# Hybrid Approach to Instance Matching

**Abstract** The proliferation of web data and the advent of extensive knowledge graphs have led to the creation of vast volumes of disconnected data, resulting in data silos. Integrating and sharing web data across diverse domains require that equivalent instances across different sources are correctly identified. Instance matching, often referred to as Entity Resolution, encompasses the task of determining whether two instances correspond to same resource or entity. This process poses significant challenges, particularly in distinguishing between identical entities and those with similar attributes. Candidate generation has a pivotal role in facilitating appropriate comparisons between entities across disparate datasets. This paper employs an inverted index based approach to identify candidates for the matching task and a query likelihood model based selection to further reduce the candidate set. This paper proposes a novel system architecture employing hybrid ensemble classifiers and a methodology for identifying equivalent instances despite the challenges posed by diverse data representations in instance matching. Through experimental evaluation on real-world datasets, we demonstrate that our hybrid ensemble learning approach consistently outperforms standalone matchers in terms of accuracy and F1score. A comprehensive literature review on instance matching, discussing practical considerations and challenges for data interlinking is also presented here. Future research directions aimed at contributing to seamless data integration and knowledge sharing across disparate domains are also outlined.
**Keywords:** Instance matching, Semantic Web, Entity Resolution, Candidate Generation, Ensemble Classification.

## 1. INTRODUCTION

The World Wide Web is witnessing an information explosion due to technological advancement such as the Internet of Things (IoT), Social media platforms and Augmented Reality(AR) [1] [2]. The rapid expansion of data across the World Wide Web has resulted in the widespread emergence of isolated data silos.[3]. These silos inhibit the sharing, integration, and utilization of data across systems and applications, resulting in duplication of efforts, inconsistencies, and missed opportunities for collaboration and innovation. In view of these issues, there is a need to organize and coordinate data exchange for intelligent IoT applications, Social Network analysis, Automated Control systems etc. This need can be addressed through data interlinking, which offers numerous opportunities for organizations across various domains. By linking related data from disparate sources, data interlinking facilitates seamless integration and enables innovative applications to leverage interconnected datasets.

Semantic web technologies[4] provide the foundational framework and tools necessary for accurate and efficient instance matching, enabling the integration and interoperability of data across diverse domains and sources. RDF (Resource Description Framework) provides a standardized model for describing resources using triples, which consist of subject-predicate-object statements. RDF enables the creation of machine-readable metadata about web resources, facilitating data interchange and integration. Ontology web language(OWL) define concepts and relationships within a domain. OWL builds upon RDF and provides a richer vocabulary for describing complex relationships and constraints, enabling instance matching algorithms to compare attributes and relationships between instances accurately [5]. Knowledge graphs play a vital role in the representation of complex objects enriched with semantic annotations[6]. The use of ontologies and vocabularies ensure a shared understanding of the semantics of instances, leading to more precise and context-aware instance matching. Leveraging linked data principles, knowledge graphs integrate and interlink datasets from multiple sources, providing a global view of data and facilitating cross-referencing and alignment between related instances. Data interlinking is thus essential for effective data integration and knowledge discovery across various domains, thereby driving transformative benefits and fostering innovation through intelligent applications.

Open Data [7] [8] published by Governments and Open Data Community can be interlinked for the creation of innovative applications for data enrichment and analytics, crime prediction, early warning systems or business processes improvement . In the Healthcare sector, data linking allows healthcare providers to create comprehensive patient records by linking information from electronic health records (EHRs), medical imaging and insurance claims. In the Financial sector, linking data from various financial transactions helps to identify suspicious activities, detect anomalies, and ensure regulatory compliance. Knowledge based Systems(KBS) extensively rely on knowledge bases that contain facts, rules, heuristics, procedures, relationships and best practices relevant to the domain they operate in. By linking knowledge base with external sources, such as

databases, APIs, or external knowledge repositories. KBS can enrich their knowledge through linking which provide an insight in decision support, contextual understanding and knowledge discovery [9], [10].

There are several challenges to interlinking data from Open Data Community projects, Knowledge bases or conventional data sources notably data heterogeneity, scalability, and instance matching. Data heterogeneity arises as data may come from various sources with different formats, schemas, and quality standards. Matching instances across heterogeneous data can be challenging due to inconsistencies and discrepancies. Semantic heterogeneity where Instances may represent the same entity but use different terminologies, abbreviations or representations need to be addressed. Resolving semantic heterogeneity requires understanding the underlying semantics and relationships between data elements. Another concern is Scalability as matching large datasets with millions or billions of records can be computationally intensive and time-consuming. Efficient algorithms and techniques are required to handle scalability issues. Instance matching, also referred to as Entity Resolution, entails the process of ascertaining whether two instances pertain to the same resource in real world(e.g., person, product) which involves resolving ambiguities. Differentiating between identical entities and distinct entities with similar attributes is a key challenge [11], [12]. Several software tools and frameworks have emerged to handle the issue of identifying semantically equivalent objects across different data sources. The underlying principle behind these methodologies is to automatically identify pairs of resources from two sets, denoted as A and B, that should be linked together, often based on a predefined semantic relationship such as $< owl : sameAs >$.

This work focus on the aspect of achieving an effective and efficiency approach to instance matching. Effectiveness is related to identifying all relevant links between the sources without generating incorrect links. It is necessary that a system learn the similarity conditions that two resources, denoted as $a \in A$ and $b \in B$, must satisfy to be considered as related or equivalent. Instance matching as a computational task need to compare all elements of $A$ with all elements of $B$ would have a computational complexity of $O(|A||B|)$, which can be very high for large datasets. Hence, efficient algorithms and optimization techniques, are essential to ensure scalability and practical applicability in real-world scenarios.

The key contribution of our work can be outlined as follows. First, an comprehensive review of state-of-art is presented that highlights the research directions in the literature. Second, we propose a system architecture that employs hybrid classifier models for instance matching. Third, we present a candidate generation and selection approach along with concept clustering for improving the efficiency of instance matching

The outline of the paper is as follows. A comprehensive review of literature is given Section 2 . Section 3 explains the system architecture. The experimental evaluation and dataset used is given in Section 4. Discussions and results pertaining to experimentation are in section 5 followed by conclusions in Section 6.

## 2. Literature Review

Instance matching is essential for integrating data from diverse sources and plays a crucial role in data integration, ontology alignment, semantic web, and knowledge engineering applications. This literature review provides a comprehensive overview of instance matching approaches namely similarity based approaches, machine learning based approach, probabilistic matching models and graph based matching models. This work also highlights the key concepts, methodologies, strengths, and limitations of these approaches .

Similarity-based approaches[10] in instance matching focus on computation of similarity score of instances based on attributes values. Similarity measures like Levenstein Distance offer character-level comparisons, while set-based measures such as Jaccard Similarity and Cosine Similarity provide robust solutions for string and document matching tasks [11]. WordNet-based metrics leverage semantic knowledge to identify similar entities based on their semantic meanings. [13]incorporates schema based and instance based matching within a two-level matching framework, introducing Multivariate Statistical Matching to automate schema scoring and leverage domain knowledge and attribute correlations . Similarity based approaches are effective in capturing similarity of numeric, categorical, or textual attributes and are adaptable to different domains and matching tasks. However, limitations are the challenge of defining appropriate similarity measures for diverse data types and the need for manual parameter tuning. Additionally, these approaches may struggle with handling noise, inconsistencies, and variations in data, especially in cases where attributes are poorly defined or have high dimensionality.

Machine learning-based instance matching approaches [14] have gained significant prominence due to their capability to automatically learn patterns and relationships from data, enabling efficient matching across heterogeneous sources. These approaches typically involve training machine learning models on labeled or unlabeled data, where instances are represented as feature vectors. In [10] Instance matching is viewed as a binary classification task which uses a similarity metric that is schema independent based on different descriptive algorithms. The instance-level coreference resolution is a conspicuous problem faced by semantic web community. Discovering the coreference resolution links is often difficult due to the heterogeneity that exists at the schema level. [15] describes schema mapping by leveraging background knowledge which subsequently can help the instance coreference resolution process.

However this approach was found suitable to small subgraphs of the Linked Data cloud. Strengths of machine learning-based approaches include their adaptability to different data types and domains, their capability to handle complex patterns and relationships, and their potential for scalability and automation. The ability in capturing non-linear relationships and dependencies between instances, make them suitable for matching tasks with high-dimensional or noisy data. However, a limitation lies in the necessity for a substantial amount of labeled training data. The acquisition of the training data could prove costly and time-intensive. Moreover, their performance is largely depends upon the quality and inclusiveness of training data.

Graph matching approaches[16] to data interlinking not only assess the similarity of instances but also consider their neighborhood structure. These approaches employ graph matching algorithms to compute graph similarity by evaluating all possible pairs of nodes from two datasets, generating mappings if the similarity exceeds a specified threshold [17]. However, these algorithms often lack heuristics to handle situations where ontologies differ. Alternatively, reconciliation algorithms leverage context information as evidence to conclude whether entities refer to the same entity [18]. Such algorithms propagate context similarity using a dependency graph, where nodes represent similarities between pairs of attribute values, and edges indicate the strength of reconciliation decisions. Additionally, there have been attempts to exploit Wikipedia as a background corpus and utilize its concepts for entity disambiguation. A method that combines deep learning and knowledge base to automate entity attribute extraction and entity Linking have been suggested in [19] . Another approach demonstrates how to calculate semantic distance on Linked Data to identify relatedness between resources using Linked Data semantic distance algorithms [2]. The instance matching task is often transformed as a document matching task and solves it using a vector space embedding methods. The method creates a virtual document for every instance and utilizes word embeddings to evaluate word similarities, considering both lexical and semantic levels. The technique leverages pre-trained word embeddings for capturing semantic relatedness of words, including synonym and terminological variants. By representing instances as virtual documents and using lexical semantic similarity techniques, the approach enables the alignment of instances in knowledge bases[20].

Probabilistic approach in 'IdMesh'[21] builds a disambiguation graph using the transitive relation of equivalence links for graphs that contain uncertainty in information. It falls back on the properties of symmetry and transitivity that is displayed by equivalent relations for defining constraints. The constraint satisfaction factor-graphs computes the posterior probabilities of link variables and helps in detecting inconsistencies within graph of entities.The probabilistic framework[22] enhances the accuracy and robustness of entity disambiguation in uncertain data environments. .

Several hybrid approaches have been used that combines the above mentioned approaches. *ZenCrowd*[23] uses crowd sourcing and probabilistic graph model to semi-automatically match data sets. It strives for reconciliation of entities present in Web pages to linked web of data. The system *ZenCrowd* employs entity extractor to identify entities from html web pages which is passed to algorithmic linkers that integrates them with the linked open data cloud. The blocking technique firstly builds an inverted index and generates likely candidates using a probabilistic approach. In second stage, system analyses the candidate matches and refine them using graph based instance matching techniques. In the third stage, it involves human computation by providing crowdsourcing tasks to improve results. Finally the hybrid approach integrates results using a probabilistic inference in order to make decision regarding the linking of entities.

Ontology matching and alignment[24]is often carried out prior to instance matching as a resource may belong to different classes and may be described using different predicates and may be part of different ontologies. For instance, a name within one dataset may be identified using the *'foaf:name'* data property within the FOAF ontology, while in another dataset, it might be denoted using the *'vcard:N'* object property from the VCard ontology. Ontology matching allows for finding correspondences between different ontologies. [5] uses a context-independent approach that addresses structural heterogeneity by considering the components of ontology model, its properties and relationships. By considering the structural heterogeneity, the context-independent approach can handle variations in the representation and organization of data across different datasets, ensuring accurate alignment of resources.
Candidate generation and selection that focuses on the issue of scalability need to be taken care of by instance matching systems[25]. A method to reduce the candidate pairs is to keep track of match histories of instances and identifying instances that are dissimilar.The similarity between histories is found using jaccard similarity. Thresholding using a sigmoid function to decide the commonality between instances on-the-fly. Another method carry out candidate selection by generating n-grams from discriminating feature and use them to recursively find candidate keys that help to discriminate instances in a given dataset using any of the learned predicates. Sorted Neighborhood technique partitions instances into blocks based on sorted order and employs these blocks to efficiently identify candidate pairs It also attempts to improve efficiency of look up by constructing an inverted index of Instances[26]. Hasan et al. introduced a framework for candidate generation using locality-sensitive hashing (LSH) to approximate the similarity between entities in large-scale datasets [27]

This research focuses on interlinking hetrogeneous datasets by following a hybrid approach.This work adopts the use of an inverted index for improving efficiency of

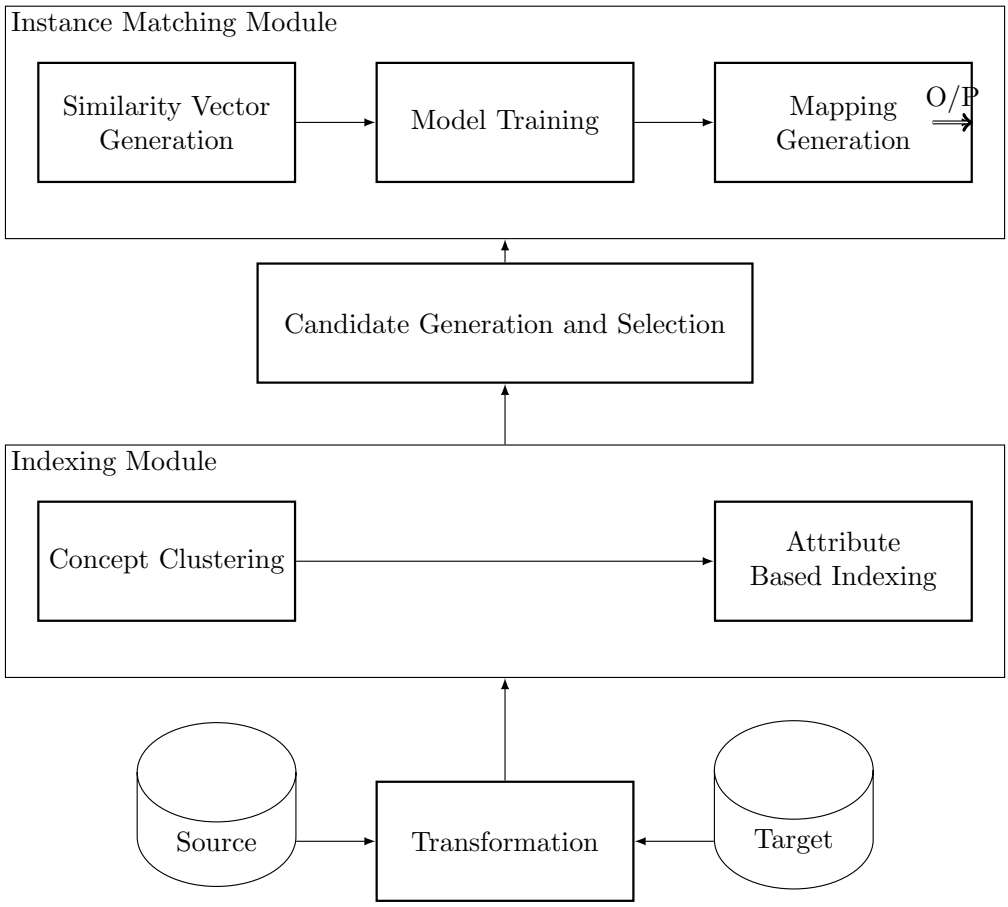



Figure 1. Proposed architecture of data interlinking system

candidate generation as in [24]. However this work makes use of an effective candidate generation and selection approach that reduces the number of candidates. The approach employs a hybrid similarity measure that combines vector space model and various edit distance measures for calculating a similarity score based on which instances can be matched. Overcoming semantic heterogeneity can be typically achieved by matching entities to find a set of correspondences or to decide on alignment according to application needs. Unlike in [28] where a threshold is used to identify candidate pair as match/mismatch, we use unsupervised learning to identify true matches

### A. *Problem Definition*

**Instance Matching**: Given a source dataset $A = \{a_1, a_2, a_3, ..., a_n\}$ and a target dataset $B = \{b_1, b_2, b_3, ..., b_m\}$ that consist of collection of instances, Instance matching task generates mapping $M = \{(a_i, b_j)\}$ between source and target datasets containing only matching instances where $a_i \equiv b_j$ that represent the same world entity.

## 3. Proposed System

The main concern of Data Interlinking is to find out equivalent instances from disparate data sources and generate mapping. The proposed architecture for data interlinking is given in Fig.1. The approach to find mappings involves four phases: ( i ) Transformation (ii) Indexing/blocking ( iii ) Candidate generation and selection and ( iv) Instance matching.

### A. *The transformation phase*

The need for transformation of data [29]prior to instance matching arises due to heterogeneity of data sources, which may use different formats, structures, and representations to describe instances and relationships. Transformation phase includes several data processing tasks for cleaning and standardization of data. transformation phase includes . The data format of the source and target dataset may exist in the form of XML/RDF, OWL, CSV file etc. The transformation steps that involved in instance matching are listed below.

- Handling Missing Values and Special Characters:

Data cleaning of literal values by handling missing values, removing special characters, stemming and normalizing string representations are essential prior to indexing.

- Normalizing String Representations:Normalization of literal values, such as *Brand,Category,Price, Weight, Date, location* converting all prices to a common currency format, standardizing brand names, converting weights and dimensions to a common unit of measurement, ensuring consistent date formats and naming conventions helps to tackle discrepancies.

- Handling Multilingual Data : If multilingual data is present, language tags such as *@en, @ar, @es, @fr* need to handled appropriately. RDF/XML data is converted into a format that can be easily manipulated by the instance matching algorithms using RDF parsers [30].

- RDF/XML Parsing and Blank Node Resolution: Blank nodes may represent anonymous resources or entities in DBpedia. Resolving blank nodes involves assigning temporary identifiers or treating them as anonymous entities to ensure consistency and effective data processing.

- Ontology Term Usage and Consistency in OWL Data: Entities and relationships are described using different ontologies and vocabularies to describe entities and relationships. Ensuring use of appropriate ontology terms and maintaining consistency of ontology terms across instances are vital when data is in OWL format.

### B. Blocking or Indexing

Instance Matching involves comparison of instances from source dataset with all instances from target dataset to identify potential matches.This leads to scalability issues[27] as the rise in number of instances will cause the number of pairwise comparisons to grow quadratically. Hence comparing every pair of instances from two datasets would be a computationally intensive task. We use concept based clustering and *Inverted Indexing* to deal with such large number of comparisons and efficient data processing strategy to address the scalability.

#### 1) Concept Clustering

Concept-based clustering organizes data into groups based on the underlying concepts or semantic similarities present in the data. Concept identification[31] in structured or semistructured datasets is done through pattern matching, Domain specific rules or through ontology matching.

- Pattern Matching[32] Identifying patterns in product names or descriptions to extract relevant concepts such as brand names, product models, or category keywords.

- Another way is to apply Domain-specific Rules[33] based on domain knowledge to identify specific product categories (e.g., smartphones, laptops) based on attributes like screen size or processor type.

- Ontology Mapping approach[33] maps attributes to concepts defined in product ontologies or taxonomies, aligning the "Category" attribute with standardized product categories defined in a product ontology. we map the "Category" attribute to standardized product categories defined in a product ontology (e.g., Mobile Phones, Tablets, Laptops).

Instances described using RDF are clustered with the help of property *RDF:type* used to indicate the class of a resource or through schema information obtained through *RDFS:class RDFS:subclassOf* tags. For e.g., *Authors* are described as instances of the *foaf:Person* class where as *Books* are described as instances of the *dcterms:BibliographicResource* class. Hence clustering helps to bring together instances belonging to the same class. Instances in a cluster from the source dataset need to be compared only with instances belonging to same class in target dataset.

#### 2) Indexing

The clustering is followed by construction of an inverted index.Blocking or indexing techniques helps to reduce the search space by creating index based on key attributes. This helps prune irrelevant comparisons and improve efficiency[34]. Here we construct an attribute based inverted index that can be queried to produce a reduced dataset called as Candidate set.

Inverted Index is constructed by selecting one or more most informative attribute. For each instance, terms corresponding to this attribute are inserted as $(key, value)$ pair in the index where *key* corresponds a term and *value* corresponds to an instance identifier. The inverted index is updated by adding an entry for the key if it doesn't exist, or appending the instance identifier to the values list associated with the key. Let $t$ be a term in the index, $i$ be an instance in the dataset, then Index $I$ is represented as below in (1).

$$I = \{[t_1 : i_1, i_2, ..],\ t_2 : [i_1, i_3, \ldots],\ \ldots\} \tag{1}$$

The inverted index is modified to include additional information such as term frequency, document frequency, or positions of terms within instances. Then, the weighted inverted index $I$ is represented as a mapping from terms $t$ to lists of instance-weight pairs as shown in (2).

$$I = t_1 : [(i_1, w_{t_1}), (i_2, w_{t_1}), ..],\ t_2 : [(i_1, w_{t_2}), (i_2, w_{t_2}), ...], .. \tag{2}$$

Each term $t$ .is associated with a list of instance-weight pairs, where each pair $(i, w_{t,i})$ represents the instance $i$ and the weight of term $t$ within that instance.

TABLE I. Dataset Statistics

| Dataset | Domain | Source Dataset instances | Source Dataset properties | Target Dataset instances | Target Dataset properties | Matches |
|---|---|---|---|---|---|---|
| D1 (Amazon-GP) | E-Commerce | 1363 | 5 | 3226 | 5 | 1300 |
| D2 (Abt-Buy) | E-Commerce | 1082 | 4 | 1092 | 5 | 1097 |

### C. Candidate generation and Selection

Candidate generation involves finding a subset of instances called as the candidate set. The key attribute acts as the query $Q = \{q_1, q_2, q_3...\}$ consisting of multiple words or terms. For each query term, the matching instances(E) from target dataset can be found by querying the Index. The query result contain the instance identifiers from target dataset that contain each query term. This may lead to a large candidate set and hence to prune candidates set, a probabilistic query likelihood model is used as a selection strategy. The equation (3) for the query likelihood model represents the probability of observing the query $Q$ given the language model $\theta_e$ [35] [36] .

$$\begin{aligned} Score(E, Q) &= p(Q|\theta_E) \\ p(q|\theta_E) &= \frac{C(q, E)}{|E|} \end{aligned} \tag{3}$$

The query likelihood model estimates the probability of term $q$ in instance $e$ using maximum likelihood estimation, where $C(q, E)$ is the frequency of $q$ in $E$ and $|E|$ is the number of instances (3).

$$p(q|\theta_E) = \frac{|E|}{|E| + \mu} p(q|E) + \frac{\mu}{|E| + \mu} p(q|C) \tag{4}$$

The equation (4) represents the smoothed probability of term $q$ in instances $E$ using Dirichlet prior smoothing, where $\mu$ is the smoothing parameter and $p(q|C)$ is the probability of $q$ in the collection [37].

$$\begin{aligned} \log\ p(Q|\theta_E)_{\text{rank}} = &\sum_{q \in Q \cap E} c(q, Q) \log\left(1 + \frac{c(q, E)}{\mu p(q|C)}\right) \\ &+ |Q| \log \frac{\mu}{|E| + \mu} \end{aligned} \tag{5}$$

The query likelihood score is calculated using (5) for instances in $E$ given query $Q$, where $c(q, Q)$ is the count of term $q$ in query $Q$ and $|Q|$ represents the length of the query. The scoring function enables selection of candidates for further processing.

### D. Similarity vector Generation

Candidate pairs are compared to quantify the similarity or dissimilarity between them. This comparison often relies on similarity metrics to determine closeness or relatedness of the instances . Let the candidate pair $(A, B)$ correspond to an instance of source and target set respectively. The feature vector $A = [a_1, a_2, ..., a_n]$ and $B = [b_1, b_2, ..., b_n]$ correspond to attributes values of the instances. Then, the similarity vector (6) of candidate pair$(A, B)$ can be expressed as

$$\mathcal{X} = [x_1, x_2, ..., x_n] \quad where \quad x_i \in (0, 1) \subset R \tag{6}$$

where $x_i = Sim(a_i, b_i)$ is computed using any similarity measure. The choice of similarity metric $Sim$ depends on several factors, the type of data, the nature of task at hand, and the characteristics of the features being compared. For textual features cosine similarity and TF-IDF measures are commonly used. Applying Term frequency Inverse document frequency ( TF-IDF) $sim(a_i, b_i)$ is expressed as the cosine similarity of the corresponding word vectors $\vec{a_i}$ and $\vec{b_i}$ as shown in (7).

$$Sim_{Cosine}(a_i, b_i) = cos(\vec{a_i}, \vec{b_i}) = \frac{\vec{a_i} \cdot \vec{b_i}}{|\vec{a_i}| \cdot |\vec{b_i}|} \tag{7}$$

Use of n-gram similarity measures allows for more flexible and forgiving comparisons, accommodating variations in text data while capturing similarities based on shared subsequences. This makes it suitable for features containing textual data having variations like typos, abbreviations, or minor alterations. For e.g. product names:$a$ = "$Samsung\ Galaxy\ S21\ Ultra$" and $b$ = "$Galaxy\ S21\ U$" may not be identified as similar in direct comparison. Transforming the data using n-grams(eg. 3-gram) {"Sam", "ams",..."S21"..."} and using Jaccard similarity can improve the similarity score.

Let $N_a$ and $N_b$ denote the sets of n-grams extracted from features $a$ and $b$ , $N_{intersection}$ be set of common n-grams between a and b and $N_{union}$ be set of all unique n-grams in $a$ and $b$ as shown in (8). Then Jaccard similarity is a measure of the proportion of n-grams shared between two sets divided by count of distinct elements in two sets.

$$\begin{aligned} N_{\text{intersection}} &= N_a \cap N_b \\ N_{\text{union}} &= N_a \cup N_b \\ \text{Sim}_{\text{Jaccard}}(a_i, b_i) &= \frac{|N_{\text{intersection}}|}{|N_{\text{union}}|} \end{aligned} \tag{8}$$

Traditional way to compute similarity of instances is to apply aggregation functions to get a combined similarity score and to apply threshold using which candidate pairs will be classified as *'matches'* or *'nonmatches'*. When applying a combined similarity score, it's essential to carefully design the combination scheme based on the nature of the data and requirements pertaining to the application.The candidate pairs that match the similarity criteria will selected and be mapped as equivalent instances. However this approach differ by applying supervised learning models to find matches among entities. We use techniques such as weighted averaging, ensemble methods of classifiers to effectively classify instance pairs. Additionally, validation and evaluation on relevant benchmarks and through cross-validation are we assess the performance of the models in classifying candidate pairs.

### *E. Instance Matching Model Training*

The proposed approach views Instance matching as a classification problem[38]. Given the collection of similarity vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ each similarity vector $\mathbf{x}_i$ is associated with a class a binary class label $y \in \{0, 1\}$, where $y = 1$ indicates that $\mathbf{x}_1$ and $\mathbf{x}_2$ refer to the same real-world entity, and $y = 0$ indicates that they refer to different entities. using this training set $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ...(\mathbf{x}_n, y_n)$ where N is the number of training instances, our goal is to learn a binary classification model $h(\mathbf{f}(\mathbf{x}))$ that maps the similarity vector to the binary class label y. Two hybrid ensemble models are trained with the aim of improving performance (i) BagBoost Classifier (ii) Voting classifier.

#### *1) **BagBoost Classifier***

The Bag-Boost ensemble model combines bagging and boosting techniques to create a powerful ensemble learner.Bagging (Bootstrap Aggregating) involves generating multiple bootstrap samples from the original training dataset. The working of BagBoost classification is as given below.

1) Let a training set $D = \{(x_i, y_i)\}$ where $(x_i, y_i)$ represents individual data points in the training set, where $x_i$ represents the features and $y_i$ represents the corresponding labels.
2) Bagging creates bootstrap samples $D_1^*, D_2^*, ..., D_B^*$ containing n samples. Each bootstrap sample is created by randomly sampling n examples with replacement from the original dataset.
3) Each Boosting classifier(AdaBoost) is trained on its corresponding bootstrap sample $D_i^*$. Let $h_i(x)$ denote the hypothesis function learned by the $i^th$ classifier where $h_i(x)$ is the aggregated prediction for input x, $\alpha_j$ is the weight assigned to the $j^th$ weak learner component of boosting classifier as shown in (9).

$$h(x) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j h_j(x)\right) \tag{9}$$

4) Once the boosting classifiers are trained, class labels predicted are aggregated using a voting scheme. The class label with the highest frequency of occurrence among the predictions is chosen as the final prediction.

Here the use of bagging classifier aims to reduce variance by training classifiers on different subsets of the training data, which helps to decrease the correlation between individual models.Through the use of Boosting Classifier focuses on iteratively training base learners to correct the errors made by previous models. In the context of Bag-Boost classification approach, boosting can be achieved through algorithms like AdaBoost(Adaptive Boosting) or Gradient Boosting. Mathematically, boosting involves assigning weights to training instances and adjusting these weights based on the performance of previous models. Given a weighted training set $D_t = \{(x_1, y_1, w_{1,t}), (x_2, y_2, w_{2,t}), ..., (x_n, y_n, w_{n,t})\}$ at iteration t, where $w_{i,t}$ represents the weight of instance i at iteration t, boosting computes the weighted error of each base learner and updates the weights of instances accordingly. The next base learner is trained on the updated weighted training set. The boosting process continues for T iterations, with each subsequent base learner focusing more on instances that were misclassified by previous models. The final Bag-Boost ensemble model combines the predictions of individual base learners through a weighted averaging or voting scheme. Mathematically, given base learners $h_1, h_2, ..., h_B$ , the ensemble prediction H(x) for a new instance x can be computed as in (10)

$$H(x) = \frac{1}{B} \sum_{i=1}^{B} h_i(x) \tag{10}$$

Here B represents the total number of base learners, $h_i(x)$ represents the prediction of the $i^{th}$ base learner for the input x. Alternatively, if the base learners produce probabilistic outputs, the ensemble prediction can be computed as the weighted average of the predicted probabilities. While this description provides an overview of the Bag-Boost ensemble model and its mathematical components, the specific implementation may vary depending on the choice of base learners, boosting algorithm, and combination method.

We train the classification model by minimizing the loss function over the training dataset D, using grid search optimization . The trained model $h(\mathbf{f}(\mathbf{x}))$ learns to predict the likelihood that a pair of instances refers to the equivalent real-world entity.

#### *2) Voting Classifier*

In a Voting Classifier, the aggregation of predictions is done by combining the predictions of multiple classifiers through a voting scheme. The working of this model is as given below

1) In soft voting, the predicted class probabilities from

TABLE II. Performance of Classifiers on Amazon-GoogleProducts dataset

| Classifier | Accuracy | F1-Score | Precision | Recall | Roc_Auc |
|---|---|---|---|---|---|
| DT | 0.779 | 0.781 | 0.773 | 0.788 | 0.779 |
| KNN | 0.809 | 0.821 | 0.774 | 0.874 | 0.870 |
| LR | 0.734 | 0.727 | 0.746 | 0.711 | 0.787 |
| MLP | 0.733 | 0.738 | 0.726 | 0.751 | 0.800 |
| NB | 0.672 | 0.595 | 0.774 | 0.484 | 0.776 |
| SVM | 0.732 | 0.725 | 0.745 | 0.705 | 0.789 |
| ADA | 0.739 | 0.738 | 0.740 | 0.736 | 0.797 |
| RF | 0.811 | 0.824 | 0.818 | 0.821 | 0.890 |
| H1(BB) | 0.815 | 0.816 | 0.817 | 0.815 | 0.898 |
| H2(Vot) | 0.831 | 0.834 | 0.817 | 0.852 | 0.905 |

TABLE III. Performance of Classifiers on Abt-Buy dataset

| Classifier | Accuracy | F1-Score | Precision | Recall | Roc_Auc |
|---|---|---|---|---|---|
| DT | 0.737 | 0.736 | 0.737 | 0.735 | 0.738 |
| KNN | 0.755 | 0.767 | 0.731 | 0.805 | 0.814 |
| LR | 0.656 | 0.650 | 0.662 | 0.639 | 0.727 |
| MLP | 0.658 | 0.668 | 0.650 | 0.692 | 0.742 |
| NB | 0.642 | 0.645 | 0.640 | 0.651 | 0.703 |
| SVM | 0.644 | 0.644 | 0.645 | 0.644 | 0.725 |
| ADA | 0.670 | 0.659 | 0.683 | 0.638 | 0.744 |
| RF | 0.790 | 0.795 | 0.779 | 0.812 | 0.870 |
| H1(BB) | 0.775 | 0.781 | 0.761 | 0.801 | 0.858 |
| H2(VOT) | 0.793 | 0.797 | 0.783 | 0.812 | 0.869 |

the base classifiers are averaged. The final prediction is determined by selecting the class having the highest average probability as in(11).

$$H(x) = \text{argmax}_c \frac{1}{M} \sum_{i=1}^{M} P_i(c|x) \tag{11}$$

where $P_i(c|x)$ is the probability assigned by the $i^{th}$ base classifier to the input x belonging to class c, and H(x) is the final aggregated prediction, selecting the class c having the maximum average probability.

### F. Mapping Generation

The instance matching model classifies the instance pairs as *'matches'* or *'nonmathces'*. Mappings are generated for instance pairs that *match* as a triple $< a_i, \equiv, b_j >$ or $< a_i,$ *owl:sameAs* $, b_j >$indicating that instances with identifiers $a_i$ and $b_j$ represent the same real world entity[39].

## 4. Experimental Evaluation

A systematic evaluation of instance matching techniques using real-world datasets are presented here. The proposed framework was subjected to evaluation on two real world datasets. Both the datasets belong to E-Commerce domain.

### A. Datasets Used

The dataset *Amazon-Google Products(D1)* comprise of instances from Amazon.com and Google Products. The dataset *Abt-Buy(D2))* features instances from Abt.com and Buy.com. A gold standard containing perfect mapping between the data sources is provided along with datasets. The deatiled description of the datasets are shown in Table No I.

### B. Performance metrics

The performance of different classification algorithms are evaluated using the following metrics : *accuracy, recall, precision and F1-Score*. The accuracy metric measures the correctness of predictions made by a model by calculating the proportion of correctly predicted instances over the total number of instances in a dataset. Precision measures the ability of classifier to find total number of predictions. Instance matching problems face the issue of *class imbalance problem* as the the number of matches(positive class) is very

less compared to non-matches. Therefore the performance of classifier is evaluated also using *F1-score* that take *Recall* and *Precision* into account.

- **Precision** (12) is a metric that calculates the proportion of accurately predicted positive instances relative to the total number of instances predicted as positive by the model.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (12)$$

- **Recall** Recall is calculated as the proportion of accurate positive predictions in relation to the overall count of positive predictions in the dataset ( 13). It can be mathematically represented as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (13)$$

- **F1-score** is obtained by calculating the harmonic mean of recall and precision. It integrates precision and recall by assigning each metric an equivalent weight. The F1-score is computed utilising the formula provided below in14

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (14)$$

- **ROC-AUC** Area Under the ROC Curve ( 15) is a singular scalar value reflecting the overall performance of a classifier across various threshold settings. A higher ROC AUC signifies superior classifier performance.

$$ROCAUC = \int_0^1 \text{TPR}(fpr)\, \text{d(FPR)} \quad (15)$$

Where True Positive Rate (TPR), which is equivalent to Recall, signifies the ratio of accurately predicted positive instances to the overall count of positive instances that actually occurred. The False Positive Rate (FPR) is calculated as the proportion of false positive predictions in relation to the overall count of true negative instances.

### C. Experimental setup

The experiment was conducted by creating training and test datasets using 10 fold cross validation along with stratified sampling. Each fold underwent two repetitions with stratified sampling, preserving the original class distribution. We oversampled the minority class and undersampled the majority class when constructing the training set. However the severe imbalance due to the presence of large number of mismatched instance pairs was handled by generating synthetic instances for the minority class, thereby augmenting its representation in the dataset known as synthetic minority over sampling(SMOTE)[40].

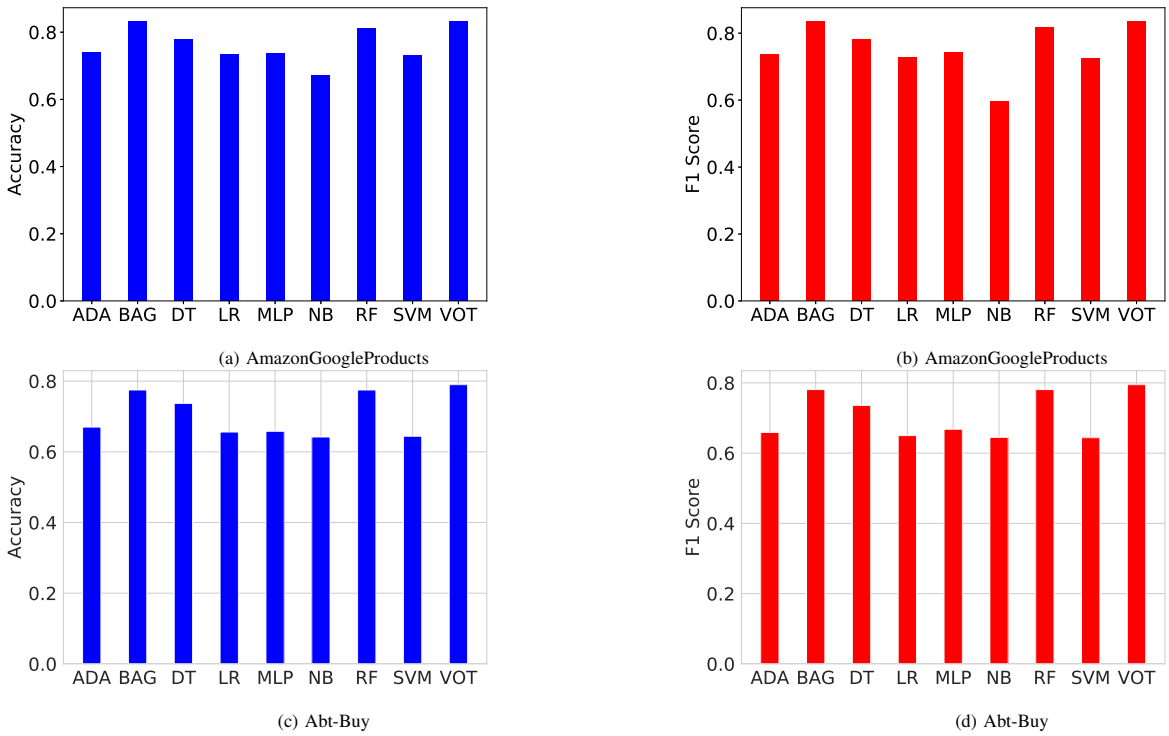
Classifiers were trained on datasets prior to oversampling through repeated cross validation to serve as baseline classifier. The process was repeated using over sampled datasets . Several machine learning algorithms have been used to evaluate the datasets. Out of these algorithms *Naive Bayes(NB)* and *Logistic regression(LR)* are probabilistic algorithms, *Multilayer Perceptron(MLP)* algorithm is based on feed forward neural network and *Decision tree(DT)* and *Random Forests(RF)* are rule based tree models.Decision tree algorithm J48 which is an implementation of C4.5 , *k-nearest neighbour (k-NN)* , *naive Bayes(NB)* are used as base line classifiers. We compare the performance of these classifiers with two hybrid classifiers namely *H1-BagBoost* and *h2-Voting classifier*. Experiments were carried out using Python langauage. For baseline classifers we use the default parameter settings.

## 5. Results and Discussions

Table. II show the prediction performances of the classifier models over the dataset *D1( Amazon GP )*. The accuracy of H2-VOT is best compared to other classifier models with 83% . The accuracy of H1BB and RF is approximately 81.The Figure 2(a)and (b) shows the comparison of accuracy and f1-score of various models on D1.

Similarly prediction performances of the classifier models over the dataset *D2(Abt-Buy )* is shown in Table. III. H2-VOT and RF outperforms all other classifiers with the highest accuracy and F1-score (79%) . The accuracy of H1BB is approximately 77%.The Figure 2 (c)and (d) shows the comparison of accuracy and f1-score of models on dataset Abt-Buy.It can be seen that the ensemble models namely RF,H1(BB) and H2(VOT) performs better and are effective than other classifier models as they aggregate predictions from diverse models, which helps smooth out biases and reduce variance. Although Decision Tree (DT) and k-Nearest Neighbors (KNN) classifiers achieve relatively high accuracy 73% and 75%, respectively, they demonstrate slightly lower F1-scores compared to H2VOT and RF and ensemble methods. This suggests that they may be prone to imbalanced performance w.r.t precision and recall. Naïve Bayes (NB), Logistic Regression (LR), Multi-Layer Perceptron (MLP), and Support Vector Machine (SVM) classifiers yield accuracy ranging from 0.642 to 0.658, indicating moderate performance compared to other classifiers.

Precision and Recall provide great insight into classification performance as the data is imbalanced.A model with higher precision yields more relevant results than irrelevant ones. The models H2(Vot) performs the best in both databases. H1(BB) performs well on D1 with a precision of 81% where not so well in D2. RF return more relevant results on D1 where as H2(VOT) 77%, SVM 77% and DT 76% are better at precision than other models as seen in **??**. Analyzing recall results of classification models




(a) AmazonGoogleProducts

(b) AmazonGoogleProducts

(c) Abt-Buy

(d) Abt-Buy

Figure 2. Comaprison of classifiers on datasets

is crucial for understanding how well the models perform in correctly identifying instances of a particular class. A high recall indicates that the model is effectively capturing most of the positive instances. Highest recall is achieved by KNN 87% followed by H2(Vot) and H1(BB) on D1. Ensemble methods like the voting classifier help reduce overfitting, especially when the individual base models are diverse and not highly correlated. Aggregating the predictions of multiple models helps in mitigating the variance in the predictions, leading to more generalizable results.

*1) Comparison with existing systems*

The performance of Instance matching [41] shows f1-score to be46% with $\{trainingratio = 90\%, skew = true\}$. Our models H1(Bag) and H2(VOT) in similar settings gives f-score of 81% and83% respectively. We have also observed grid search optimization performs better than random search optimization as in stated in [41]. The f1-score obtained for abt-Buy dataset for our models are 78% and 79% respectively as opposed to 33% F-score. A comparison of many machine learning models and deep learning model is given in[42] and the highest f-1 score achieved is shown as 69.3% , 62.8% respectively for Dataset1 and Dataset2. Therefore the performance exhibited by these models are comparable with state-of-art systems.

## 6. Conclusions and Future Work

The instance matching approach presented in this study used various techniques to address challenges such as data hetrogeneity, large number of comparison of instance pairs, identification of true matches that are of atmost importantance in real-world scenarios. Through the utilization of synthetic instance generation methods such as SMOTE, the imbalanced class distribution was effectively mitigated, enhancing the robustness of the classifiers. The experimental results demonstrated notable improvements in prediction performance across different datasets, with ensemble models like H2-VOT , H1BagBoost and RF consistently outperforming individual classifiers. These ensemble methods exhibited superior accuracy and F1-scores, showcasing their ability to aggregate diverse model predictions and mitigate biases, thus yielding more reliable outcomes. Furthermore, precision and recall analysis revealed valuable insights into the classification performance, with ensemble methods demonstrating strong capabilities in both metrics, particularly in correctly identifying positive instances while minimizing false positives.

Comparisons with existing systems underscored the effectiveness of the proposed approach, achieving significantly higher F1-scores compared to prior methodologies. Also, the utilization of grid search optimization improved the performance of model compared to results obtained through random search optimization. The findings indicate that instance matching approach using ensemble techniques can be successfully put to use for application in real-world scenarios.

In future work, enhancements to the instance matching approach involving advanced ensemble techniques to poten-

tially improve classification performance can be employed. Further investigation for the integration of deep learning models may offer opportunities to capture more intricate patterns and dependencies within the data, potentially leading to enhanced predictive capabilities. Also, incorporating domain-specific knowledge through ontology could potentially enhance the interpretability and effectiveness of the classification models. Lastly, conducting comprehensive experiments on larger and more diverse datasets from various domains could provide valuable insights into the scalability and generalizability of the proposed methodology across different real-world applications.

## References


[1] F. Wang and G. Liu, "Exploiting wikipedia for entity disambiguation in data interlinking," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 10, pp. 2255–2268, 2017.

[2] H. Jones and I. Smith, "Calculation of semantic distance on linked data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 9, pp. 1785–1798, 2020.

[3] C. Bizer, M. Vidal, and H. Skaf-Molli, "Linked open data," *Encyclopedia of Database Systems*, pp. 2096–2101, 2018.

[4] N. Kumar and S. Kumar, "Querying rdf and owl data source using sparql," in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 2013, pp. 1–6.

[5] A. Barbosa, I. I. Bittencourt, S. W. Siqueira, D. Dermeval, and N. J. Cruz, "A context-independent ontological linked data alignment approach to instance matching," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 18, no. 1, pp. 1–29, 2022.

[6] D. Song, F. Schilder, S. Hertz, G. Saltini, C. Smiley, P. Nivarthi, O. Hazai, D. Landau, M. Zaharkin, T. Zielund, H. Molina-Salgado, C. Brew, and D. Bennett, "Building and querying an enterprise knowledge graph," *IEEE Transactions on Services Computing*, vol. 12, no. 3, pp. 356–369, 2019.

[7] A. M. Alsukhayri, M. A. Aslam, I. H. Khan, R. A. Abbasi, and A. Babour, "Toward building a linked open data cloud to predict and regulate social relations in the saudi society," *IEEE Access*, vol. 10, pp. 50 548–50 561, 2022.

[8] N. Shadbolt, K. O'Hara, T. Berners-Lee, N. Gibbins, H. Glaser, W. Hall, and M. schraefel, "Linked open government data: Lessons from data.gov.uk," *IEEE Intelligent Systems*, vol. 27, no. 3, pp. 16–24, 2012.

[9] A. Nikolov, V. Uren, and E. Motta, "Data linking: capturing and utilising implicit schema-level relations," in *Proceedings of the WWW2010 Workshop on Linked Data on the Web, LDOW 2010, Raleigh, USA, April 27, 2010*, ser. CEUR Workshop Proceedings, vol. 628, 2010.

[10] J. Wang, J. Li, and Q. Zhang, "A new hybrid semantic similarity measure based on wordnet," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 628–639, 2015.

[11] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson Education, 2005.

[12] J. Smith and L. Johnson, "Record linkage: Finding similar records with domain-specific similarities," *Journal of Data Integration*, vol. 10, no. 3, pp. 215–230, 2021.

[13] M. Asif-Ur-Rahman, B. A. Hossain, M. Bewong, M. Z. Islam, Y. Zhao, J. Groves, and R. Judith, "A semi-automated hybrid schema matching framework for vegetation data integration," *Expert Systems with Applications*, vol. 229, p. 120405, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423009077

[14] A. Smith, B. Jones, and C. Brown, "Probabilistic record linkage in distributed healthcare data," *Journal of Biomedical Informatics*, vol. 62, pp. 161–169, 2016.

[15] A. Nikolov, V. Uren, and E. Motta, "Data linking: capturing and utilising implicit schema-level relations," in *Proceedings of the WWW2010 Workshop on Linked Data on the Web, LDOW 2010, Raleigh, USA, April 27, 2010*, ser. CEUR Workshop Proceedings, vol. 628, 2010.

[16] A. Smith and B. Jones, "Graph matching approach to data interlinking," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 5, pp. 1123–1135, 2016.

[17] C. Johnson and D. Brown, "Reconciliation algorithm for data interlinking with context information," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 3, pp. 679–693, 2018.

[18] J. Doe and E. Smith, "Context-based reconciliation algorithm for data interlinking," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 7, pp. 1421–1434, 2019.

[19] X. Liao, Y. Li, Y. Lou, X. Ge, S. Gao, and P. Sun, "The sg-cim entity linking method based on bert and entity name embeddings," in *2023 4th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, 2023, pp. 362–366.

[20] A. Assi, H. Mcheick, A. Karawash, and W. Dhifli, "Context-aware instance matching through graph embedding in lexical semantic space," *Knowledge-Based Systems*, p. 104925, 08 2019.

[21] P. Cudré-Mauroux, P. Haghani, M. Jost, K. Aberer, and H. de Meer, "idmesh: Graph-based disambiguation of linked data," in *Proc. 18th International World Wide Web Conference (WWW)*. Madrid, Spain: ACM, 2009, pp. 591–600.

[22] A. Smith and B. Jones, "Probabilistic approach for entity disambiguation in uncertain data environments," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1736–1749, 2017.

[23] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux, "Large-scale linked data integration using probabilistic reasoning and crowdsourcing," *The VLDB Journal*, vol. 22, no. 5, pp. 665–687, 2013.

[24] F. Saıs, N. Niraula, N. Pernelle, and M.-C. Rousset, "LN2R – a knowledge based reference reconciliation system: OAEI 2010 Results," in *Proceedings of the Ontology Matching Workshop (OM-2010)*, 2010.

[25] D. Song, Y. Luo, and J. Heflin, "Linking heterogeneous data in the semantic web using scalable and domain-independent candidate selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 143–156, 2017.

[26] A. Sharma *et al.*, "An efficient candidate generation technique for

large-scale entity matching," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 9, pp. 1978–1991, 2017.

[27] M. J. Hasan, W. H. Chiang, M. Jiang, C. Li, S. Lv, K. Tu, T. L. Veldhuizen, J.-R. Wen, W. Zhang, and K. Zheng, "Efficient candidate generation for large-scale entity matching using locality-sensitive hashing," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1134–1145, 2016.

[28] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "Semantic similarity-based instance matching," in *International Semantic Web Conference (ISWC)*, 2011, pp. 333–348.

[29] A. Smith and B. Johnson, "Data cleaning techniques for improved data quality," *Journal of Data Science*, vol. 15, no. 2, pp. 123–135, 2019.

[30] C. Brown and D. Wilson, "Handling multilingual data in information systems: Challenges and solutions," *Journal of Information Systems Management*, vol. 25, no. 4, pp. 321–335, 2017.

[31] J. Smith and A. Johnson, "Concept identification techniques for structured data extraction," *Journal of Data Science*, vol. 10, no. 2, pp. 123–135, 2020.

[32] L. Jones *et al.*, "Pattern matching algorithms for concept extraction in structured datasets," *ACM Transactions on Information Systems*, vol. 25, no. 3, pp. 345–358, 2017.

[33] R. Doe and S. Roe, "Ontology-based concept extraction for semantic annotation of electronic product data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 5, pp. 987–1001, 2018.

[34] J. Li, Z. Wang, X. Zhang, and J. Tang, "Large scale instance matching via multiple indexes and candidate selection," *Knowledge-Based Systems*, vol. 50, p. 112–120, Sep 2013. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2013.06.004

[35] G. Zuccon and L. Azzopardi, "A multinomial relevance model for retrieval," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016, pp. 363–372.

[36] C. Zhai and J. Lafferty, "Statistical language models for information retrieval: A critical review," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 3, pp. 137–213, 2008.

[37] F. Zhao, Z. Tian, and H. Jin, "Entity-based language model smoothing approach for smart search," *IEEE Access*, vol. 6, pp. 9991–10 002, 2018.

[38] A. Eibeck, S. Zhang, M. Q. Lim, and M. Kraft, "A simple and efficient approach to unsupervised instance matching and its application to linked data of power plants," *Journal of Web Semantics*, vol. 80, p. 100815, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570826824000015

[39] J. Dao, S. T. Ng, and C. Y. Kwok, "Interlinking bim and gis data for a semantic pedestrian network and applications in high-density cities," *Developments in the Built Environment*, vol. 17, p. 100367, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666165924000486

[40] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[41] M. Kejriwal and D. P. Miranker, "Decision making and bias," in *Proceedings of the 14th International Semantic Web Conference (ISWC 2015)*, 2015.

[42] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra, "Deep learning for entity matching: A design space exploration," in *Proceedings of the 2018 International Conference on Management of Data (SIGMOD)*. ACM, 2018.