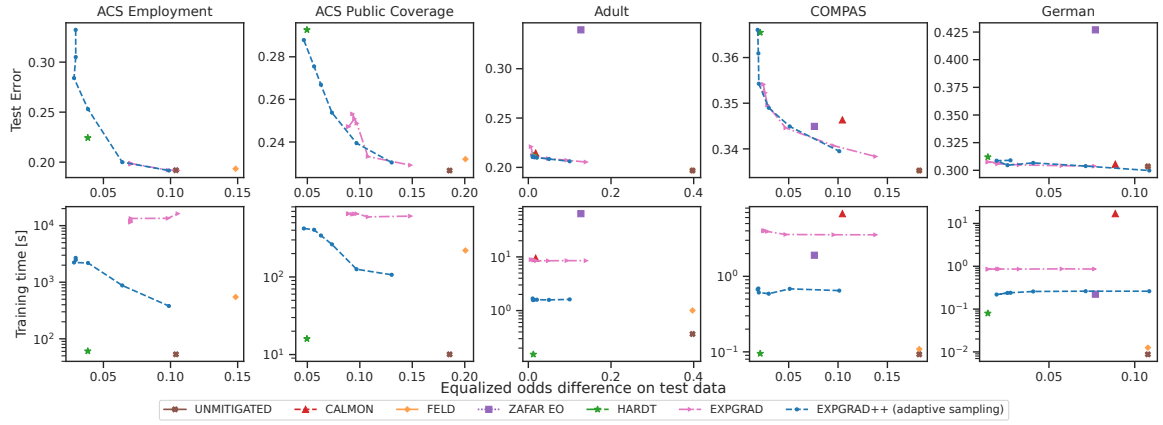
7.4 Conclusion

We have introduced EXPGRAD^{++}, a reduction approach that effectively and efficiently incorporates fairness constraints into standard ML algorithms. EXPGRAD^{++} allows users to select their preferred notion of fairness and the allowed violation of fairness constraints. The approach can handle multi-valued sensitive attributes and, as a reduction approach, can be applied to any ML model family. EXPGRAD^{++} overcomes the scalability limitations of the previous approach, EXPGRAD, by incorporating two innovations: column generation and adaptive subsampling. As a result, EXPGRAD^{++} can be applied to real-world scenarios characterized by large datasets.

Our empirical evaluation, which includes both small and large datasets with both binary and multi-valued sensitive attributes, shows that EXPGRAD^{++} is an effective, efficient, robust and easy-to-tune approach for the development of end-to-end ML pipelines.

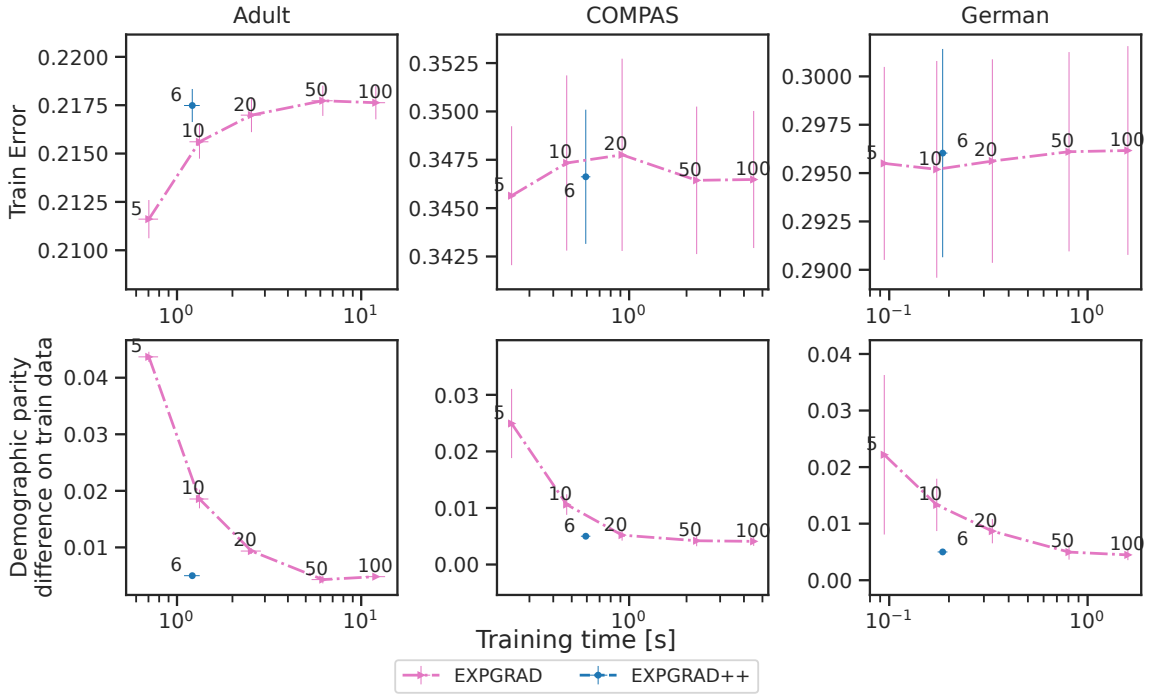


(a) Demographic Parity

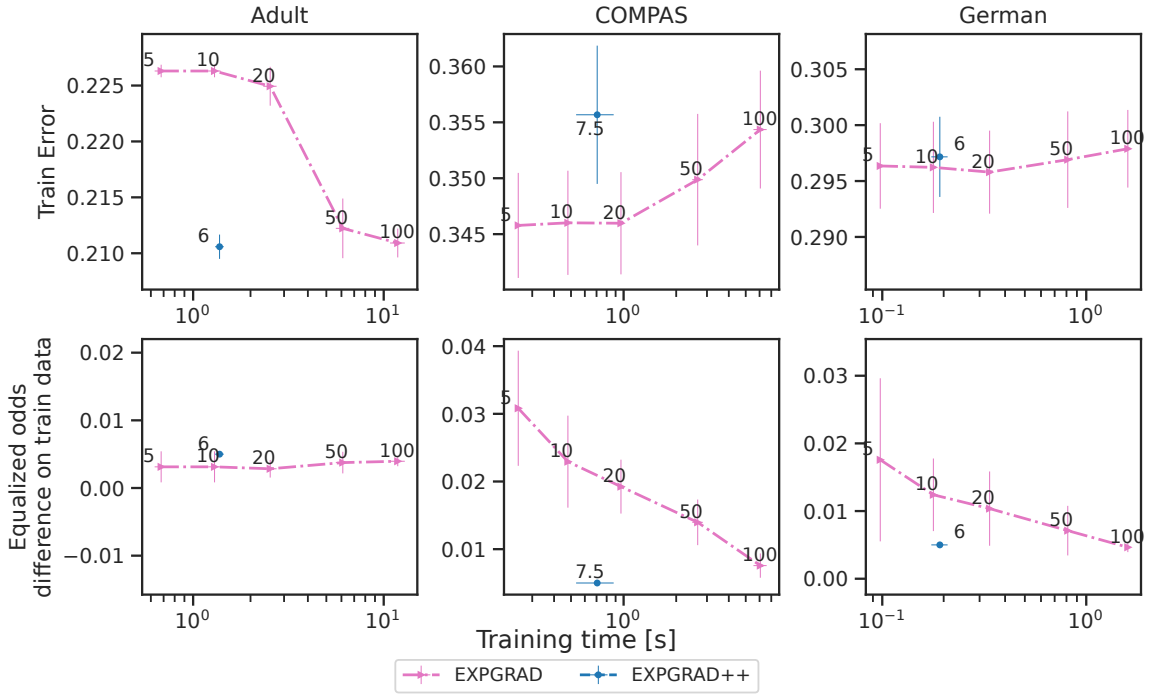


(b) Equalized Odds

Fig. 7.1 *End-to-end comparison of* ExpGrad^{++} *with baselines for two fairness measures (demographic parity and equalized odds), across 5 datasets.* Plotting test error and training time as a function of test fairness violation, averaged over 6 replicates. ExpGrad^{++} and ExpGrad shown as curves, because they allow tuning of fairness–accuracy trade-off; other methods shown as points. Results closer to lower-left corner of subplots are more favorable. Plots show that ExpGrad^{++} achieves the best tradeoff between accuracy and fairness, except HARDT, which however requires access to the sensitive attribute at inference time. ExpGrad^{++} successfully decreases the training runtime compared with ExpGrad and matches the runtime of other methods except HARDT and UNMITIGATED (and FELD on the two smallest datasets).

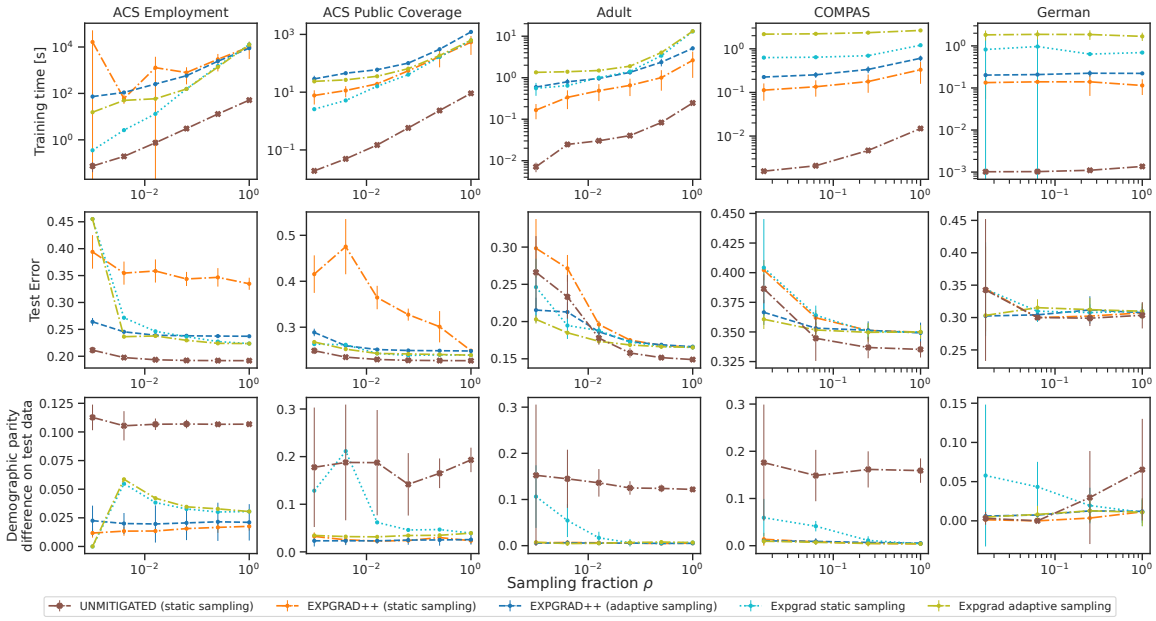


(a) Demographic Parity.

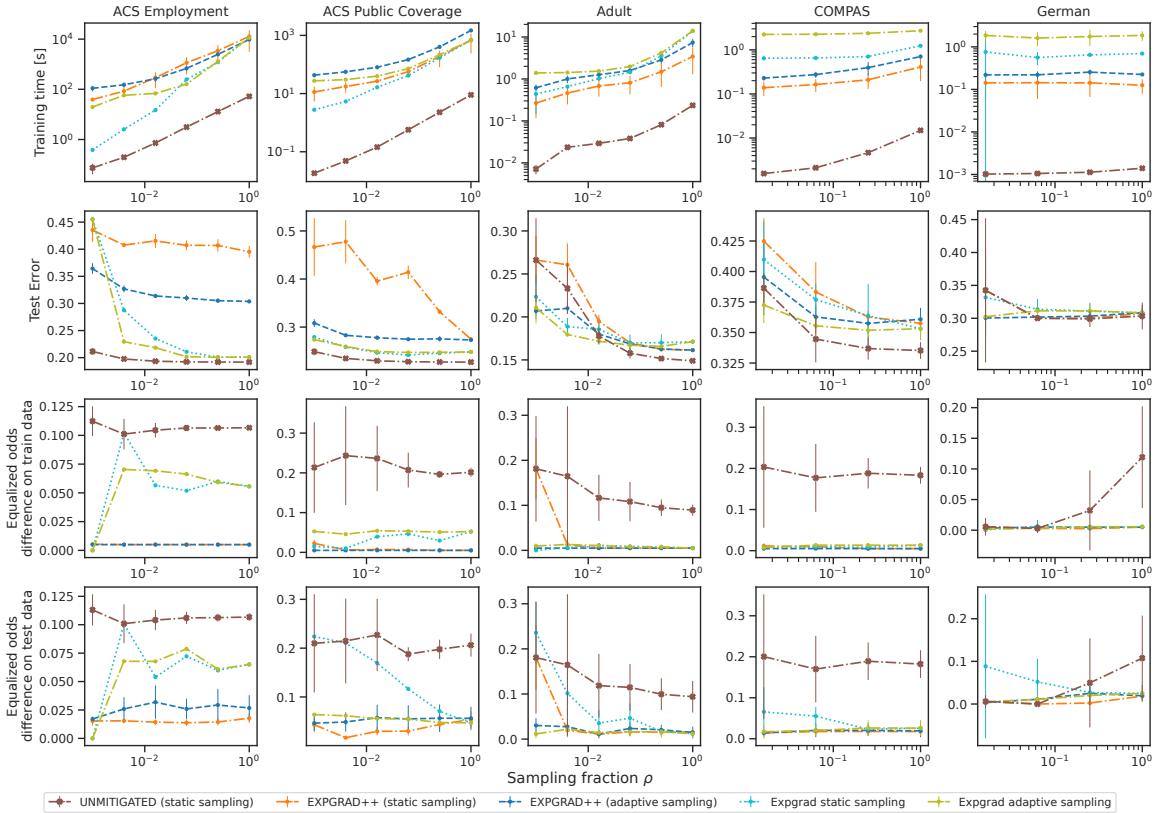


(b) Equalized odds.

Fig. 7.2 *Impact of column generation.* Plotting train error and train fairness violation as a function of training runtime, for two fairness measures (demographic parity and equalized odds) and three datasets, averaged over 6 replicates. For ExpGrad, showing the performance at iterations ranging from 5 to 100 (indicated by numerical labels); for ExpGrad^{++}, showing the performance at convergence (with the numerical label indicating the average number of iterations). Plots show that ExpGrad^{++} requires fewer iterations and thus less runtime to reach a solution of the same quality as ExpGrad.



(a) Demographic Parity.



(b) Equalized Odds.

Fig. 7.3 *Robustness to sampling.* Plotting train runtime, test error, and test fairness violation as a function of sampling ratio; comparing adaptive and static sampling in ExpGrad^{++}. Plots show that both adaptive and static sampling achieve low fairness violation, but adaptive sampling is making better use of data (lower test error). Sampling ratios as low as 0.1 (or even less) yield improved runtime without sacrificing accuracy.

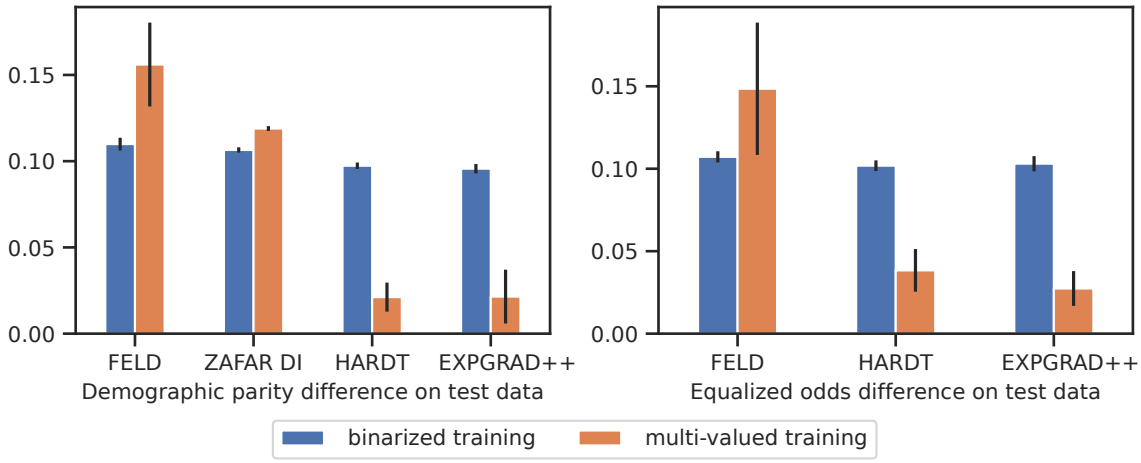


Fig. 7.4 *Comparison of training with multi-valued and binarized sensitive attribute.* Test fairness violation is measured using a multi-valued attribute. Plots show that binarized training does not effectively decrease the fairness violation. With multi-valued training, HARDT and ExpGrad^{++} successfully reduce fairness violation (but other methods do not). Note that Zafar EO is only applicable to binary sensitive attributes, so it is omitted in the equalized odds evaluation.

Chapter 8

Conclusions and Future Work

This thesis advances the understanding and application of explainability in artificial intelligence through three main research directions: (1) novel explainable AI techniques for entity matching and deep learning analysis in data integration (Chapters 3, 5, and 4), (2) transparent computational fact-checking systems that demonstrate how evidence is utilized in veracity prediction (Chapter 6), and (3) scalable fair AI methods that maintain accuracy while ensuring fairness guarantees (Chapter 7).

8.1 Summary of Contributions

Explainable Entity Matching The research introduced two significant approaches concerning explainable entity matching. First, Landmark Explanationextended the capabilities of post-hoc perturbation-based explainers to effectively handle the unique challenges of entity-matching datasets. This system generates dual explanations from different entity perspectives and employs token injection to improve explanation quality for non-matching records. Second, WYM(Why do You Match?) is presented, an intrinsically interpretable entity-matching architecture based on decision units. This novel approach achieved comparable accuracy to state-of-the-art techniques while providing transparent explanations for its decisions.

Understanding Deep Learning Models in Data Integration The analysis of BERT's behavior in entity-matching tasks revealed important insights into how transformer architectures process structured data. The research demonstrated that BERT can effectively recognize dataset structure and extract semantic knowledge beyond simple pair-wise token associations. Notably, the fine-tuning process enables BERT to distribute attention patterns based

on whether descriptions refer to matching or non-matching entities, providing a valuable understanding of how these models achieve their high performance.

Evidence-based Fact-checking and Fair AI In the domain of computational fact-checking, the research developed a framework for explaining how models utilize evidence in their decision-making process. The work demonstrated that post-hoc explanation methods can match or exceed the quality of intrinsic explainers while offering greater flexibility.

Efficient Fair Classification Additionally, the development of ExpGrad^{++} overcomes the scalability limitations of the previous approach, by incorporating two innovations: column generation and adaptive subsampling. As a result, ExpGrad^{++} can be applied to real-world scenarios characterized by large datasets.

8.2 Final Remarks

This thesis has made significant strides in making AI systems more transparent, interpretable, and fair while maintaining their effectiveness. The developed methods and insights provide a foundation for future research in explainable AI, particularly in domains where interpretability is crucial for practical adoption. As AI systems continue to be deployed in critical applications, the importance of explainability and fairness will only grow, making these contributions increasingly relevant to both research and practice.

The work presented here demonstrates that it is possible to develop AI systems that are both highly effective and interpretable, challenging the notion that there must be a trade-off between performance and explainability. This opens new possibilities for the responsible development and deployment of AI systems across various domains.

References

[1] Agarwal, A., Beygelzimer, A., Dudík, M., Langford, J., and Wallach, H. (2018). A Reductions Approach to Fair Classification. In *International Conference on Machine Learning*, pages 60–69.

[2] Agarwal, A., Dudík, M., and Wu, Z. S. (2019). Fair Regression: Quantitative Definitions and Reduction-Based Algorithms. In *International Conference on Machine Learning*, pages 120–129.

[3] Aly, R., Guo, Z., Schlichtkrull, M. S., Thorne, J., Vlachos, A., Christodoulopoulos, C., Cocarascu, O., and Mittal, A. (2021a). FEVEROUS: Fact extraction and VERification over unstructured and structured information. In *NeurIPS (Datasets and Benchmarks)*.

[4] Aly, R., Guo, Z., Schlichtkrull, M. S., Thorne, J., Vlachos, A., Christodoulopoulos, C., Cocarascu, O., and Mittal, A. (2021b). FEVEROUS: fact extraction and verification over unstructured and structured information. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

[5] Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016). Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[6] Arrieta, A. B., Rodríguez, N. D., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 58:82–115.

[7] Atanasova, P., Simonsen, J. G., Lioma, C., and Augenstein, I. (2020a). Generating fact checking explanations.

[8] Atanasova, P., Simonsen, J. G., Lioma, C., and Augenstein, I. (2020b). Generating fact checking explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7352–7364. Association for Computational Linguistics.

[9] Authors, A. (2024). Fair Classification with a Scalable Reduction Approach (Technical Report). https://anonymous.4open.science/r/SIGMOD2025-622/TechnicalReport.pdf.

[10] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

[11] Baraldi, A., Buono, F. D., Guerra, F., Paganelli, M., and Vincini, M. (2023). An intrinsically interpretable entity matching system. In *EDBT*, pages 645–657. OpenProceedings.org.

[12] Baraldi, A., Buono, F. D., Paganelli, M., and Guerra, F. (2021a). Landmark explanation: An explainer for entity matching models. In *CIKM*, pages 4680–4684. ACM.

[13] Baraldi, A., Buono, F. D., Paganelli, M., and Guerra, F. (2021b). Using Landmarks for Explaining Entity Matching Models. In *EDBT*.

[14] Barlaug, N. (2022). Lemon: Explainable entity matching. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–16.

[15] Barlaug, N. and Gulla, J. A. (2021). Neural networks for entity matching: A survey. *ACM Trans. Knowl. Discov. Data*, 15(3):52:1–52:37.

[16] Barocas, S., Hardt, M., and Narayanan, A. (2019). *Fairness and Machine Learning: Limitations and Opportunities*. fairmlbook.org. http://www.fairmlbook.org.

[17] Becker, B. and Kohavi, R. (1996). Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

[18] Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilovic, A., Nagar, S., Ramamurthy, K. N., Richards, J. T., Saha, D., Sattigeri, P., Singh, M., Varshney, K. R., and Zhang, Y. (2018). AI fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *CoRR*, abs/1810.01943.

[19] Benjamin, R. (2019). *Race After Technology: Abolitionist Tools for the New Jim Code*. Polity.

[20] Bilenko, M., Mooney, R. J., Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003). Adaptive name matching in information integration. *IEEE Intell. Syst.*, 18(5):16–23.

[21] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146.

[22] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[23] Broussard, M. (2018). *Artificial Unintelligence: How Computers Misunderstand the World*. MIT Press.

[24] Brunner, G., Liu, Y., Pascual, D., Richter, O., Ciaramita, M., and Wattenhofer, R. (2020). On identifiability in transformers. In *ICLR*. OpenReview.net.

[25] Brunner, U. and Stockinger, K. (2020). Entity matching with transformer architectures - A step forward in data integration. In *EDBT*, pages 463–473. OpenProceedings.org.

[26] Buolamwini, J. and Gebru, T. (2018). Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, pages 77–91.

[27] Calmon, F. P., Wei, D., Vinzamuri, B., Ramamurthy, K. N., and Varshney, K. R. (2017). Optimized pre-processing for discrimination prevention. In *NIPS*, pages 3992–4001.

[28] Cao, S., Sanh, V., and Rush, A. M. (2021). Low-complexity probing via finding subnetworks. *CoRR*, abs/2104.03514.

[29] Caton, S. and Haas, C. (2023). Fairness in machine learning: A survey. *ACM Comput. Surv.*

[30] Cavazos, R. (2019). The economic cost of bad actors on the internet. fake news in 2019. Commissioned by CHEQ, in collaboration with the University of Baltimore's Merrick School of Business.

[31] Chen, J., Song, L., Wainwright, M. J., and Jordan, M. I. (2018). Learning to explain: An information-theoretic perspective on model interpretation. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 882–891. PMLR.

[32] Chen, Z., Chen, Q., Hou, B., Li, Z., and Li, G. (2020). Towards interpretable and learnable risk analysis for entity resolution. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1165–1180.

[33] Cicco, V. D., Firmani, D., Koudas, N., Merialdo, P., and Srivastava, D. (2019). Interpreting deep learning models for entity resolution: an experience report using LIME. In *aiDM@SIGMOD*, pages 8:1–8:4. ACM.

[34] Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. (2019). What does BERT look at? an analysis of bert's attention. *CoRR*, abs/1906.04341.

[35] Crampton, N. (2022). Microsoft's framework for building AI systems responsibly. https://blogs.microsoft.com/on-the-issues/2022/06/21/microsofts-framework-for-building-ai-systems-responsibly/.

[36] Crawford, K. (2013). The Hidden Biases in Big Data. *Harvard business review*, 1(4).

[37] Crawford, K. (2017). The Trouble with Bias. NeurIPS keynote. https://www.youtube.com/watch?v=fMym_BKWQzk.

[38] Dammu, P. P. S., Naidu, H., Dewan, M., Kim, Y., Roosta, T., Chadha, A., and Shah, C. (2024). Claimver: Explainable claim-level verification and evidence attribution of text through knowledge graphs.

[39] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

[40] Ding, F., Hardt, M., Miller, J., and Schmidt, L. (2021). Retiring adult: New datasets for fair machine learning. In *NeurIPS*, pages 6478–6490.

[41] Du, M., Liu, N., and Hu, X. (2020). Techniques for interpretable machine learning. *Commun. ACM*, 63(1):68–77.

[42] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., and et al., D. C. (2024). The llama 3 herd of models.

[43] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. S. (2012). Fairness through awareness. In *ITCS*, pages 214–226. ACM.

[44] Ebaid, A., Thirumuruganathan, S., Aref, W. G., Elmagarmid, A., and Ouzzani, M. (2019). Explainer: Entity resolution explanations. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 2000–2003. IEEE.

[45] Ebraheem, M., Thirumuruganathan, S., Joty, S. R., Ouzzani, M., and Tang, N. (2018). Distributed representations of tuples for entity resolution. *Proc. VLDB Endow.*, 11(11):1454–1467.

[46] (EC), E. (2021). Proposal for a Regulation of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain Union legislative acts. COM (2021) 206 final.

[47] Eisemann, K. (1957). The trim problem. *Management Science*, 3(3):279–284.

[48] Eisenschlos, J. M., Dhingra, B., Bulian, J., Börschinger, B., and Boyd-Graber, J. (2021). Fool me twice: Entailment from wikipedia gamification.

[49] Fabris, A., Messina, S., Silvello, G., and Susto, G. (2022). Algorithmic fairness datasets: the story so far. *Data Mining and Knowledge Discovery*, 36:2074–2152.

[50] Feige, U., Mirrokni, V. S., and Vondrák, J. (2011). Maximizing non-monotone submodular functions. *SIAM J. Comput.*, 40(4):1133–1153.

[51] Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., and Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In *KDD*, pages 259–268. ACM.

[52] Feng, S., Wallace, E., II, A. G., Iyyer, M., Rodriguez, P., and Boyd-Graber, J. L. (2018). Pathologies of neural models make interpretation difficult. In *EMNLP*, pages 3719–3728. Association for Computational Linguistics.

[53] Freund, Y. and Schapire, R. E. (1996). Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory (COLT)*, pages 325–332.

[54] Ghorbani, A. and Zou, J. Y. (2019). Data shapley: Equitable valuation of data for machine learning. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR.

[55] Gladbach, M., Sehili, Z., Kudrass, T., Christen, P., and Rahm, E. (2018). Distributed privacy-preserving record linkage using pivot-based filter techniques. In *ICDE Workshops*, pages 33–38. IEEE Computer Society.

[56] Goldberg, Y. (2019). Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287.

[57] Google (2023). Google AI principles progress update 2023. https://ai.google/static/documents/ai-principles-2023-progress-update.pdf.

[58] Griva, I., Nash, S. G., and Sofer, A. (2008). *Linear and Nonlinear Optimization*. SIAM, 2nd edition.

[59] Guo, Z., Schlichtkrull, M. S., and Vlachos, A. (2022). A survey on automated fact-checking. *Trans. Assoc. Comput. Linguistics*, 10:178–206.

[60] Hao, Y., Dong, L., Wei, F., and Xu, K. (2019). Visualizing and understanding the effectiveness of BERT. In *EMNLP/IJCNLP (1)*, pages 4141–4150. Association for Computational Linguistics.

[61] Hao, Y., Dong, L., Wei, F., and Xu, K. (2020). Investigating learning dynamics of BERT fine-tuning. In *AACL/IJCNLP*, pages 87–92. Association for Computational Linguistics.

[62] Hardt, M., Price, E., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*. https://arxiv.org/abs/1610.02413.

[63] Hewitt, J. and Liang, P. (2019). Designing and interpreting probes with control tasks. In *EMNLP/IJCNLP (1)*, pages 2733–2743. Association for Computational Linguistics.

[64] Hewitt, J. and Manning, C. D. (2019). A structural probe for finding syntax in word representations. In *NAACL-HLT (1)*, pages 4129–4138. Association for Computational Linguistics.

[65] Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77.

[66] Hort, M., Chen, Z., Zhang, J. M., Sarro, F., and Harman, M. (2022). Bias mitigation for machine learning classifiers: A comprehensive survey. *CoRR*, abs/2207.07068.

[67] Htut, P. M., Phang, J., Bordia, S., and Bowman, S. R. (2019). Do attention heads in BERT track syntactic dependencies? *CoRR*, abs/1911.12246.

[68] Islam, M. T., Fariha, A., Meliou, A., and Salimi, B. (2022). Through the data management lens: Experimental analysis and evaluation of fair classification. In *SIGMOD Conference*, pages 232–246. ACM.

[69] Iwama, K. and Miyazaki, S. (2008). A survey of the stable marriage problem and its variants. In *International Conference on Informatics Education and Research for Knowledge-Circulating Society (icks 2008)*, pages 131–136.

[70] Jain, S. and Wallace, B. C. (2019). Attention is not explanation. *CoRR*, abs/1902.10186.

[71] Jr., J. R. B. (2023). Executive order on the safe, secure, and trustworthy development and use of artificial intelligence. https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence/.

[72] Jui, T. D. and Rivas, P. (2024). Fairness issues, current approaches, and challenges in machine learning models. *Int. J. Mach. Learn. Cybern.*, 15(8):3095–3125.

[73] Kamiran, F. and Calders, T. (2012). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33.

[74] Kivinen, J. and Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63.

[75] Koenecke, A., Nam, A., Lake, E., Nudell, J., Quartey, M., Mengesha, Z., Toups, C., Rickford, J. R., Jurafsky, D., and Goel, S. (2020). Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, 117(14):7684–7689.

[76] Kotonya, N. and Toni, F. (2020). Explainable automated fact-checking for public health claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7740–7754, Online. Association for Computational Linguistics.

[77] Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. (2019). Revealing the dark secrets of BERT. In *EMNLP/IJCNLP (1)*, pages 4364–4373. Association for Computational Linguistics.

[78] Larson, J., Mattu, S., Kirchner, L., and Angwin, J. (2016). How we analyzed the COMPAS recidivism algorithm. ProPublica.

[79] Letham, B., Rudin, C., McCormick, T. H., Madigan, D., et al. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371.

[80] Li, J., Monroe, W., and Jurafsky, D. (2016). Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220.

[81] Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W. (2020a). Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60.

[82] Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. (2020b). Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60.

[83] Lin, Y., Tan, Y. C., and Frank, R. (2019). Open sesame: Getting inside bert's linguistic knowledge. *CoRR*, abs/1906.01698.

[84] Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., and Smith, N. A. (2019a). Linguistic knowledge and transferability of contextual representations. In *NAACL-HLT (1)*, pages 1073–1094. Association for Computational Linguistics.

[85] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[86] Loftus, J. R., Russell, C., Kusner, M. J., and Silva, R. (2018). Causal reasoning for algorithmic fairness. *CoRR*, abs/1805.05859.

[87] Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA. Curran Associates Inc.

[88] Matteo Paganelli, Andrea Baraldi, Francesco Del Buono, Francesco Guerra (2021). Analyzing How BERT Performs Entity Matching. https://www.dropbox.com/sh/n6dkvyv6smdw0da/AADh93zfShup3xvdrZKsZu1za?dl=0. Technical Report.

[89] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6).

[90] Meliou, X. W. L. H. A. (2018). Explaining data integration. *Data Engineering*, page 47.

[91] Michel, P., Levy, O., and Neubig, G. (2019). Are sixteen heads really better than one? In *NeurIPS*, pages 14014–14024.

[92] Molnar, C. (2018). Interpretable machine learning. https://christophm.github.io/interpretable-ml-book.

[93] Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., and Raghavendra, V. (2018). Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 19–34. ACM.

[94] Nakov, P., Corney, D. P. A., Hasanain, M., Alam, F., Elsayed, T., Barrón-Cedeño, A., Papotti, P., Shaar, S., and Martino, G. D. S. (2021). Automated fact-checking for assisting human fact-checkers. In *IJCAI*, pages 4551–4558. ijcai.org.

[95] Nie, Y., Williams, A., Dinan, E., Bansal, M., Weston, J., and Kiela, D. (2020). Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

[96] Noble, S. U. (2018). *Algorithms of Oppression: How Search Engines Reinforce Racism*. NYU Press.

[97] Obermeyer, Z., Powers, B., Vogeli, C., and Mullainathan, S. (2019). Dissecting Racial Bias in an Algorithm Used to Manage the Health of Populations. *Science*, 366(6464):447–453.

[98] O'Neil, C. (2016). *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown.

[99] (OSTP), W. (2022). A Blueprint for an AI Bill of Rights. https://www.whitehouse.gov/ostp/ai-bill-of-rights/.

[100] Paganelli, M., Buono, F. D., Baraldi, A., and Guerra, F. (2022). Analyzing how BERT performs entity matching. *Proc. VLDB Endow.*, 15(8):1726–1738.

[101] Paganelli, M., Buono, F. D., Pevarello, M., Guerra, F., and Vincini, M. (2021). Automated machine learning for entity matching tasks. In *EDBT*, pages 325–330. OpenProceedings.org.

[102] Paganelli, M., Sottovia, P., Guerra, F., and Velegrakis, Y. (2019). Tuner: Fine tuning of rule-based entity matchers. In *CIKM*, pages 2945–2948. ACM.

[103] Pan, L., Wu, X., Lu, X., Luu, A. T., Wang, W. Y., Kan, M., and Nakov, P. (2023). Fact-checking complex claims with program-guided reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6981–7004. Association for Computational Linguistics.

[104] Peeters, R. and Bizer, C. (2021). Dual-objective fine-tuning of BERT for entity matching. *Proc. VLDB Endow.*, 14(10):1913–1921.

[105] Pessach, D. and Shmueli, E. (2023). A review on fairness in machine learning. *ACM Comput. Surv.*, 55(3):51:1–51:44.

[106] Peters, M. E., Neumann, M., Zettlemoyer, L., and Yih, W. (2018). Dissecting contextual word embeddings: Architecture and representation. In *EMNLP*, pages 1499–1509. Association for Computational Linguistics.

[107] Precedence Research (2024). Data Integration Market Size, Share, and Trends 2024 to 2033. https://www.precedenceresearch.com/data-integration-market. Last Updated : April 2024, Report Code : 2744, Category : ICT.

[108] Raghavan, M., Barocas, S., Kleinberg, J., and Levy, K. (2020). Mitigating bias in algorithmic hiring: evaluating claims and practices. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, pages 469—-481, New York, NY, USA. Association for Computing Machinery.

[109] Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

[110] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In *SIGKDD*, pages 1135–1144.

[111] Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *AAAI*, pages 1527–1535. AAAI Press.

[112] Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in bertology: What we know about how BERT works. *Trans. Assoc. Comput. Linguistics*, 8:842–866.

[113] Rudin, C. (2018). Please stop explaining black box models for high stakes decisions. *CoRR*, abs/1811.10154.

[114] Saakyan, A., Chakrabarty, T., and Muresan, S. (2021). Covid-fact: Fact extraction and verification of real-world claims on COVID-19 pandemic. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2116–2129. Association for Computational Linguistics.

[115] Saeed, M., Alfarano, G., Nguyen, K., Pham, D., Troncy, R., and Papotti, P. (2021). Neural re-rankers for evidence retrieval in the FEVEROUS task. In Aly, R., Christodoulopoulos, C., Cocarascu, O., Guo, Z., Mittal, A., Schlichtkrull, M., Thorne, J., and Vlachos, A., editors, *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 108–112, Dominican Republic. Association for Computational Linguistics.

[116] Schlichtkrull, M., Guo, Z., and Vlachos, A. (2023). Averitec: A dataset for real-world claim verification with evidence from the web.

[117] Schuster, T., Fisch, A., and Barzilay, R. (2021). Get your vitamin C! robust fact verification with contrastive evidence. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 624–643, Online. Association for Computational Linguistics.

[118] Serrano, S. and Smith, N. A. (2019). Is attention interpretable? In *ACL (1)*, pages 2931–2951. Association for Computational Linguistics.

[119] Shahbazi, N., Danevski, N., Nargesian, F., Asudeh, A., and Srivastava, D. (2023). Through the fairness lens: Experimental analysis and evaluation of entity matching. *Proc. VLDB Endow.*, 16(11):3279–3292.

[120] Shelby, R., Rismani, S., Henne, K., Moon, A., Rostamzadeh, N., Nicholas, P., Yilla-Akbari, N., Gallegos, J., Smart, A., Garcia, E., et al. (2023). Sociotechnical harms of algorithmic systems: Scoping a taxonomy for harm reduction. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, pages 723–741.

[121] Si, J., Zhu, Y., and Zhou, D. (2023a). Consistent multi-granular rationale extraction for explainable multi-hop fact verification. *CoRR*, abs/2305.09400.

[122] Si, J., Zhu, Y., and Zhou, D. (2023b). Exploring faithful rationale for multi-hop fact verification via salience-aware graph learning. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

[123] Singh, R., Meduri, V. V., Elmagarmid, A. K., Madden, S., Papotti, P., Quiané-Ruiz, J., Solar-Lezama, A., and Tang, N. (2017). Synthesizing entity matching rules by examples. *Proc. VLDB Endow.*, 11(2):189–202.

[124] Smith, H. (2020). Algorithmic bias: should students pay the price? *AI & Society*, 35:1077–1078.

[125] Stoyanovich, J., Abiteboul, S., Howe, B., Jagadish, H. V., and Schelter, S. (2022). Responsible data management. *Commun. ACM*, 65(6):64–74.

[126] Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

[127] Tenney, I., Das, D., and Pavlick, E. (2019a). BERT rediscovers the classical NLP pipeline. In *ACL (1)*, pages 4593–4601. Association for Computational Linguistics.

[128] Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S. R., Das, D., and Pavlick, E. (2019b). What do you learn from context? probing for sentence structure in contextualized word representations. In *ICLR (Poster)*. OpenReview.net.

[129] Teofili, T., Firmani, D., Koudas, N., Martello, V., Merialdo, P., and Srivastava, D. (2022). Effective explanations for entity resolution models. In *ICDE*, pages 2709–2721. IEEE.

[130] Teong, K. S., Soon, L., and Su, T. T. (2020). Schema-agnostic entity matching using pre-trained language models. In *CIKM*, pages 2241–2244. ACM.

[131] Thirumuruganathan, S., Li, H., Tang, N., Ouzzani, M., Govind, Y., Paulsen, D., Fung, G. M., and Doan, A. (2021). Deep learning for blocking in entity matching: A design space exploration. *Proc. VLDB Endow.*, 14(11):2459–2472.

[132] Thirumuruganathan, S., Ouzzani, M., and Tang, N. (2019). Explaining entity resolution predictions: Where are we and what needs to be done? In *HILDA@SIGMOD*, pages 10:1–10:6. ACM.

[133] Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018). FEVER: a large-scale dataset for fact extraction and verification. In Walker, M. A., Ji, H., and Stent, A., editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.

[134] Vashishth, S., Upadhyay, S., Tomar, G. S., and Faruqui, M. (2019). Attention interpretability across NLP tasks. *CoRR*, abs/1909.11218.

[135] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *NIPS*, pages 5998–6008.

[136] Veale, M. and Binns, R. (2017). Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data. *Big Data Soc.*, 4(2):205395171774353.

[137] Wadden, D., Lin, S., Lo, K., Wang, L. L., van Zuylen, M., Cohan, A., and Hajishirzi, H. (2020). Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online. Association for Computational Linguistics.

[138] Wallach, H. and Dudík, M. (2021). Fairness-related harms in AI systems: Examples, assessment, and mitigation. Microsoft Research webinar. https://www.youtube.com/watch?v=1RptHwfkx_k.

[139] Wang, J., Li, G., Yu, J. X., and Feng, J. (2011). Entity matching: How similar is similar. *PVLDB*.

[140] Wang, Z., Sisman, B., Wei, H., Dong, X. L., and Ji, S. (2020). Cordel: A contrastive deep learning approach for entity linkage. In *ICDM*, pages 1322–1327. IEEE.

[141] Wiegreffe, S. and Pinter, Y. (2019). Attention is not not explanation. In *EMNLP/IJCNLP (1)*, pages 11–20. Association for Computational Linguistics.

[142] Winkler, W. (1990). String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. *Proceedings of the Section on Survey Research Methods*.

[143] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *EMNLP (Demos)*, pages 38–45. Association for Computational Linguistics.

[144] Wu, Z., Chen, Y., Kao, B., and Liu, Q. (2020). Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *ACL*, pages 4166–4176. Association for Computational Linguistics.

[145] Zafar, M. B., Valera, I., Gomez-Rodriguez, M., and Gummadi, K. P. (2017a). Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *WWW*, pages 1171–1180. ACM.

[146] Zafar, M. B., Valera, I., Gomez-Rodriguez, M., and Gummadi, K. P. (2017b). Fairness constraints: Mechanisms for fair classification. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pages 962–970. PMLR.

[147] Zhang, Z., Rudra, K., and Anand, A. (2021). Explain and predict, and then predict again. In *WSDM*, pages 418–426. ACM.

[148] Zhao, C. and He, Y. (2019). Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *WWW*, pages 2413–2424. ACM.

[149] Zliobaite, I. (2015). A survey on measuring indirect discrimination in machine learning. *CoRR*, abs/1511.00148.

[150] Öncan, T. (2007). A survey of the generalized assignment problem and its applications. *INFOR: Information Systems and Operational Research*, 45(3):123–141.