An Intrinsically Interpretable Entity Matching System

Andrea Baraldi
University of Modena and Reggio Emilia
Modena, Italy
andrea.baraldi96@unimore.it

Francesco Del Buono
University of Modena and Reggio Emilia
Modena, Italy
francesco.delbuono@unimore.it

Francesco Guerra
University of Modena and Reggio Emilia
Modena, Italy
francesco.guerra@unimore.it

Matteo Paganelli
University of Modena and Reggio Emilia
Modena, Italy
matteo.paganelli@unimore.it

Maurizio Vincini
University of Modena and Reggio Emilia
Modena, Italy
maurizio.vincini@unimore.it

ABSTRACT

Explainable classification systems generate predictions along with a weight for each term in the input record measuring its contribution to the prediction. In the entity matching (EM) scenario, inputs are pairs of entity descriptions and the resulting explanations can be difficult to understand for the users. They can be very long and assign different impacts to similar terms located in different descriptions. To address these issues, we introduce the concept of decision units, i.e., basic information units formed either by pairs of (similar) terms, each one belonging to a different entity description, or unique terms, existing in one of the descriptions only. Decision units form a new feature space, able to represent, in a compact and meaningful way, pairs of entity descriptions. An explainable model trained on such features generates effective explanations customized for EM datasets.

In this paper, we propose this idea via a three-component architecture template, which consists of a decision unit generator, a decision unit scorer, and an explainable matcher. Then, we introduce WYM (Why do You Match?), an implementation of the architecture oriented to textual EM databases. The experiments show that our approach has accuracy comparable to other state-of-the-art Deep Learning based EM models, but, differently from them, its predictions are highly interpretable.

1 INTRODUCTION

Understanding if entries in a dataset refer to the same real-world entity (i.e., entity matching – EM) is a challenging task even for human experts. 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.

Local explainability techniques have been introduced for allowing humans to understand the causes determining the model predictions for dataset records given as input. Typically, explainability can be achieved through *intrinsic* and *post-hoc* techniques [22, 23]. Intrinsic interpretability is obtained by restricting the complexity, hence using self-explanatory learning models. Post-hoc interpretability is obtained by applying interpretable methods to the model of interest [24]. The main difference between these two groups lies in the trade-off between model accuracy and explanation fidelity. Inherently interpretable models could provide accurate and undistorted explanations but may sacrifice prediction performance to some extent [12].

Regardless of the technique, an explanation 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 [24]. 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 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[5, 33]. 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 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

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

Table 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

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 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 paper, 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 3.1): 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 the paper (see Section 4). In WYM, the entity descriptions are encoded with word embeddings generated through the BERT language model [11]. We addressed the assignment problem [40] concerning the generation of the decision units with a special implementation of the stable marriage (SM) algorithm [17] 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 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 the paper 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 paper is organized as follows. In Section 2 we describe the related work. Section 3 introduces the main features of the architecture template; Section 4 describes its implementation in WYM; Section 5 includes the experimental evaluation; Finally, in Section 6, we sketch out some conclusions and future work.

2 RELATED WORK

In this section, we review the state-of-the-art literature on Entity Matching by analyzing the main DL-based models proposed in the literature, and introducing the main approaches for their interpretation.

2.1 Deep Learning for EM

Deep Learning (DL) models can effectively address EM. Preliminary evidence of this fact was provided by *DeepER* [14] and *DeepMatcher* [25], pioneering DL-based systems for EM. Subsequently, significant advances in the EM domain were achieved thanks to the integration of transformer architectures [37] into EM systems. Some examples of such systems are [7, 21, 27]. In [7] the most recent transformer-based models are fine-tuned on the

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

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. [27] exploits the ability of transformer-based models to capture highly contextualized semantic relationships between entity tokens to implement EM adapter for AutoML systems. [21] proposes *DITTO*, the currently best performing EM model in the literature, which combines advanced encoding and augmentation techniques of EM data with the BERT architecture [11]. Another recent DL-based framework for EM is CorDEL [35], 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. We have adopted a similar concept for the generation of decision units by distinguishing between paired and unpaired ones. The scores assigned to the units provide evidence of matching and non-matching entities. Despite the performance of these approaches, their main limitation is the poor interpretability, i.e. it is hard to understand the rationality they employ in generating predictions [9].

2.2 EM interpretability

The interpretation of EM models is approached in two ways: 1) by exploiting post-hoc analysis [39] or 2) by designing intrinsically explainable systems. *LIME* [29], *SHAP* [16], *ExplainER* [13], *Mojito* [10], *LEMON* [5], *Landmark Explanation* [4], and CERTA [31] are approaches that belong to the first category of systems and can be employed for the explanation of any black-box model. Even if most of the approaches (e.g., LIME, SHAP, ExplainER, Mojito and Landmark) compute feature-based explanations, there is no consensus about the best way to explain EM results [33]. Only a few approaches provide explanations with different, larger granularity. 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. To the best of our knowledge, no other approach exploits the concept of decision unit.

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 [38]. 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]. This represents a very controversial aspect especially in modern DL architectures, for which the presence of some form of attention mechanism [2] 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 [30, 34]. 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 [19, 26, 30].

In this paper, we explore the design of interpretable EM models that are also capable of achieving good performance. Following this direction, we have drawn inspiration from the paradigm explain-then-predict [38] and adapted it for the EM domain. According to this paradigm, a set of features is first extracted from the input and then the prediction is created. In this way, there is a direct attribution between the choices of the system and the reasons that determined this output. WYM is compliant with this paradigm since it first extracts match and non-match evidence from the entity descriptions and then builds the prediction.

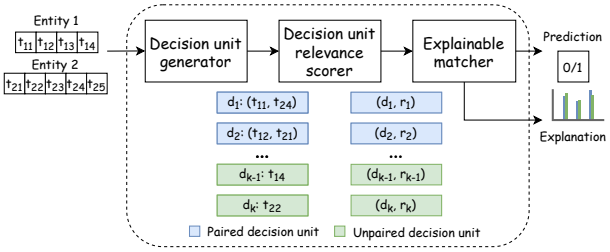


Figure 1: EM Architecture Template

3 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 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.

3.1 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 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.

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

$$d = \begin{cases} (t_i, t_j), & t_i \in e_x, t_j \in e_y \text{ paired unit} \\ t_i \vee t_j, & t_i \in e_x, t_j \in e_y \text{ unpaired unit} \end{cases} \quad (1)$$

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 [18], 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.

3.1.2 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 5.1.3).

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 $(t_x, t_y) \in \mathcal{M}_r$, $rs((t_x, t_y)) = rs((t_y, t_x))$. 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: $R_r = \{(d, rs(d)), \forall d \in D_r\}$.

3.1.3 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 $I_r = F_r W$.

3.2 Challenges

We list some challenges that an implementation of the architecture template has to address. In Section 4, 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.

3.3 Running example

Figures 3c and 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 1. Figure 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 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.

4 THE WYM EXPLAINABLE MATCHER

WYM is an implementation of the architecture template for textual databases. The main components are represented in Figure 2 and described in Section 4.1 (the Decision unit generator), Section 4.2 (the Relevance scorer), and Section 4.3 (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.

4.1 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.

4.1.1 Text tokenization and encoding. We apply the BERT language model [11] 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 5.1.3, 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 [28], obtained by a modification of the pre-trained BERT network that uses siamese and triplet network structures.

4.1.2 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 3.1), 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 θ, η, ϵ to evaluate the similarity of the token embeddings in the search spaces. It returns the paired $\mathcal{M}$ and unpaired $\mathcal{N}$ 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 $\mathcal{M}$. 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] [18]. 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

[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.

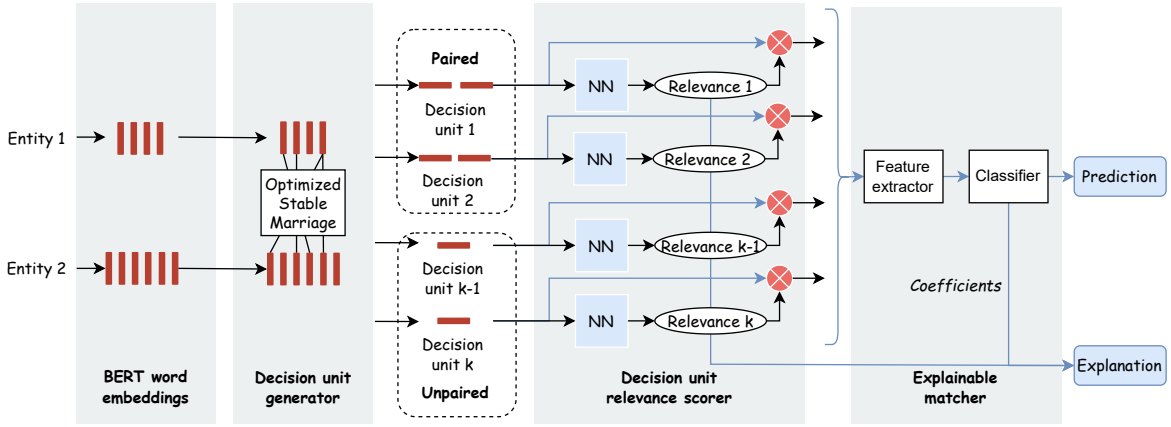


Figure 2: The Functional Architecture Template implemented in WYM.

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

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

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 3.2, since the encodings rely on the contextualized knowledge provided by the other decision units in the entity descriptions.

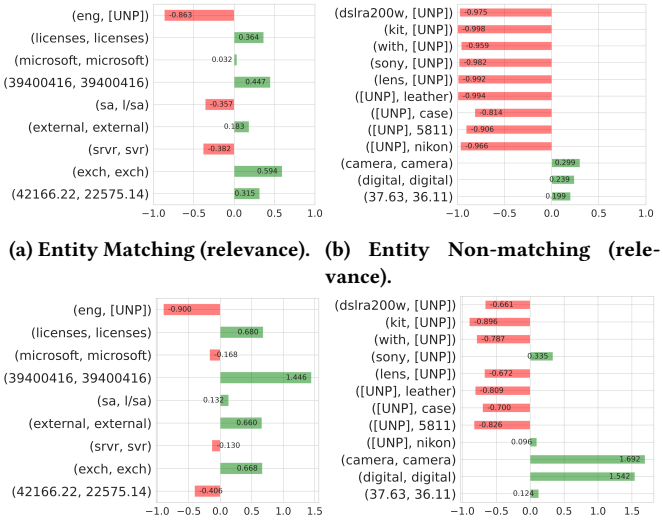
4.2 Decision unit relevance scorer

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 3.1. 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 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.

$$\hat{y}(l,r,y) = \begin{cases} 0 & \text{if y = 1} \ \wedge sim(l,r) < \alpha \\ 1 & \text{if y = 1} \ \wedge sim(l,r) >= \alpha \\ -1 & \text{if y = 0} \ \wedge sim(l,r) < \beta \\ 0 & \text{if y = 0} \ \wedge sim(l,r) >= \beta \end{cases} \quad (2)$$

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 3, where $|E_{lr}|$ is the number of entities in the dataset containing the decision units.

$$y^*_{lr} = \frac{1}{|E_{lr}|} \sum \hat{y}(l,r,y) \quad (3)$$



(a) Entity Matching (relevance). **(b) Entity Non-matching (relevance).**

(c) Entity Matching (impact). **(d) Entity Non-matching (impact).**

Figure 3: Explanations with relevance and impact scores

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 $3 \cdot 10^{-5}$.

4.2.1 Running Example. Figures 3a and 3b show the relevance scores computed by WYM on the running example in Section 3.3. Figure 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 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 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.

4.3 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.

4.3.1 Running Example. Figures 3c and 3d show the actual impact scores computed by WYM on the examples in Section 3.3. Figure 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 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 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 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 EXPERIMENTS

The experimental evaluation aims at answering four main questions: 1) How effective is WYM in solving EM tasks (Section 5.1); 2) If the impact scores provide a reliable interpretation of the EM predictions (Section 5.2); 3) If the time required to train the system and to compute the explanations makes it usable in real-world scenarios (Section 5.3); 4) If the decision-based explanations are effective for the users (Section 5.4). 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.

Table 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

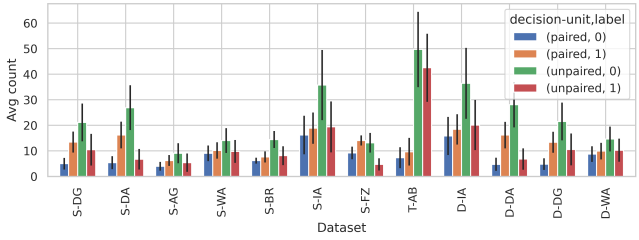


Figure 4: Average distribution of the decision units in the datasets.

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 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 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 $\epsilon = 0.7$.

5.1 Effectiveness of the EM Model

The effectiveness of WYM is evaluated according to three main perspectives: in Section 5.1.1, the performance is compared with competing systems; in Section 5.1.2, we evaluate how WYM behaves by varying the size of training sets, and in Section 5.1.3 we introduce a study of the components of the WYM architecture, evaluating different settings and implementation choices.

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

5.1.1 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] [27], an approach that provides the automatic application of ML models to the EM problem by pipelining AutoML systems with transformer-based encoders, CorDEL [35] and *DITTO* [21], 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 3.

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 ± 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 [12], 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.

5.1.2 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 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:

[5]DM+ is the combination of experiments / implementations as defined in [21]
[6]We average the results related to the best configuration (Hybrid-EM-Adapter) for AutoSklearn, AutoGluon and H20AutoML

Table 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)**				

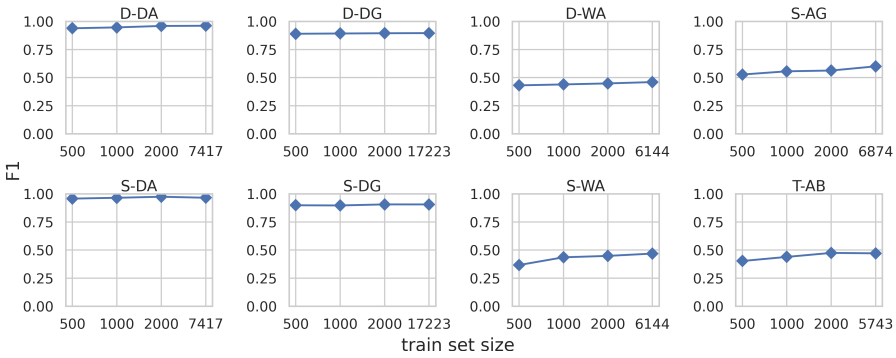


Figure 5: Learning Curves. The training set size is shown in the x-axes, the accuracy of the model (F1 score) in the y-axes.

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.

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.

5.1.3 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 [28]. 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 [36], 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 4.

Table 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)

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 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

Table 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	–	–

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 [32]), 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 [36] 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 4.

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 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 (the last two rows and columns represent the average and the standard deviations on the datasets and classifier, respectively).

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.

[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 [6] and not so simple as the edit distance.

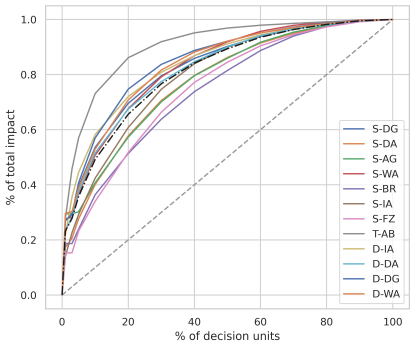


Figure 6: Conciseness of the explanations.

5.2 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 5.2.1), the faithfulless (Section 5.2.2), the contribution (Section 5.2.3) of the explanations and by comparing the WYM impacts with the explanations generated by a different tool (the Landmark EM explanation system – Section 5.2.4).

5.2.1 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 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.

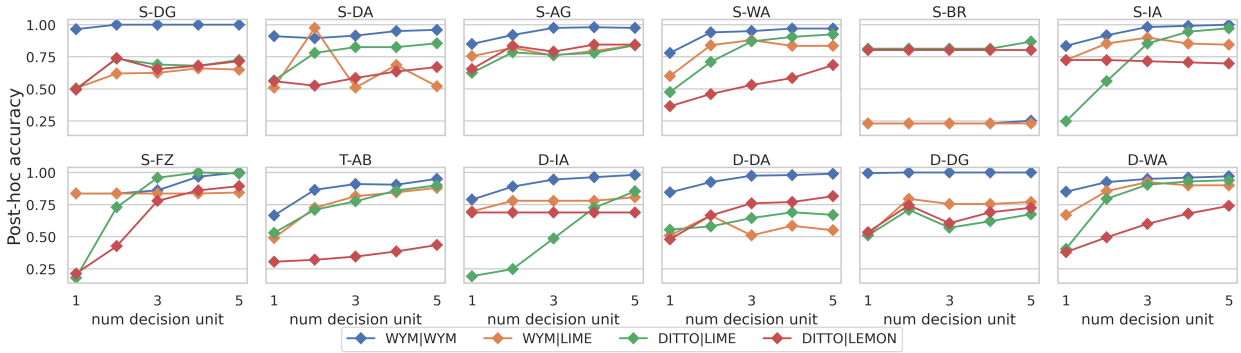


Figure 7: Sufficiency evaluated with the post-hoc accuracy. To high values correspond accurate explanations.

5.2.2 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 [1, 12, 15, 20]. We adopt the post-hoc accuracy [8] 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,

$$post - hoc - acc(v) = \frac{1}{M}\sum_{m=1}^{M} 1[y_v^{(m)} = y^{(m)}] \quad (4)$$

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 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.

5.2.3 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 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 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.

5.2.4 Correlation with Landmark. The WYM explanations are compared with the ones generated by *Landmark Explanation*[3, 4], 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 9 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.

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

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

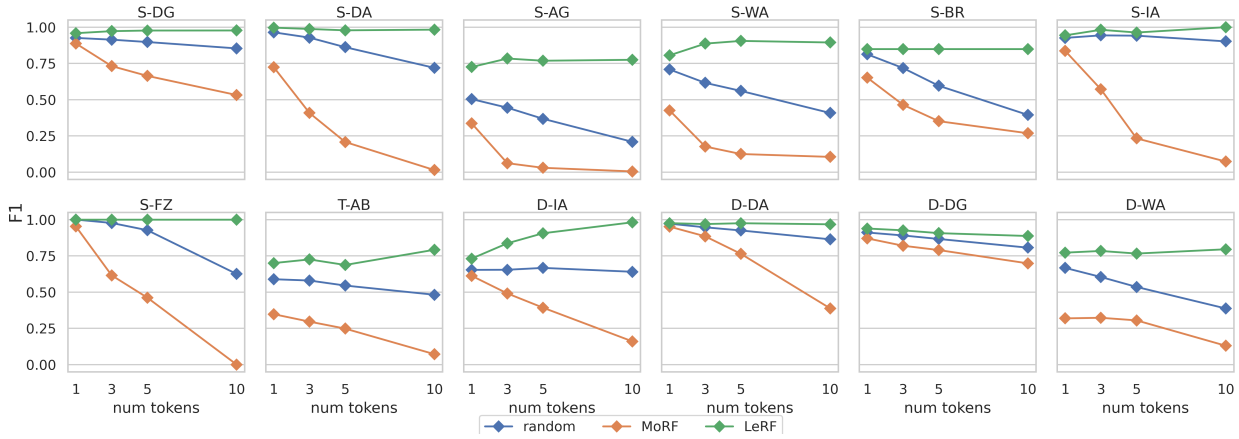
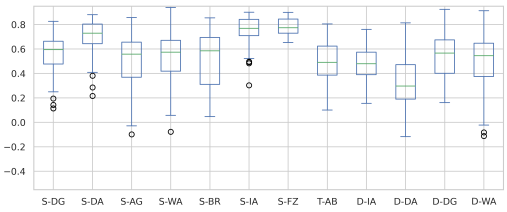
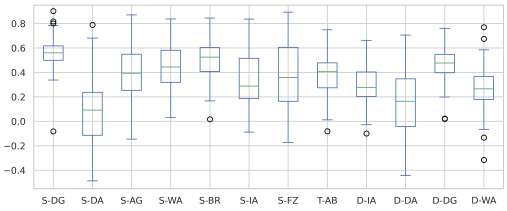


Figure 8: F1 score obtained in EM Models after the removal of the most relevant decision units (MoRF), less relevant decision units (LeRF) and random units.



(a) Impact score match records



(b) Impact score non-match records

Figure 9: Pearson correlation between the explanations generated by WYM and Landmark.

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 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).

5.4 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.

6 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.

REFERENCES

[1] Alejandro Barredo Arrieta, Natalia Díaz Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* 58 (2020), 82–115.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[3] Andrea Baraldi, Francesco Del Buono, Matteo Paganelli, and Francesco Guerra. 2021. Landmark Explanation: An Explainer for Entity Matching Models. In *CIKM*. ACM, 4680–4684.

[4] Andrea Baraldi, Francesco Del Buono, Matteo Paganelli, and Francesco Guerra. 2021. Using Landmarks for Explaining Entity Matching Models. https://edbt2021proceedings.github.io/docs/p259.pdf. In *EDBT*.

[5] Nils Barlaug. 2022. LEMON: Explainable Entity Matching. *IEEE Transactions on Knowledge and Data Engineering* (2022), 1–16. https://doi.org/10.1109/TKDE.2022.3200644

[6] Mikhail Bilenko, Raymond J. Mooney, William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. Adaptive Name Matching in Information Integration. *IEEE Intell. Syst.* 18, 5 (2003), 16–23.

[7] Ursin Brunner and Kurt Stockinger. 2020. Entity Matching with Transformer Architectures - A Step Forward in Data Integration. In *EDBT*. OpenProceedings.org, 463–473.

[8] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. 2018. Learning to Explain: An Information-Theoretic Perspective on Model Interpretation. In *ICML (Proceedings of Machine Learning Research)*, Vol. 80. PMLR, 882–891.

[9] Zhaoqiang Chen, Qun Chen, Boyi Hou, Zhanhuai Li, and Guoliang Li. 2020. Towards Interpretable and Learnable Risk Analysis for Entity Resolution. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1165–1180.

[10] Vincenzo Di Cicco, Donatella Firmani, Nick Koudas, Paolo Merialdo, and Divesh Srivastava. 2019. Interpreting deep learning models for entity resolution: an experience report using LIME. In *aiDM@SIGMOD*. ACM, 8:1–8:4.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.

[12] Mengnan Du, Ninghao Liu, and Xia Hu. 2020. Techniques for interpretable machine learning. *Commun. ACM* 63, 1 (2020), 68–77.

[13] Amr Ebaid, Saravanan Thirumuruganathan, Walid G Aref, Ahmed Elmagarmid, and Mourad Ouzzani. 2019. Explainer: Entity resolution explanations. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2000–2003.

[14] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *Proc. VLDB Endow.* 11, 11 (2018), 1454–1467.

[15] Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan L. Boyd-Graber. 2018. Pathologies of Neural Models Make Interpretation Difficult. In *EMNLP*. Association for Computational Linguistics, 3719–3728.

[16] Amirata Ghorbani and James Y. Zou. 2019. Data Shapley: Equitable Valuation of Data for Machine Learning. In *ICML (Proceedings of Machine Learning Research)*, Vol. 97. PMLR, 2242–2251.

[17] Marcel Gladbach, Ziad Sehili, Thomas Kudrass, Peter Christen, and Erhard Rahm. 2018. Distributed Privacy-Preserving Record Linkage Using Pivot-Based Filter Techniques. In *ICDE Workshops*. IEEE Computer Society, 33–38.

[18] Kazuo Iwama and Shuichi Miyazaki. 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)*. 131–136. https://doi.org/10.1109/ICKS.2008.7

[19] Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. *CoRR* abs/1902.10186 (2019).

[20] Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding Neural Networks through Representation Erasure. *CoRR* abs/1612.08220 (2016).

[21] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.* 14, 1 (2020), 50–60.

[22] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. 2021. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy* 23, 1 (2021), 18.

[23] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.* 267 (2019), 1–38.

[24] C. Molnar. 2018. Interpretable Machine Learning. https://christophm.github.io/interpretable-ml-book.

[25] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *SIGMOD Conference*. ACM, 19–34.

[26] Matteo Paganelli, Francesco Del Buono, Andrea Baraldi, and Francesco Guerra. 2022. Analyzing How BERT Performs Entity Matching. *Proc. VLDB Endow.* 15, 8 (2022), 1726–1738.

[27] Matteo Paganelli, Francesco Del Buono, Marco Pevarello, Francesco Guerra, and Maurizio Vincini. 2021. Automated Machine Learning for Entity Matching Tasks. In *EDBT*. OpenProceedings.org, 325–330.

[28] Nils Reimers and Iryna Gurevych. 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)*. Association for Computational Linguistics, Hong Kong, China, 3982–3992. https://doi.org/10.18653/v1/D19-1410

[29] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[30] Sofia Serrano and Noah A. Smith. 2019. Is Attention Interpretable? *CoRR* abs/1906.03731 (2019).

[31] Tommaso Teofili, Donatella Firmani, Nick Koudas, Vincenzo Martello, Paolo Merialdo, and Divesh Srivastava. 2022. Effective Explanations for Entity Resolution Models. In *ICDE*. IEEE, 2709–2721.

[32] Kai Sheng Teong, Lay-Ki Soon, and Tin Tin Su. 2020. Schema-Agnostic Entity Matching using Pre-trained Language Models. In *CIKM*. ACM, 2241–2244.

[33] Saravanan Thirumuruganathan, Mourad Ouzzani, and Nan Tang. 2019. Explaining Entity Resolution Predictions: Where are we and What needs to be done?. In *HILDA@SIGMOD*. ACM, 10:1–10:6.

[34] Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, and Manaal Faruqui. 2019. Attention Interpretability Across NLP Tasks. *CoRR* abs/1909.11218 (2019).

[35] Zhengyang Wang, Bunyamin Sisman, Hao Wei, Xin Luna Dong, and Shuiwang Ji. 2020. CorDEL: A Contrastive Deep Learning Approach for Entity Linkage. In *ICDM*. IEEE, 1322–1327.

[36] William Winkler. 1990. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research Methods* (01 1990).

[37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP (Demos)*. Association for Computational Linguistics, 38–45.

[38] Zijian Zhang, Koustav Rudra, and Avishek Anand. 2021. Explain and Predict, and then Predict Again. In *WSDM*. ACM, 418–426.

[39] Zijian Zhang, Jaspreet Singh, Ujwal Gadiraju, and Avishek Anand. 2019. Dissonance Between Human and Machine Understanding. *Proc. ACM Hum. Comput. Interact.* 3, CSCW (2019), 56:1–56:23.

[40] Temel Öncan. 2007. A Survey of the Generalized Assignment Problem and Its Applications. *INFOR: Information Systems and Operational Research* 45, 3 (2007), 123–141. https://doi.org/10.3138/infor.45.3.123 arXiv:https://doi.org/10.3138/infor.45.3.123