AN INTELLIGENT ANALYSIS OF POLICING DATA FOR IDENTITY RESOLUTION

AI and Data Science Research Group

School of Computing and Digital Media

London Metropolitan University

United Kingdom

The dissertation is submitted for the degree of Doctor of Philosophy (PhD)

By

Asif NAWAZ

Supervisor:

Professor Hassan Kazemian

October 2024

ABSTRACT

Identity refers to the unique characteristics or attributes that distinguish an individual. Identity crimes, such as theft or fraud, occur when someone unlawfully acquires and uses personal information for fraud. Identity resolution, the process of identifying and merging duplicates or similar entries, is critical for law enforcement agencies globally. However, matching identities in big data presents challenges due to inconsistencies, including typographical errors, naming variations, and abbreviations. Traditional record and identity matching techniques aim to consolidate or eliminate redundant data entries, ensuring accuracy and integrity. Manual identity matching is infeasible in big data environments. However, machine learning techniques offer a solution by automating pattern extraction and reducing reliance on manually coded rules.

This research proposes a fuzzy approach for identity resolution, combining unsupervised learning with fuzzy string similarity metrics to improve identity matching. The model incorporates an iterative search process using a combination of the Soundex and Jaro-Winkler algorithms to compute an aggregate score for names. The Soundex method has been enhanced to generate a six-digit numerical code, overcoming traditional limitations. Additionally, with the help of the FuzzyWuzzy Python library, the Edit-distance algorithm is applied to match attributes such as “address” and “ethnicity description.” The Mean-Shift clustering technique dynamically generates clusters based on the final dataset, avoiding needing a predefined number of clusters.

The three name variations of the iterative search process allow the categorisation of records into Match, Related or Close Match, and Possible Match while excluding duplicates. By grouping entities based on similarity scores and applying graph analysis, the framework accurately identifies target identities, even when links span different addresses. The results demonstrate the framework’s ability to enhance the speed and accuracy of identity resolution, offering a more efficient method than existing solutions.

This research significantly contributes to identity resolution techniques, improving investigative processes with minimal information and offering valuable applications for law enforcement and other sectors, such as fraud detection in the financial industry.

ACKNOWLEDGEMENT

I extend my profound gratitude to the esteemed individuals and organisations whose invaluable contributions have greatly influenced the successful completion of this doctoral thesis:

First and foremost, I express my most profound appreciation to my esteemed lead supervisor**, *Professor Hassan Kazemian***. His exceptional guidance, unwavering support, and scholarly expertise have been instrumental in shaping the trajectory of this research endeavour. I thank my second supervisor, **Dr Rick Adderley,** for providing guidance and de-identified policing data for this research. Their insightful feedback and constructive criticism have significantly enhanced the quality and rigour of this thesis.

I would also like to thank *ITCE Bank of China UK* for their financial support, which has been crucial in completing this research. Their investment and belief in the significance of this study have provided the necessary resources and opportunities to carry out this work.

Lastly, I extend my deepest gratitude to all individuals, organisations, and institutions involved in this journey, directly or indirectly. Your support, encouragement, and belief in my abilities have been invaluable. This thesis stands as a testament to our collective efforts.

DEDICATION

This thesis is dedicated to my:

Late Parents,

I dedicate this thesis to my beloved late parents in the loving embrace of cherished memories. Reflecting on their unwavering love and the invaluable lessons they taught me, my heart swells with gratitude, admiration, and profound sadness. Their absence felt deep, yet their presence lingers in every step I take on this academic journey. This thesis is a homage to their enduring impact, a testament to the love that continues to shape and guide me, even in their physical absence.

Beloved Wife,

To my unwavering pillar of strength, my wife, I dedicate this thesis with overflowing emotions and gratitude. You have been a constant source of solace and support in the face of loss and longing. Through the tears and moments of overwhelming grief, your unwavering love has carried me forward. This thesis is not only a celebration of our shared dreams and triumphs but also a tribute to the love we carry with us, a love that transcends boundaries and reaches out to embrace the memories of my late parents.

To the cherished memories of my late parents and my beloved spouse, this thesis is a testament to the indomitable power of love, resilience, and the bittersweet emotions shaping our lives. Their love continues to reside within me, filling every chapter of this academic journey with a profound sense of purpose and remembrance.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

1. INTRODUCTION

Fraud is a significant ongoing threat to society, the economy, law enforcement agencies, and other institutions globally, and it remains a complex task. It has become a great challenge for law enforcement agencies to identify the correct identity among the false identities from a colossal identity pool. For example, one might have many identities that can be used differently. Before diving deep into the problem solution, some basic understanding of key terms will be established in the sections below.

1.1. Fraud and Types of Fraud

Fraud refers to the act of deceiving or intentionally misleading others for personal or financial gain. It typically involves dishonesty, misrepresentation, or manipulation to deceive individuals, organisations, or systems (Albrecht W, Albrecht C, A, 2008; Pedneault *et al.*, 2012). There are various types of fraud, including the following that are described by (Hedayati, 2012) :

- Identity Theft: This occurs when someone steals another person's personal information, such as Social Security numbers, credit card details, or bank account information, to commit fraudulent activities.
- Credit Card Fraud: This type of fraud involves using another person's credit card information to purchase or withdraw funds without their consent.
- Insurance Fraud: This fraud involves false claims or exaggerating events or damages to receive insurance benefits illegally.
- Investment Fraud: This fraud encompasses fraudulent schemes or practices that deceive investors into making decisions based on false or misleading

information. Ponzi schemes and pump-and-dump schemes are examples of investment fraud.

- Healthcare Fraud: It refers to fraudulent activities in the healthcare industry, such as submitting false claims, overbilling, providing unnecessary treatments, or selling counterfeit drugs.

- Tax Fraud: This involves intentionally providing false information on tax returns to avoid paying taxes or obtaining tax refunds illegitimately.

- Online Scams: These frauds occur online, including email scams, phishing attacks, online auction fraud, and pyramid schemes conducted through online platforms.

- Mortgage Fraud: This fraud involves providing false information or engaging in illegal activities to obtain a mortgage loan, such as inflating property values or misrepresenting financial information.

- Wire Fraud: This fraud refers to fraudulent activities conducted through electronic communications, such as email or online messaging, to deceive individuals or organisations into sending money or sensitive information.

- Employment Fraud: This includes fraudulent practices related to employment, such as fake job postings, résumé fraud, or fraudulent recruitment agencies.

These are just a few examples of fraud, and there are many other variations and combinations of fraudulent activities that individuals or groups may engage in for personal gain. It is vital to remain vigilant and take necessary precautions to protect oneself from fraud.

1.2. Identity and Types of Identity

However, one true identity must be identified at the right time among all the identities. (Niblett, 2015) defined, “Identity is the root of who we are as individuals when it comes to the matter of trust”. Matching identity is a technique to find a relationship between

two or more identities of the same person. Identity can be understood in different contexts, and various identities exist. Here are some common types of identities:

- Personal Identity: Personal identity refers to the unique characteristics, experiences, beliefs, and values that define an individual as a distinct person. It includes gender, age, ethnicity, nationality, sexual orientation, religion, and personal interests (Li and Wang, 2015).

- Cultural Identity: Cultural identity encompasses the shared beliefs, customs, traditions, language, and values that shape a person's sense of belonging to a particular cultural group or community. It can include ethnic, regional, or national identities (McCallum-Bayliss, 2004).

- Social Identity: Social identity refers to an individual identifying with the group or groups. Typically, this includes family, occupation, socioeconomic status, political affiliation, or membership in specific communities or organisations (Li and Wang, 2011).

- National Identity: National identity relates to an individual's sense of belonging and allegiance to a particular nation or country. It includes a shared sense of history, culture, language, and citizenship (Soltani and Abhari, 2013).

- Gender Identity: Gender identity is an individual's internal sense of gender, which may or may not align with the sex assigned to them at birth. Gender identity can be male, female, or non-binary, among other identities along the gender spectrum (Li and Wang, 2015).

- Professional Identity: Professional identity encompasses the roles, skills, and values of a person's chosen profession or career. It includes professional affiliations, qualifications, and the sense of professional purpose and identity within a specific field (Yan, Bajaj and Bhasin, 2011).

- Online Identity: With the growth of the internet and social media, online identity has become significant. It refers to the persona or representation of oneself

created through online platforms, including usernames, profiles, and online interactions (Yadav, Sinha and Kumar, 2019).

- Self-identity: Self-identity encompasses an individual's subjective understanding and perception of oneself, including self-image, self-esteem, and self-concept. Personal experiences, beliefs, and values can influence it (Chung *et al.*, 2014).

It is important to note that identities are complex and multidimensional, and individuals may identify with multiple identities simultaneously. Additionally, identities can evolve and change over time as individuals develop and experience new aspects of their lives.

1.3. Types of Identity Crimes

Identity refers to the distinguishing characteristics, traits, or attributes that make an individual unique. Identity crimes, also known as identity theft or fraud, occur when someone wrongfully obtains and uses another person's personal information for fraud. Using this stolen information can commit various types of identity crimes described by (Albrecht W, Albrecht C, A, 2008; Hedayati, 2012) , such as:

- Financial Identity Theft: This involves using another person's personal information, such as Social Security numbers, credit card details, or bank account information, to make unauthorised financial transactions, open fraudulent accounts, or apply for loans or credit cards.

- Medical Identity Theft: In this type of identity crime, someone fraudulently uses another person's personal information to receive medical services, prescriptions, or insurance coverage. It can lead to incorrect medical records, fraudulent insurance claims, and potential health risks for the victim.

- Criminal Identity Theft: This occurs when someone uses another person's identity during the commission of a crime. The criminal can provide false

identification to law enforcement or use the stolen identity to avoid detection or prosecution.

- Synthetic Identity Theft: Synthetic identity theft involves creating a new identity by combining real and fake information. Fraudsters may use a combination of stolen and fabricated details to establish credit or conduct fraudulent activities.
- Child Identity Theft: This type of identity theft targets minors. The perpetrator may use a child's personal information, such as their Social Security number, to open fraudulent accounts or commit financial fraud. Since children typically have limited financial activity, their stolen identities can go undetected for years.
- Social Media Identity Theft: With the increasing use of social media, individuals' personal information can be exploited. Fraudsters may use stolen identities to create fake profiles, conduct scams, or spread malicious content.
- Tax Identity Theft: Tax identity theft occurs when someone uses another person's Social Security number or other identifying information to file fraudulent tax returns and claim refunds illegally.

It is crucial to safeguard personal information and regularly monitor financial and personal records to detect any signs of identity theft. Taking preventive measures like using strong passwords, being cautious of phishing attempts, and shredding all personal documents can help mitigate the risk of falling victim to identity crimes.

1.4. Machine Learning

Machine learning is a subfield of artificial intelligence that focuses on developing algorithms and models that enable computers to learn and make predictions or decisions without explicit programming. Machine learning algorithms are designed to automatically analyse and interpret complex patterns and relationships in data and use that knowledge to perform specific tasks or make predictions. Machine learning can be

broadly categorised into the following types (Christopher M. Bishop, 2006; Hastie, Tibshirani and Friedman, 2009):

1.4.1. Supervised Learning

Supervised learning involves training a model on labelled data, where corresponding desired outputs or labels accompany the input data. The model learns from this labelled data and generalises the patterns to make predictions or classify new, unseen data. Examples of supervised learning algorithms include linear regression, logistic regression, decision trees, random forests, support vector machines (SVM), and neural networks (Christen, Vatsalan and Wang, 2016; Jurek *et al.*, 2017).

1.4.2. Unsupervised Learning

Unsupervised learning involves training a model on unlabelled data, where the algorithm must discover patterns or structures in the data without prior knowledge of the outcomes or labels. Unsupervised learning algorithms aim to find meaningful representations, groupings, or relationships in the data. Clustering algorithms, such as k-means and hierarchical clustering, Self-organising Maps, and dimensionality reduction techniques, such as principal component analysis (PCA), are typical examples of unsupervised learning (Kohonen, 1990; Du, 2010). The supervised learning will be discussed in later chapters.

1.4.3. Semi-Supervised Learning

Semi-supervised learning combines elements of supervised and unsupervised learning. It involves training a model on a mixture of labelled and unlabelled data. The model learns from the labelled data to predict new, unseen data and uses the unlabelled data to enhance its understanding of the underlying patterns or structures (Sun, 2013).

1.4.4. Reinforcement Learning

Reinforcement learning involves training an agent to interact with an environment and learn optimal actions to maximise a reward signal. The agent learns through trial and error, receiving feedback as rewards or penalties based on its actions. Reinforcement learning is commonly used in robotics, game-playing, and autonomous systems (Christopher M. Bishop, 2006).

Machine learning algorithms can be applied to a wide range of tasks, including:

- Classification – Predicts the class or category of an input based on its features and, for example, classifying emails as spam or non-spam (Jurek *et al.*, 2017).
- Regression – Predicts a continuous value or outcome based on input features, for example, predicting housing prices based on location, size, and number of rooms (Yan *et al.*, 2020).
- Clustering – Groups similar data points based on their attributes or characteristics, for example, segmenting customers into distinct groups based on their purchasing behaviour (Bezdek, 1981).
- Anomaly detection – Identifies rare or abnormal data points or events that deviate from the norm, for example, detecting fraudulent transactions or network intrusions (Studiawan, Payne and Sohel, 2017).
- Recommendation systems – Recommends items or content to users based on their preferences and behaviour, for example, suggesting movies or products based on previous interactions (Ghahramani, 2015).
- Natural Language Processing (NLP) – Analyse and understand human language, including tasks like text classification, sentiment analysis, and machine translation (Liao and Zhao, 2019).
- Computer Vision – Extract meaningful information and patterns from images or videos, including tasks like object recognition, image segmentation, and facial recognition (Liu *et al.*, 2013).

Machine learning has become increasingly popular and widely used across various industries and domains, driving advancements in data analysis, automation, and decision-making.

1.5. Understanding Basic Concepts of Matching Techniques

1.5.1. Record Matching

Record matching, or entity resolution or deduplication, identifies and merges duplicate or similar records within a dataset. In record matching, various attributes or fields in records are compared to determine if they likely refer to the same entity. The goal is to consolidate or eliminate redundant or duplicate data entries to ensure data accuracy and integrity. The comparison can be based on criteria such as name, address, phone number, or other identifying information. By identifying and merging duplicate records, record matching helps create a clean and consolidated dataset that avoids data redundancy and inconsistency (Bharambe, Jain and Jain, 2012).

For example, in a customer database, record-matching techniques can identify and merge duplicate entries for the same customer based on criteria such as name, address, phone number, or other identifying information. Usually, this helps create a clean and consolidated database with accurate customer information. The following section will briefly highlight the records matching techniques.

1.5.1.1. Records Matching Techniques

- ***Deterministic Matching***

This technique uses strict rules or algorithms to compare specific fields or attributes between records. It relies on exact matches or predefined rules to identify duplicates or

similarities, for example, by comparing Social Security numbers or unique identifiers to find exact matches (Sayers *et al.*, 2016).

➲ *Probabilistic Matching*

This technique assigns probabilities or weights to the similarity of different attributes between records. It considers fuzzy matches and calculates a likelihood score to determine the similarity or likelihood of a match. Probabilistic matching techniques use algorithms like the Jaro-Winkler distance or the Levenshtein distance to quantify the similarity between strings or attributes (Fellegi and Sunter, 1969; Sayers *et al.*, 2016).

➲ *Rule-Based Matching*

This technique employs predefined rules or logic to compare and match records. Rules can be defined based on specific criteria, such as matching names, addresses, or phone numbers. Rule-based matching allows for flexibility in defining the conditions and thresholds for matching (Christen, 2012).

➲ *Machine Learning-Based Matching*

Machine learning techniques can be applied to record matching by training models on labelled data. These models learn patterns and similarities from the data to identify potential matches. Supervised learning algorithms, such as decision trees, random forests, or support vector machines, can be used for matching records (Christen, Vatsalan and Wang, 2016).

1.5.2. Identity Matching

Identity matching, on the other hand, focuses on verifying or establishing the identity of an individual or entity. It involves comparing different attributes or data points to

determine if they belong to the same identity or if they match against a known identity (Soltani and Abhari, 2013).

Identity matching can be used in various contexts, such as stated by (Boongoen and Shen, 2009):

- Law enforcement agencies use fingerprints or DNA matching to identify suspects or link crime scenes.
- Financial institutions verify the identity of customers through "Know Your Customer (KYC)" processes, comparing personal information and identification documents.
- Online platforms use identity-matching techniques to authenticate users during account registration or login processes.

Identity matching techniques may employ various data points, including personal identifiers, biometric data, photographs, or unique identifiers like Social Security or passport numbers. The goal is to ensure accurate identification and prevent fraud or unauthorised access.

While record matching focuses on identifying and merging duplicate records within a dataset, identity matching focuses on establishing the identity of individuals or entities by comparing data points to known identities or reference data. Both techniques serve distinct purposes in data management and identification processes.

1.5.2.1. Identity Matching Techniques

➲ *Biometric Matching*

Biometric matching involves comparing biometric data, such as fingerprints, iris patterns, or facial features, to establish identity. Biometric systems use mathematical algorithms to analyse and match the unique characteristics of an individual's biometric data (Jain, Ross and Prabhakar, 2004).

➲ *Document Verification*

This technique involves comparing identification documents, such as passports or driver's licenses, to establish identity. It may involve manual verification by experts or automated systems that analyse the document's security features and data consistency or compare it against known reference data (Wang and Dong, 2020).

➲ *Knowledge-Based Authentication*

Knowledge-based authentication involves verifying an individual's identity through information they should know, such as personal identification questions, passwords, or PINs. This technique assumes that the person claiming an identity possesses knowledge specific to that identity (Jain, Ross and Prabhakar, 2004).

➲ *Multifactor Authentication*

Multifactor authentication combines multiple identity verification techniques. It typically involves a combination of something the individual knows, e.g., a password, something they have, e.g., a smart card or mobile device, or something they are, e.g., biometric data. By requiring multiple factors, multifactor authentication enhances the security and reliability of identity matching.

It is important to note that the selection of matching techniques depends on the specific context, data quality, available resources, and desired level of accuracy. Different techniques may be employed to achieve accurate and reliable matching or identification results (Jain, Ross and Prabhakar, 2004).

1.5.3. Deterministic Matching

Deterministic matching is a technique used in record matching where strict rules or algorithms are applied to compare specific fields or attributes between records. The goal is to identify exact matches or predefined patterns to determine if two records refer to the same entity. Deterministic matching relies on precise matching criteria and predefined rules to identify duplicates or similarities. In deterministic matching, specific fields, such as names, addresses, phone numbers, or unique identifiers, are compared between records. The matching process follows predetermined rules, such as requiring an exact match or a specific pattern to consider records as a match, for example, by comparing Social Security numbers or unique customer IDs to find exact matches. Deterministic matching is often faster and more straightforward than probabilistic matching as it relies on strict rules (Aiken *et al.*, 2019). However, it may miss potential matches if the data has inconsistencies or minor variations, such as misspellings or formatting differences. This technique is effective when the data quality is high and exact matches or specific patterns are sufficient to identify duplicates or similarities.

1.5.3.1. Deterministic Matching Techniques

Deterministic matching techniques are to identify exact matches or predefined patterns between records. These techniques apply strict rules or algorithms to compare specific fields or attributes. Here are some commonly used deterministic matching techniques:

- ***Exact Matching***

  Exact matching involves comparing specific fields or attributes between records to find exact matches. This technique requires the values in the compared fields to be identical for the records to be considered a match, for example, by comparing unique identifiers like Social Security numbers or customer IDs to find exact matches (Al-khamaiseh and Alshagarin, 2014).

➲ *Rule-Based Matching*

Rule-based matching involves defining predefined rules or logic to compare and match records. The rules can have specific criteria or patterns that must be met for two records to be considered a match, for example, matching records based on exact matches of names, addresses, phone numbers, or a combination of multiple attributes (Yi *et al.*, 2020).

➲ *Key-based Matching*

Key-based matching involves selecting a specific attribute or set of attributes as the key identifier for matching records. Records with the same key value are considered a match, for example, by using a unique customer ID or a combination of attributes like name and address as the key for matching customer records (Dey, Mookerjee and Liu, 2011).

➲ *Token-based Matching*

Token-based matching involves breaking down attributes into smaller units or tokens and comparing these tokens between records. Tokens can be words, phrases, or specific patterns. The matching is based on the presence or absence of specific tokens or the order of tokens. Token-based matching helps handle variations or inconsistencies in textual attributes (Cohen, Ravikumar and Fienberg, 2003).

➲ *Dictionary-based Matching*

Dictionary-based matching involves creating a predefined dictionary or reference dataset with known values or patterns. Records are matched by comparing their attribute values against the entries in the dictionary. This technique is commonly used for matching standard names, addresses, or other reference data (Yi *et al.*, 2020).

➲ ***Hierarchical Matching***

Hierarchical matching involves establishing a hierarchical structure or grouping records based on specific attributes or criteria. Matching is performed at different levels of the hierarchy, allowing for more efficient matching by narrowing down the search space. This technique is effective when dealing with large datasets (Ravikumar and Cohen, 2004).

1.5.4. Probabilistic Matching

Probabilistic and fuzzy matching techniques are used in record matching to identify potential matches by assigning probabilities or weights to the similarity of different attributes between records. It considers fuzzy matches and calculates a likelihood score to determine the similarity or likelihood of a match. Probabilistic matching considers variations, inconsistencies, or errors in the data. It uses algorithms that measure the similarity between strings or attributes, such as the Jaro-Winkler distance or the Levenshtein distance. These algorithms quantify the degree of similarity between two strings by calculating the number of transformations required to convert one string into the other. The probabilistic matching process involves comparing multiple attributes between records and assigning weights or scores to each attribute's similarity. The final match score is calculated based on the combination of attribute scores and a threshold set to determine whether the records match. The threshold can be adjusted based on the desired level of precision and recall (Ravikumar and Cohen, 2004). It can identify potential matches that deterministic matching might overlook. However, it may also introduce false results if the threshold is too low or the data quality is poor.

1.5.4.1. Probabilistic Matching Techniques

Probabilistic and fuzzy matching techniques are used to identify potential matches between records by assigning probabilities or weights to the similarity of different

attributes. These techniques allow for flexibility in handling variations, inconsistencies, or errors in the data. Here are some commonly used probabilistic matching techniques:

➲ *Jaro-Winkler Distance*

The Jaro-Winkler distance is a string similarity measure that calculates the similarity between two strings by comparing their characters' positions and transpositions. It assigns a similarity score between 0 and 1, with 1 indicating a perfect match. The Jaro-Winkler distance compares names or other textual attributes (Winkler, 1994).

➲ *Levenshtein Distance*

The Levenshtein distance, also known as the edit distance, measures the minimum number of single-character edits required to transform one string into another. It calculates a distance value, and a similarity score can be derived by taking the inverse of the distance. The Levenshtein distance helps compare attributes with potential misspellings or slight variations (Ristad and N.yianilos, 1998).

➲ *Soundex*

Soundex is a phonetic algorithm that converts words or names into a four-character code based on pronunciation. It allows for matching based on similar-sounding names, even with different spellings. Soundex matches names or surnames that have different spellings but sound similar (A. J. Lait and Randell, 1996).

➲ *N-gram Matching*

N-gram matching involves breaking strings into smaller n-gram components and comparing these components between records. Similarity scores can be calculated by comparing the occurrence and position of n-grams. This technique helps match textual attributes that may have variations or typographical errors (Kukich, 1992).

➲ *Blocking*

Blocking is a technique that divides records into smaller subsets or blocks based on specific attributes or criteria. It reduces the number of record pairs that must be compared, focusing only on records within the same block. Blocking helps to improve the efficiency and speed of probabilistic matching algorithms (Kopcke and Rahm, 2010).

These probabilistic matching techniques can be combined and customised based on the specific requirements and characteristics of the matched data. The choice of technique depends on the nature of the attributes, data quality, and the desired level of precision and recall in the matching process. It is often necessary to experiment and fine-tune the parameters and thresholds to achieve the desired matching accuracy. Both deterministic matching and probabilistic matching techniques have their advantages and limitations. The selection of the appropriate matching technique depends on the specific use case and the quality and nature of the data being matched.

1.5.5. Record Linkage

Record matching and record linkage are separate concepts in data management and analysis. Record linkage, also known as data linkage or entity resolution across datasets, is the process of finding and connecting records across multiple datasets that refer to the same entity. It involves linking or associating records from different sources representing the same real-world entity. Record linkage goes beyond record matching within a single dataset and involves integrating and connecting data from various sources. The goal is to identify and establish connections between records that pertain to the same entity, even if the attribute values or formats may differ across datasets. Record linkage techniques typically involve comparing attributes or fields across datasets and assigning similarity or matching scores to determine the likelihood of a match. These techniques may use deterministic or probabilistic matching algorithms to

identify potential matches and establish links between records. Record linkage is used in scenarios such as data integration, data warehousing, or data analytics, where data from different sources must be combined and linked for comprehensive analysis or decision-making (Culotta and McCallum, 2005).

1.6. Understanding Identity Resolution

Traditionally, to match two or more records for similarities, some attributes are required, including name, date of birth, nationality, passport number and address. However, matching two records using attributes is insufficient for a person's true identity. For example, matching records by name will not resolve this issue as there can be many similar names, such as in China where "Wang", "Wu", and "Li" are common names (Lisbach and Meyer, 2013). While (Wang, Chen and Atabakhsh, 2004) stated that each entity has attributes and can be identified by these critical attributes, e.g., ID, name, and date of birth. These are used in traditional resolution techniques to describe the individual and are available in traditional record management systems. However, the traditional resolution techniques are ineffective due to typographical errors and intentional issues such as data entry errors and intentional errors for fraud. Many other reasons make it challenging to find the correct identity. According to (Barkay and Rein, 2015) analysis, a massive amount of unstructured, incomplete, and incorrect data is available publicly to extract the required information, leading to a monumental task. Differentiating and combining are the two main tasks for detecting the true identity by finding the association between the records.

It is a must for law enforcement agencies to understand the flow of information between individuals and other sources. This analysis helps to identify lead actors, their roles and specialisations, communication channels, knowledge distribution, and ultimately, vulnerabilities in the organisational structure that the agency can exploit. It requires large datasets and a stream of information to analyse the identity by following any change in the information or activity pattern, which can give some indication of the information flow of an individual to identify, such as travelling or financial information.

For example, an individual may stay under the radar if viewed as isolated. However, when considering regular meetings with unrelated people connected to a bank account, showing regular or irregular transfer patterns over a specific company may reveal a very different picture and be flagged as the lead offender.

It will help police question and track an individual for any criminal activity and match the identity in the database to find other links, associations and connections if the individual profile already exists. However, if this individual has changed the name from Billy Smith to Bobby Jones or changed these names concurrently, then in that case, it is a complex task for law enforcement agencies to identify and match the correct identity from the database. Because the change of name may mislead the police, and the lack of tools available to detect true identity will not be easy, primarily when the name has been used concurrently on different occasions. Figure 1.1 depicts this type of identity fraud pattern as described by (Adderley, 2015).

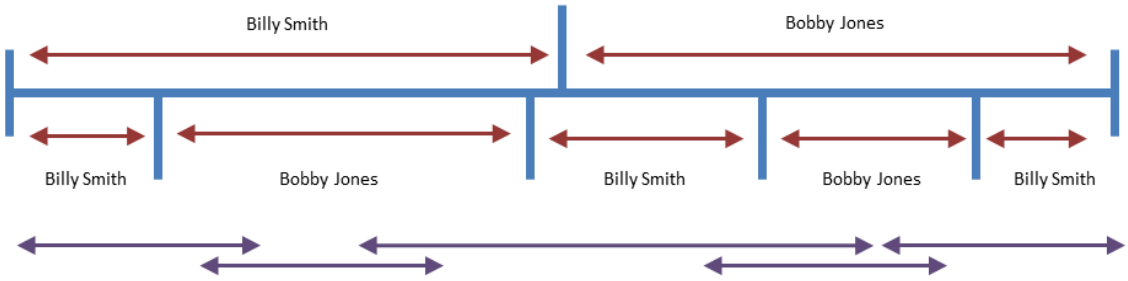

Figure 1.1 - Identity Resolution Scenario (Adderley, 2015)

According to the US report, many cases documented that terrorists and other criminals around the world commit identity crimes to achieve their financial needs and execute different attacks in the real as well as in the cyber world (US Department of State, 2008). Criminals can easily have fake identities, commonly used to mislead law enforcement agencies. The social contextual information can improve the resolution accuracy in addition to traditional identity attributes for identity matching (Bhattacharya and Getoor, 2007). Thus, detecting the identity fraud pattern is a very critical task in criminal structure as criminals use bogus identities to achieve malicious goals by hiding true identities, and such situations are critical for national security. Finding the solution for

identity duplication is critical as it will help fight terrorism and any other crime. Because false identities are very commonly used for crime and terrorism, this can mislead law enforcement agencies by having multiple identities (Li and Wang, 2015). (John S. Pistole, 2003) emphasised that law enforcement agencies must determine an individual identity to detect potential terrorists and prevent terrorism from occurring.

Detecting fraud becomes increasingly complex when the data grows with time. Traditional record-matching techniques can not accurately find the relationship within the records due to the poor quality of input data, with a chance of human typing errors and missing or incomplete information. As discussed above, the probabilistic and deterministic techniques can be used for record matching. The probabilistic technique typically uses training data to match records, which can result in lower accuracy. To resolve this issue, the system must be retrained each time to process the entire dataset (Sayers *et al.*, 2016). Meanwhile, the deterministic technique uses predefined rules to match the record. If the record does not satisfy the predefined rule, no match will be found, and no data will be collected (Jonas, 2006).

According to (Duncan *et al.*, 2015), both these techniques used for record matching and linkage divide the record into three states: “Match, No Match and Possible Match”, where the possible match might not be accurate if the information has been changed over time. Therefore, it will lead to an incorrect match being flagged as a possible match. According to (Godby *et al.*, 2009), record matching refers to entity resolution, which focuses on information extracted using names, while identity resolution is a technique to determine the extracted information belongs to whom and how it is linked to others in real-world associations.

Detecting the relationship between different data elements and entities while scanning individual records to match data is vital. Nevertheless, it is also essential to consider the difference between available datasets from different resources when matching any record, which might lead to another problematic task due to different dataset standards. According to (Jonas, 2006), this will lead to finding the uniqueness and commonalities between different datasets from different sources to answer who is who and who knows whom. Considering the above, it can lead to identity resolution rather than simply matching records, as finding link associations between records is essential for finding

true identity. Therefore, (Jonas, 2006) defined the semantic reconciliation process as identifying an individual identity as identity resolution, even if it is described differently.

It can be easily manipulated to distinguish between different records in a small volume dataset and find any link association between them. It is difficult for a human brain to accurately and efficiently match the identity as the data grows in the dataset. In a large dataset, matching and identifying the record is challenging; therefore, achieving this goal from computers using machine learning is a complex task. A step towards this task involves machine learning and data mining techniques such as supervised and unsupervised learning. The data mining techniques help to extract useful information and related patterns from the dataset, which can help in machine learning. Data mining uses link analysis to associate the data in a graph to detect the related patterns and find the links between them. It identifies any abnormal activity or occurrence (Brown and Hagen, 2002). In machine learning, the supervised learning technique uses a known set of predefined patterns to identify similarities in the new records. The unsupervised learning technique does not use predefined patterns to find similarities. It focuses on a dynamic algorithm to detect similarities in the object, which starts behaving differently from objects to which they were similar in the past. (Li and Wang, 2015) defined identity resolution as an entity resolution type used for identity management. In different domains, an entity resolution is known as record linkage and deduplication. A white paper published by (Dun & Bradstreet, 2013) describes that identity resolution helps in solid decision-making and detecting the correct information at the right time to make the right decisions. It brings a standard within the datasets by defining the entity. It allows performing different processes and techniques for analysis to identify entities. Identity Resolution is matching information and detecting, identifying, and considering past information associations for the target entity. According to (Edwards et al., 2016), linking a person's different variations of records to find the real-world identity is called identity resolution.

Therefore, this research establishes and introduces machine learning techniques and algorithms to match and identify related records. The following section will discuss the literature review to help understand the gaps.

1.7. Literature Review

Some foremost and serious incidents, such as the 9/11 New York terror attack in 2001 and the 7/7 London bombing in 2005, led law enforcement agencies, independent organisations, and institutions to rethink to find the best possible approach for harvesting and detecting the identities by finding similarities and links associations between multiples identities which are referring to one real identity. The only way to fulfil this approach is to enable a computer system intelligent enough to detect, identify, and find an association between different records using different record-matching techniques in less time and with less effort to produce positive outcomes. (Ananthakrishna, Chaudhuri and Ganti, 2002) proposed eliminating duplicates in the dataset by applying dimensional hierarchy over the link relations such as city, state, and country. This approach only matches the identity record if both identities belong to the same area; otherwise, the record will not be matched against similar entries in the data set for different areas.

Similarly, (Brown and Hagen, 2002) introduced the record-linking data association method to match the criminal records referring to the same record. In this method, the two records are compared by calculating the total similarity using the sum of the weighting of matching attribute values. However, this requires more computing power as the dataset increases with time and will not filter the referring records efficiently. It might also ignore records on fewer similarity measurements. Pasula et al., 2003 introduced a citation-matching approach to match the records using the foreign keys in the relational database using the probabilistic relational model (PRM). The problem with the rule-based matching approach is that it relies on the data quality in the dataset. The actual match will not be accurately generated if the dataset is incomplete or missing information.

Wang, Chen and Atabakhsh, 2004 proposed to compare four attributes of an individual, such as full name, date of birth, address, and social security number, for detecting identities and to combine the total similarity score. He introduced a record linkage algorithm for detecting deceptive identities. However, this approach is limited and cannot produce accurately matched results if one or more attributes are missing from

the dataset. To eliminate the duplicate records, (Culotta and McCallum, 2005) proposed another rules-based model called “the conditional random field model (CRF)” to measure the associations among other different entities. However, this approach failed and could not find the links between the same entity type (Li and Wang, 2015). One of the graph-based methods proposed by (Bhattacharya and Getoor, 2006) is that between each pair of reference entities, the relational graph matching is based on similarity with the same attribute that matches the similarity measure. Furthermore, (Bhattacharya and Getoor, 2007) enhanced the proposed approach by adding a collective entity resolution algorithm to match social information based on the already matched records to reuse for matching more records and not just for comparing two records.

Machine learning technique extracts the patterns from the training data rather than manually coding the rules to match the record in the data set. After extracting patterns from the record, the system creates its reference to match new records in the dataset (Li and Wang, 2015). To match user profiles from social media websites such as Facebook and Twitter, (Bartunov *et al.*, 2012) proposed combining user profiles using different attributes into a graph by detecting the social linkages between these two user profiles using a CRF-based approach. According to (Li and Wang, 2015), IBM’s InfoSphere Identity Insight is one of the best commercial applications for entity resolution and data analysis. The application uses a set of rules predefined in the system by humans to analyse the identities using sophisticated algorithms. If the two given identities have identical attributes, such as dates of birth and last names, and the threshold value is higher than the matching score, the system will combine them.

The main issue with the rules-based system is creating a set of rules, which is time-consuming and limited for specific attribute values. Therefore, different attributes within different datasets might not be able to be compared.

1.7.1. State of The Art - An Entity Resolution

Entity resolution is essential to identity resolution to increase data quality. Poor data quality is a consistent issue, especially in the policing system. Suppose errors are

introduced to the dataset when data enters the system, and the object is identified incorrectly. In this case, it may affect the final results significantly. Therefore, it is worthwhile to explore different entity resolution techniques and approaches. In simple words, entity resolution extracts the correct record from the data set while removing any record duplication to purify the result. According to (Kopcke and Rahm, 2010), entity resolution refers to record linkage, matching, reference reconciliation, and duplication identification. Identifying the real-world entity from the given entities is challenging due to the poor data quality. (Winkler, 1994) states that this problem was initially addressed by Newcombe in 1959, and later, in 1969, Fellegi and Sunter refined it to do entity matching in the structured data.

The main challenge in entity resolution is the mismatch in the data due to typographical errors and different variations of string appearing in the dataset. There are different methods have been developed to tackle such issues, and the main techniques are explained below:

1.7.1.1. Entity Resolution Techniques

- ***Character-based Similarity Metrics***

Edit distance is the distance between two strings calculated by applying insertion, deletion, or substitution operations to match one string to another. It is also called the Levenshtein distance, as Levenshtein introduced it in 1965 (Elmagarmid, Ipeirotis and Verykios, 2007). Suppose the distance of strings is less than a set threshold value by applying the three edit operations. In this case, the strings are considered a close match to each other with slight variation. The Edit distance matches the value of two attributes to see if they represent the same information for the same entity. (Needleman and Wunsch, 1970), They have slightly modified the edit distance by introducing different cost values for different edit operations on the string. While (Ristad and N.yianilos, 1998) introduced an algorithm that automatically calculates the cost for equivalent words written differently. However, Edit distance fails when strings are in short form or any

abbreviation is used instead of the complete word because the three edit operations will not calculate the cost correctly due to this issue.

Affine gap distance is described by (Elmagarmid, Ipeirotis and Verykios, 2007) as the gap between two strings that consider the abbreviated or shortened string calculation which edit distance cannot perform. It applies two additional edit operations, Open Gap and Extend Gap, to find the distance between two string values (Waterman, Smith and Beyer, 1976). The Open Gap determines the starting point in the string from where the gap should be inserted instead of inserting a character to convert an abbreviated string to another string. The Extend gap is used to extend the gap by adding one extra space in the string. The penalty of the extended gap is smaller than the open gap, so the affine gap of the two strings where one is an abbreviated variation of another string will be smaller than the original edit distance (Elmagarmid, Ipeirotis and Verykios, 2007). Therefore, (Bilenko *et al.*, 2003) introduced an algorithm to train an edit distance model using an affine gap technique. Suppose the two strings are written differently by exchanged character positions. In this case, the affine gap fails to calculate the cost and cannot match the strings. For example, the two strings “Belly Smith” and “Smith B” will have a significant cost value, which will assume that the two strings are not the same and will find a close similarity to produce the match for an entity.

The Smith-Waterman distance was introduced in 1981 to find the similarity of two strings by matching the substring. In this method, the two strings are aligned to calculate the substrings and their similarity by using the edit operations, and the two strings are compared (Smith and Waterman, 1981). There is a penalty for the alignment of the strings if any mismatch of characters is found. In contrast, any match of characters in the strings will generate the score for the two aligned strings. Smith-waterman has extended the edit distance and affine gap distance algorithms to find the similarity between two strings where the start and end of the strings get lower cost for mismatches while mismatches in the middle of the string get higher cost (Elmagarmid, Ipeirotis and Verykios, 2007). The algorithm runs in deep detail to find the match in substrings that show the similarity between two strings. However, computing the cost requires enormous processing power.

The Jaro distance was introduced to compare the first and last names (Elmagarmid, Ipeirotis and Verykios, 2007). The Jaro distance finds the number of familiar characters between two strings and tracks the order of the familiar characters (Gomaa and Fahmy, 2013). The Jaro distance algorithm was enhanced by (Winkler, 1990) in which the name prefix gets the weighting higher than the surname matches weighting. This variation of the Jaro distance is called the Jaro-Winkler distance (Gomaa and Fahmy, 2013). The Jaro and Jaro-Winkler distance algorithms cannot perform well where there is a positional difference between two strings and more than the allowed change. e.g., in the two strings “Alice bruce Bob” and “Bob bruce Alice”, the allowed positional change is six. However, the character `B’ in the string “Bob” has a 12-position difference. So only the string “Bruce” will match between two strings, and the algorithms will not find the better match (Elmagarmid, Ipeirotis and Verykios, 2007).

Q-gram distance is used for finding two strings’ similarity by calculating the similarity between the small substring sequences, and these sequences are referred to as q-grams (Barrón-Cedeño *et al.*, 2010). As described by (Ukkonen, 1992), the string is divided into a sequence of substrings having length q. For the string “Hello” where q = 2, the q-gram sequence will be `He’, `el’, `ll’, `lo’. The matching calculation is done on the q-grams sequences from two strings where the position and occurrence of sequences are unimportant. The strings will get a high score if both strings have the exact spelling or a close match. However, q-gram can lead to a false match by scoring high for two strings, “Chris” and “Rishi”, which will get a high score even if they differ. The higher value should be used for q-gram, i.e., q = 3 or more, to tackle such an issue. So, the sequence of q-gram will contain more characters for matching. Q-gram has different variants, such as trigrams, bigrams, and unigrams, for text matching and correcting spelling errors (Kukich, 1992). Sutinen and Tarhio, 1995 enhanced the q-gram algorithm even to record the position of the q-gram of the string, and it is called *positional q-gram* (Gravano, Ipeirotis, H. V. Jagadish, et al., 2001; Gravano, Ipeirotis, Hosagrahar Visvesvaraya Jagadish, et al., 2001) proposed to efficiently use the positional q-gram for locating similar or matching strings in the relational database. Nevertheless, any slight change in the strings can quickly decrease the matching accuracy of q-gram, and a low score can be given to strings.

➲ *Token-based similarity metrics*

An atomic string algorithm was proposed by (Monge and Elkan, 1996) to match text fields which contain alphanumeric characters enclosed within punctuation characters. The two strings only match if both are equal or if one string is the prefix of another. The two fields' similarity can be calculated by the number of matched atomic strings divided by the average number of atomic strings (Elmagarmid, Ipeirotis and Verykios, 2007). The two fields and strings are A = "Comput. Sci. & Eng. Dept., University of California, San Diego" and B = "Department of Computer Science, Univ. Calif., San Diego". So by excluding the stop sign (dot) from the strings and where k = 6, the first field string matches part of the second field string such as Comput., Sci., San, Diego, Univ. and Calif. However, no matches were found for the words Eng. and Dept. (Monge and Elkan, 1996).

The WHIRL system was introduced by (Cohen, 1998) that adopts from information retrieval the cosine similarity combined with the tf.idf weighting scheme to compute the similarity of two fields. The cosine similarity metric works well for many entries but is insensitive to the location of words, thus allowing natural word moves and swaps, e.g., "John Smith" is equivalent to "Smith, John". Also, the introduction of frequent words only minimally affects the similarity of the two strings due to the low idf weight of the frequent words. For example, "John Smith" and "Mr. John Smith" would have similarities close to one another. Unfortunately, this similarity metric does not capture word spelling errors, especially if they are pervasive and affect many of the words in the strings. For example, the strings "Compter Science Department" and "Deprtment of Computer Scence" will have zero similarity under this metric.

TF/IDF - Term Frequency or Inverse Document Frequency is a measure of speech and language processing discussed by (Cohen and Richman, 2002; Bilenko and Mooney, 2003) to determine the frequency of a string and to favour matches of less standard strings, penalising more common strings. This match requires knowledge or derivation of each attribute's frequencies. For example, when transcribing an address, it may be common for an "Avenue" to be misrecorded as a "Street." If so, the matching criteria may choose to ignore the most common words in this field, i.e., "Street," "Avenue," and

"Lane," instead of concentrating on the more critical number and name (Brizan and Tansel, 2006).

Q-gram with tf.idf was an extension to WHIRL and was proposed by (Gravano *et al.*, 2003) for handling spelling errors in the string using the q-grams technique rather than words. The two strings are less affected by spelling errors by using the sequence of q-gram, such as strings "Gteway Communications" and "Comunications Gateway" gets high similarity even if the word gets different arrangements in the string. Also, the word's insertion and deletion were handled using the q-gram sequences as the two strings "Gateway Communications" and "Communications Gateway International" are highly similar. In contrast, the word "International" would have a low weight due to its appearance.

1.7.1.2. Record De-duplication Approaches

The supervised learning approach requires a set of training data samples for the records to be labelled as matched or not matched. A CART algorithm was introduced by (Cochinwala *et al.*, 2001) by using the classification and regression trees based on a supervised approach, and the data is represented in the different relevant classes. (Cohen and Richman, 2002) introduced a system based on a supervised approach to cluster the records referring to the real-world entity by learning from the training data of the records. An adaptive distance function was used to learn from the training data and represent the records on the graph as nodes. Similarly, (Singla and Domingos, 2004) proposed to use the attribute values on the graph as nodes instead of the whole record as a node on the graph. Doing so allows the values to be transmitted to other nodes, and duplicate detection can be improved. Suppose the two records "Google, MountainView, CA" and "GoogleInc, MountainView, California" are equal. It means the words of the string "CA" and "California" will also be equal. The main issue with the supervised approach is acquiring a sufficiently large and representative training dataset. It might be easy to create the training data samples labelled as duplicates or non-duplicates, but to provide ambiguous record pairs for creating accurate results is

complex, and that is why data labelling is time-consuming (Elmagarmid, Ipeirotis and Verykios, 2007).

The active learning approach is used to overcome the problem of the supervised learning technique by automatically locating the ambiguous record pairs (Elmagarmid, Ipeirotis and Verykios, 2007) and mainly by reducing the training data samples (Christen, 2007). According to (Cohn, Atlas and Ladner, 1994), active learning controls the inputs on which it trains. Active learning differs from learning from examples due to the control over the data from which it learns and receives information. (Sarawagi and Bhamidipaty, 2002; Sarawagi *et al.*, 2002) introduced a system called ALIAS, which uses the reject region approach for record duplication detection to reduce the training data sample size, and the record pairs are presented as duplicate and non-duplicate. So, the record pairs are categorised as matched and non-matched and no manual labelling is required. However, if there are many ambiguous records, humans must label them manually. The training data sample provided to the system ALIAS with categorised labelled data such as matched and non-matched, and using this training data sample, ALIAS forms initial classifiers to match the data. By using the classifiers from a small training data sample, ALIAS distinguishes the records and finds duplicates in the data set. Similarly, (Tejada, Knoblock and Minton, 2002) propose to use the training data samples to set rules for matching the records in multiple fields using the decision trees. So, the active learning approach is suitable for producing better results. However, training data or human involvement requires training the system to produce the results. Nevertheless, the system is unsuitable because these resources are unavailable to generate the results.

The distance-based technique is one way of avoiding the need for training data to be tuned through training data. It is possible to match similar records without training Using the distance metric and an appropriate matching threshold. One approach is to treat a record as a long field and use one of the distance metrics to determine which records are similar. Distance-based approaches that conflate each record into one big field may ignore valuable information that can be used for duplicate detection (Elmagarmid, Ipeirotis and Verykios, 2007).

The rule-based approach is similar to the distance-based approach, and the distance of records is calculated as either 0 or 1. (Wang and Madnick, 1989) suggested using the

rules for the cases where there is no global key for a set of attributes to detect record duplication so the rules can combine the attributes as a set to form a key. Similarly, (Lim *et al.*, 1993) proposed a rule-based approach to have additional control to produce the correct results where rules provide correct information and have functional dependencies. So, the rules are not defined heuristically. This idea was further researched by (Hernandez and Stolfo, 1998), who proposed using logical suggestions for record matching and finding the similarity between records, e.g., suppose two individuals have name similarity and similar addresses. It will show that it is the same individual. However, manual tuning requires human effort, which is time-consuming and difficult for extensive data. So, such systems are used to generate the rules using the training data sample, and human experts manually adjust the generated rules based on the training data samples.

The unsupervised learning approach avoids manually labelling data by using the clustering technique and algorithms to group similar records for comparison corresponding to the same class. The probabilistic model, which was introduced in 1969 by Fellegi and Sunter, is the root of the unsupervised learning approach (Elmagarmid, Ipeirotis and Verykios, 2007; Bharambe, Jain and Jain, 2012). A similar concept was implemented by (Elfeky, Verykios and Elmagarmid, 2002), introducing a records duplicate detection toolbox called TAILOR. Using this toolbox, the extensive and new training data samples get better accuracy with the help of generated classifiers and less labelled data is required. While (Ravikumar and Cohen, 2004) suggested a similar approach for matching the records learned from the graphical model. The algorithm compares each field as a latent variable in binary format to show whether the target fields match.

The hybrid approach uses multiple similarities and records deduplication techniques collectively. (Elfeky, Verykios and Elmagarmid, 2002) , suggested overcoming the lack of training data sample issue by combining the supervised decision trees and unsupervised k-means clustering techniques where three clusters are used to produce results as matches, non-matches and possible matches. This whole process of record linkage is performed in two steps. In the first step, the weight vector subset is clustered as match, possible match, and non-match. In the second step, using training data, the matching

sample is generated by matches and non-matches clusters for a supervised classifier for each record pair. (Islam and Inkpen, 2008) presented a method named Semantic Text Similarity (STS) to use semantic and syntactic information to determine the similarity of two strings. In this method, string and semantic word similarity are two compulsory functions. In contrast, common-word order similarity is an optional function. Using this method for the 30-sentence dataset achieves an excellent Pearson correlation coefficient. According to (Buscaldi *et al.*, 2012), combining two modules results in a promising correlation between manual and automatic similarity. The first module uses N-gram to calculate the similarity between sentences, and the second module uses concept similarity measure and WordNet to calculate the similarity between concepts in the two sentences. (Bär *et al.*, 2012) introduced the system "UKP" to combine multiple text similarity techniques using a log-linear regression model from a training data sample. The multiple matching techniques used were string similarity, semantic similarity, text expansion mechanisms and measures related to structure and style.

1.7.1.3. Efficiency Improvement Techniques

Blocking methods pursue the simple idea of partitioning the set of tuples into different blocks and then comparing all pairs of tuples only within each block as this reduces the total number of comparisons of records (Baxter, Christen and Churches, 2003; Elmagarmid, Ipeirotis and Verykios, 2007; Draisbach and Naumann, 2011; Papadakis *et al.*, 2011). According to (Draisbach and Naumann, 2011), the central part of the blocking method is to have a better partitioning strategy to partition the records by partition number and size. The partitioning strategy should be able to partition the duplicate records in the same block, for example, using the whole or part of the postal code. So, if the duplicate records are partitioned in the same block based on the postal code, they are considered duplicates. The partitioning strategy can use attributes such as first or last name, name prefixes, and whole or part of address. So, in general, the partitions should generally be the same size. However, (Elmagarmid, Ipeirotis and Verykios, 2007) describe that this method is suitable to speed up the overall comparison of records. However, at the same time, it will bring false mismatches of records. If the records do not satisfy the blocking strategy, then the records do not appear in the same block.

Therefore, the records that are supposed to be in the same block but, due to blocking strategy, appear in the wrong block instead of in the same block, which will cause missed matches. However, multiple runs should be performed to overcome this issue using a different blocking strategy for each run (Draisbach and Naumann, 2011).

So, using a different field for each run as a blocking strategy can improve the record matching in the block and reduce the chances of false mismatches, which helps to detect duplicate records having different partitioning attributes (Elmagarmid, Ipeirotis and Verykios, 2007). sorting requires a block key to implement the blocking for record duplication detection. The block key can be created by combining single or multiple fields, such as the age attribute, combined with the postcode attribute to form the key. So, records that satisfy the block critical criteria are placed into the same block (Elfeky, Verykios and Elmagarmid, 2002; Baxter, Christen and Churches, 2003). However, the blocking process can lead to many record pairs generated by the massive number of records. It can affect the blocking comparison of records, such as blocking keys based on the gender attribute. In that case, it will generate large blocks matching the key. However, the records duplication detection will miss records of the blocks that are generated by the key that are too small. According to (Baxter, Christen and Churches, 2003), different factors can affect the record comparison process in blocking, such as spelling errors or missing values, which can cause records not to appear in the same blocks. However, this can be alleviated by using the multiple blocking keys with multiple passes to improve the comparison. However, this tuning can be a difficult task. (Papadakis *et al.*, 2011) They introduced a method based on the two layers wherein the unnecessary record comparisons are excluded from the second layer. In this method, the first layer is used to block effectively by placing all records in the same block with the token in all records as the attribute value. So, the resulting records are duplicates and do not have any chance of no blocks in common. The comparison is made on the second layer with the help of different methods to increase the blocking efficiency by reducing unnecessary records. Records with low threshold values are not required to be compared, and this will reduce the number of comparisons in the blocks.

Windowing methods are more complex than blocking methods. The famous windowing method is the sorted neighbourhood method (SNM). (Hernandez and Stolfo, 1998)

proposed using a fixed window size to place over the records sorted by a sorting key. So, any records placed under the fixed-size window will be paired. The SNM method has three phases during the whole comparison process. The sorting key is identified in the first phase for records by combining the different attribute values, and it does not need to be unique. After this, all the records are sorted based on the sorting key. In the third phase, all the records are compared and paired by moving the fixed-size window on the sorted records. The first two phases of windowing are similar to the blocking technique. The windowing method reduces the number of record comparisons due to the window limit (Baxter, Christen and Churches, 2003). However, duplicate detection improves due to the window size, usually between 10 and 20. Increasing the window size can result in more duplicate detection but can slow the processing of the results (Draisbach and Naumann, 2011). The records are compared in this method only if they fall within the window size and the sorting key to sort them accordingly. So, a single key is insufficient to sort the records and place them under the fixed-size window for comparison. If there is any error in the records attribute values or any missing value, it will affect the whole comparison and duplicate detection process (Elmagarmid, Ipeirotis and Verykios, 2007).

These errors can be alleviated by using multiple attributes to form the sorting key and multi-pass to sort and compare the records under the fixed-size window (Draisbach and Naumann, 2011). It increases the possibility of more record duplicate detection by running multiple passes using different sorting keys compared to a single pass comparison, as it will miss the matched records (Baxter, Christen and Churches, 2003). (Hernandez and Stolfo, 1998) introduced a multi-pass strategy using the sorted-neighbourhood method to compare different sorting keys with small window sizes each time. In this multi-pass method, each run with different sorting keys creates a pair of records and can merge them during the comparison process under the fixed window size to produce the result (Elmagarmid, Ipeirotis and Verykios, 2007). The main issue with the sorted neighbourhood method is that not all records will be compared if the window size is smaller than the number of matched records using the sorting key, as the records will not fit under the fixed Window size. It will miss the records during the comparison. Suppose we use the surname ‘Smith’ as a sorting key; it might produce a vast number of record pairs where not all record pairs will not fit under the window size

and will miss the rest of the pairs outside the window and will not be compared (Baxter, Christen and Churches, 2003).

Clustering is an essential technique in data mining in which a group of data objects is taken as input (Bezdek, 1981). In this technique, several clusters are obtained as an output so that the objects in the same group or cluster are similar but are different to objects outside the cluster (Jain, Murty and Flynn, 1999; Halkidi, Batistakis and Vazirgiannis, 2001; Dharmarajan and Velmurugan, 2013; Nisha and Kaur, 2015). The representation of objects is the main feature of clustering, as objects are represented as patterns to find the similarity (Filippone *et al.*, 2008). The patterns are considered similar if they are in the same cluster but considered different if not in the same cluster. So, this difference should be clear and meaningful to represent patterns in the cluster (Xu and Wunsch, 2005). (Monge and Elkan, 1997) improved the nested-loop record comparison performance by showing a transitive approach for duplicate detection. For example, if 'A' and 'B' are duplicates and 'B' and 'C' are duplicates, then it is assumed that 'A' and 'C' are also duplicates. The problem with this approach is that record matching relies on the dependency of connected components of the graph. If these connected components are, then the assumption can be valid; otherwise, no relationship can be retrieved between records. So, to compute the connected components efficiently on the graph, (Monge and Elkan, 1997) used a union-find structure. The records are combined into a cluster during the union stage. The cluster is used as a comparison representation, and the number of record comparisons gets reduced for duplicate detection. So, in simple words, if 'A' is not a duplicate of 'B' already in the cluster, then other members in the cluster will not be duplicates of 'A'.

The canopies technique was introduced by (McCallum, Nigam and Ungar, 2000) to improve the speed of record duplicate detection. In this technique, records are grouped from clusters overlapping each other, and this overlapped area is named canopies. The records are grouped using the pairwise comparison with a similarity metric for better results. Let us supper if the two strings have a length difference of more than "3", so the edit distance of these strings cannot be less than "3". So, the string length is used for comparison as a canopy function for the edit distance function. (Gravano, Ipeirotis, H. V. Jagadish, *et al.*, 2001) suggest using the length of strings with the q-gram of strings as

canopies for the edit distance metric. The advantage of this technique is the use of vanilla SQL statements, which can be used to do canopy function calculations. (Cohen and Richman, 2002) suggest using tf.idf similarity metric as a canopy distance with other multiple similarity metrics duplicate record detection. Similarly, (Chaudhuri *et al.*, 2003) introduced a canopy function with indexing to match similar records for duplicate detection. On the other hand, (Baxter, Christen and Churches, 2003) showed that using the traditional blocking methods with the canopy technique improves the record duplicate detection speed and quality.

1.8. Research Aims

A framework for identity resolution is essential for incorporating intelligence in matching raw data. It should employ a suitable algorithm with pattern recognition capabilities that closely resemble the functioning of the human brain.

- How will the desired identity be extracted from the raw data set?
- How will records be matched to extract meaningful information from the raw data set?
- To what extent can establishing relationships between diverse identities be enhanced through applying pattern recognition techniques?

1.9. Problem Definition

In the entity resolution process, the record de-duplication makes it challenging to identify duplicates due to the different fundamental values used for the duplicate record. To find the duplicates in the dataset, it is a must to compare one record with the rest of the records in the database to find the similarities, but this will require immense processing power in the case of the large dataset (Wangikar, Deshmukh and Bhirud, 2016). All techniques discussed in the literature review are insufficiently adaptive regarding record de-duplication and entity matching based on the correct record

extraction. It is clear from the literature review that techniques that require human expert tuning are better but are unfeasible for large datasets due to the manual tuning required. Also, such techniques require some training data samples to generate the results. However, it is not easy to provide training samples for every situation. So, considering this, the approach is also unfeasible for better record matching. Another issue with the proposed techniques is that they do not use all the available similarity metrics as required and only utilise a couple of techniques to complete the record-matching process. It leads to unsatisfactory results as every similarity metric is domain-specific to solve a particular problem, and missing one or more metrics would not help achieve better results. From a comprehensive literature review that has been carried out, it can be concluded that so far, there is no unsupervised approach framework that can achieve the following:

- Automatically adjust to tune the data based on the input, using different similarity metrics to extract the records.
- After the data cleaning, automatically adjust record matching techniques without training data samples for record de-duplication.
- Run the similarity metric on data at different stages to output the best results for record linkage and relationship analyses.
- Use the clustering technique with the help of segmentation for identity resolution.

So, in the research, an adaptive hybrid approach will be introduced to automatically self-tune the similarity metrics using fuzzy logic. This searching process will be iterative (multi-pass searching) using an unsupervised clustering approach to analyse the output by further segmenting the records to show the relationship in a graph and extract the true identity. In this process, segmentation will be the process of putting data into groups based on similarities. At the same time, clustering will be the process of finding similarities in data to be grouped. Once the entity is resolved, the system will keep the result as a reference for the future, which will help enhance future search efficiency by

matching the same or similar record as the new search will also be matched with references as part of the iterative search process.

1.10. Thesis Chapters Overview

This chapter presents an overview of fraud, identity, and the different types of identity crimes faced by law enforcement agencies or financial institutions. This chapter briefly explores machine learning types, different machine learning techniques, and fundamental concepts related to matching records. Furthermore, the chapter provides an overview of the literature on string and record-matching techniques related to entity matching and identity resolution, research aims and the definition of the problem.

This thesis comprises five chapters, starting with "*Chapter 1* - introduction and literature review" discussed above and ending with the research aim and problem definition for this research. The remaining chapters in this thesis are as follows:

Chapter 2 provides the platform to discuss machine learning techniques, focusing mainly on fuzzy matching techniques. The chapter explains entity matching and its challenges when matching an entity for identity resolution. The chapter discusses matching techniques to handle entity matching challenges and how knowledge can help in the matching process.

Chapter 3 focuses on the methodology used in this research. The chapter explains the proposed algorithm and provides the foundation of the research methods. The later section of the chapter explores the proposed algorithm and its implementation by enhancing the string-matching technique. The details of different tools used for implementation are discussed to understand how they are used to generate matching results. These results provide a better understanding of the implemented string-matching technique in the proposed algorithm.

Chapter 4 provides the results of the implementation of the proposed algorithm. The chapter explains and analyses the policing dataset used in the research and presents the computer simulation results. It starts with evaluating and analysing the performance of different sets of names from different languages with a proposed algorithm for matching

names. Later in the chapter, the results of the policing dataset are evaluated in detail, and the performance of the generated results is discussed.

Chapter 5 concludes this research thesis by providing an overview of the research and its implementation. It explains this research's limitations, knowledge contribution, and areas where it can be beneficial with further improvements. The last section of the chapter discusses future work and provides suggestions for improvements to enhance the proposed algorithm.

The last sections of the thesis are References used in the research. Appendix A lists the published research paper, and Appendix B provides additional details on the tools used in this research.

CHAPTER 2

2. STATE-OF-THE-ART REVIEW OF MACHINE LEARNING APPROACHES FOR IDENTITY RESOLUTION

2.1. Chapter Overview

This chapter discusses different machine-learning techniques that can be used for identity resolution, and at the end, it discusses the problem definition and the research aims. It starts with a background discussion of entity resolution and the challenges in detail. Based on the challenges an entity matching faces, the string matching techniques will be discussed to understand the issues in the matching techniques. The issues with each matching technique will be presented, including all the comparisons. The clustering technique and types for grouping similar entities will be discussed. Finally, the chapter will discuss the knowledge base and the different types that can be used in machine learning.

2.2. Background

Entity Resolution (ER) is essential as it leads to identity resolution. Usually, entity and identity resolutions are used interchangeably and referred to as the same. However, these are two terms with different requirements. It is evident from the previous studies that each string-matching technique is domain-specific and cannot perform well independently to find the required results. Using only one technique is unsuitable for tackling a particular issue in entity resolution. However, a combination of these matching techniques can produce the desired results. We know that entity resolution is an essential step toward identity resolution, and incorrect or missing details in entity resolution will lead to incorrect or weak identity resolution.

2.3. Entity Resolution and Challenges

We know Entity Resolution (ER) is to identify one entity in a dataset that refers to the actual entity. Typically, a database may contain multiple entities, and there can be relationships among these entities. It is vital to correctly match the string of names, addresses, or other information to resolve any entity (Altowim, Kalashnikov and Mehrotra, 2018). Each entity has attributes and can be identified using crucial attributes such as ID number, name, and date of birth. However, many other reasons make it challenging to find the correct identity (Phillips, Amirhosseini and Kazemian, 2020).

For entity resolution, there are many different string-matching techniques. In later sections, the three most essential matching techniques will be discussed in this chapter. However, before that, there are several different challenges that entity matching faces in comparing strings and getting the desired results. Some of the challenges are listed below.

2.3.1. Text Standardisation and Name Variations

Text standardisation involves replacing different spellings in words with single and correct spellings. For example, 'Incorporated' can be represented with the standardized spelling 'Inc.' Standardisation typically separates the string into words, such as a complete name or address. After this, each word is compared in the standard table to get the standard spelling. However, it is difficult to manage such a standard list of spellings to match the words against it and bring them into the standard in the dataset (Winkler, 2006). Names matching has been troublesome for record linkage as names can have variations. The variation can be phonetic or alternate spellings; in some cases, it can be a combination of both variations (Snae, 2007). In different computer applications and record linkage algorithms, spelling variations can be allowed, and the algorithm's success is determined by identifying the differences in name spelling. However, in some cases, it is difficult to determine the name variation due to the different spellings of the same name or to consider it entirely different. Usually, surnames are more variable in spelling and can have many common alternatives.

In some cases, the variations are due to spelling errors. Such a range of name variations is a big concern due to different naming conventions and languages, so how do we identify a person based on a name if it is differently spelt or pronounced? It is a significant problem identifying a person and determining if the name variation is the spelling of the same or a different person's name. Most of these variations can be categorised as follows:

2.3.1.1. Spelling Variations

These variations are due to typographical errors by a human operator or an automated device designed for inputting data into the database. It can lead to a problem matching names in the database. However, typically, such variations do not affect the name's pronunciation (A. J. Lait and Randell, 1996). The variations can be misplaced letters, substituted letters, adding letters, or omissions of letters. These variations cause issues for matching algorithms to match strings to one another even though the string is phonetically not changed (Shah and Kumar Singh, 2014). According to (Pilania and Kumaran, 2019), a customised matching algorithm can be used to match names with variations. Many spelling error correction solutions have been designed to encounter pattern-matching issues. According to studies conducted by different researchers, these spelling errors can be divided into two categories.

- *Typographic errors* are spelling errors primarily due to keyboard inputs when a string is mistyped. However, the actual spelling of the string is known.

This type of error falls into four categories, as suggested (Naseem and Hussain, 2007) and according to (A. J. Lait and Randell, 1996; Naseem and Hussain, 2007), the spelling error or mistyping of the names can be categorized as below:

- Insertion or additional letters, e.g. "MCMANUS and MACMANUS",
- Deletion or omission, e.g. "ROBBIN and ROBIN", "Collins and Colins"
- Substitution, e.g. "SMYTH as SMITH"
- Transposition, e.g. "BREADLEY and BRAEDLEY"

These editing operations errors can be referred to as single errors. According to (Naseem and Hussain, 2007), it was confirmed by researchers at a later stage.

The most common typographic errors are substitution errors caused by keyboard inputs. It usually happens during typing due to incorrectly replacing letters by not pressing the correct "key letter" required in the string.

There are other errors called multi-errors that are generated due to more than one editing operation. These are either the addition of two extra letters, missing two letters from the string, or transposing two letters.

- *Cognitive errors* are caused by keyboard input during typing, and the actual spelling of the string is unknown. In cognitive errors, the mistyped word pronunciation does not change as the word pronunciation remains the same or similar to the correct spellings (Naseem and Hussain, 2007). e.g. "recieve" and "receive", "abyss" and "abiss"

2.3.1.2. Phonetic Variations

According to (Snae, 2007; A. J. Lait and Randell, 1996), these variations are caused by mishearing the name, which alters the name structure utterly different from the actual name. For example, "Pooh" is an English nickname while it would be spelt as "Puh" in German. Similarly, the names "MAXIME and MAXIMIEN" and "Sinclair and St. Clair" have different phonetic structures, and the names are significantly changed; however, these are related names (Shah and Kumar Singh, 2014). Sometimes, the name's phonetic variation can be hugely different, such as "ADELINE and LINE", "Christina and Tina", and that is where the name is shortened (A. J. Lait and Randell, 1996).

2.3.1.3. Character Variation

These variations caused problems due to abbreviations, capitalization, punctuation, qualifiers and namespacing. The capitalization in the name where the upper and lower letters have been used, such as "brown and Brown" and "SMITH and Smith". Sometimes,

first names or surnames are composed of two parts and contain punctuations, spaces and qualifiers such as “WILL SMITH and WILL-SMITH” or “SMIT and S.M.I.T”, “YOUNGSMITH and YOUNG SMITH”, “WILL SMITH and WILL SMITH YOUNG” or “Philips-Martin”. These full names do not need to be used as they are; instead, one part of the name may be used. For example, “Philips” or “Martin” may be used instead of “Philps-Martin”. Such variations can be called *Double Names* (A. J. Lait and Randell, 1996; Shah and Kumar Singh, 2014; Pilania and Kumaran, 2019). *Double first names* are not common in English, but other languages can contain double first names. For example, in the French language, the name “Jean-Claude” may be written in full or as “Jean” or “Claude” (A. J. Lait and Randell, 1996).

The names can also have *abbreviations* where the name has been shortened, and this mainly names the individual. Such as “ROB and ROBBIN” and “BOB and BOBBY” while “WILL SMITH” is written as “W SMITH” (A. J. Lait and Randell, 1996; Pilania and Kumaran, 2019). Sometimes, an individual prefers to change their name in future from the one he/she was known to in the past or change the surname to their partner's name. Such situations cause a big issue in matching names for an individual. Name-matching algorithms that use spellings or phonetic variation to match names will not be able to identify the person with past and future names (Snae, 2007; Shah and Kumar Singh, 2014).

2.3.1.4. Fielding Variations

There may be a situation where multi-part names have been added to the database in a different order. It can happen due to different cultural names. For example, in some cultures, the name is written in the format First-Middle-Last, while others may write it as Last-First-Middle. The name “Will Young Smith” may be used as “First-Middle-Last”, but possibly it may be used in the database as a “First-Last” name, e.g. “Will Smith” while “Middle” name as “Young” used differently (Pilania and Kumaran, 2019).

2.4. Approximate String Matching

The data cannot always be easily identifiable through unique identifiers. If the object is identified incorrectly, it will hugely affect the results. Approximate matching is also called fuzzy matching, where each matching value can be between 0 and 1 but not just 0 or 1. The matching involves a comparison of substrings with the given string to find the similarity between two strings. In the entity resolution, the records are compared by matching the string similarity. Based on the similarities, records are classified into match and non-match categories. This matching of records takes place on the value of the attributes to find the similarities (Christen, Vatsalan and Wang, 2016). Approximate matching is suitable for handling issues such as typographical errors in the string as it closely matches the search pattern (Al-khamaiseh and Alshagarin, 2014). However, (Winkler, 2006; Naseem and Hussain, 2007) state that this area is still under research and requires suitable matching techniques. Fuzzy matching is generally used for pattern matching by calculating an estimated match between two strings. Approximate matching is required for a name in an extensive database where names are misspelt rather than the correct spelling. It makes an exact match difficult in large databases; therefore, an approximate can be essential in pattern matching (Shah and Kumar Singh, 2014). Pattern matching, phonetic encoding, or Lexographic matching techniques are the way to do string matching. Pattern matching is to calculate the distance between each character of the string, and phonetic matching converts a string into code based on the pronunciation of each string. At the same time, the Lexicographic technique produces all possible variations of the string (Shah and Kumar Singh, 2014; Pilania and Kumaran, 2019).

The following section will explore how pattern matching and phonetic encoding algorithms work for matching names.

2.4.1. Edit Distance Algorithm

Edit distance is a measure of string comparison (Shah and Kumar Singh, 2014). Approximate matching is about calculating the distance between each string, and the commonly known technique is “Edit distance” (Al-khamaiseh and Alshagarin, 2014). It calculates the distance between two strings by applying three operations on the strings. Usually, three joint operations are performed on the string to find the similarities or convert one string into another. These operations are called edit operations, in which inserting, deleting, or substituting the characters in the strings take place to match strings. Edit distance, also called Levenshtein distance, was introduced in 1965 (Elmagarmid, Ipeirotis and Verykios, 2007). According to (Soundex, 2015), if the distance of strings is less than a set threshold value by applying the three edit operations, the strings are considered a close match to each other with slight variation, or it can be said that the more the Levenshtein distance, the distinct the strings will be. In substitution, one character replaces the other character in the string. In contrast, in deletion, a character gets removed from the string. In insertion, a character gets added to the string (Elmagarmid, Ipeirotis and Verykios, 2007). The distance 0 means the strings are identical. These edit operations give cost to the number of required iterations to match the name. i.e., the cost of deleting a character, inserting a character and substituting two different characters is “1”; otherwise, the cost value is “0”. According to (Hasan and Ahmed, 2015), the edit distance where each operation on the single string has the cost = 1, then such edit distance is called unit-cost edit distance. (Needleman and Wunsch, 1970) , modified the edit distance by introducing different cost values for different edit operations on the string. (Ristad and N.yianilos, 1998) introduced an algorithm for automatically calculating the cost for equivalent words written differently. For example, (Shah and Kumar Singh, 2014) describe the Levenshtein distance between "bitten" and "sitting" as “3” since the following three edits change one into the other. There is no way to do it with fewer than three edits, e.g.

Step 1: bitten → sitten (substitution of "s" for "b")

Step 2: sitten → sittin (substitution of "i" for "e")

Step 3: sittin → sitting (insertion of "g" at the end)

Another example of calculating Edit distance to compare the name “ALEXANDRE” to “ALEKSANDER” is 4. To do this (Soundex, 2015):

Step 1: substitute X with K

Step 2: insert S after K

Step 3: insert E after D

Step 4: delete E at the end

(Balabantaray *et al.*, 2012) state that “the Levenshtein algorithm only edits operations between two strings, and it does not directly provide a knowledge base to identify phonetic similarity among the languages that appeared in different phonemes. However, several research studies have been conducted by assigning different costs for operations to integrate the knowledge base concept to the Levenshtein algorithm”.

2.4.1.1. Issues in Levenshtein Edit Distance Algorithm

The Edit Distance algorithms perform several iterations to change one string into another to find the similarity. If the two strings are similar, then the cost of operation is “0”; otherwise, it will be “1” based on the iteration operations applied to the strings. However, implementing the Levenshtein Edit distance algorithm to match two strings encounters the following issues.

Edit distance fails when strings are written in short form, or any abbreviation has been used instead of the complete word. In this situation, the three edit operations will not calculate the cost correctly due to this issue. It is a significant drawback of the Edit Distance algorithm while finding the match between strings. Table 2.1 is taken from (Pilania and Kumaran, 2019), which shows an example of matching two strings by matching string 1 with string 2. The two strings are different and show an incorrect match.

Table 2.1 - Leveshtien Distance Algorithm Issue (Pilania and Kumaran, 2019)

String1	String2
Uma	Keshk**uma**ri
Kunta	Sha**kunta**la Devi
Kusum	**Kusum**alta
Tara	Si**tara**m
Indra	**Indra**kala

The Levenshtein Edit distance matches strings by comparing each character in the strings, so matching strings in Table 2.1 shows that string 'UMA' is found in string 'KESHK**UMA**RI'. Therefore, no edit operations are applied, and the cost of operation is 0, which shows it is a match, nevertheless these two strings differ entirely (Pilania and Kumaran, 2019).

2.4.2. Jaro-Winkler Algorithm

The Jaro distance technique is usually used to tackle typical spelling errors and was introduced by M. A. Jaro in 1989 (Jaro, 1989). This technique was designed to compare two short strings, such as first and last names (Elmagarmid, Ipeirotis and Verykios, 2007). The Jaro distance finds the familiar characters between strings while tracking the order of these characters (Gomaa and Fahmy, 2013). The Jaro distance can be computed using the following formula:

$$\boldsymbol{simjaro}(\boldsymbol{s1}, \boldsymbol{s2}) = \frac{\mathbf{1}}{\mathbf{3}} \left(\frac{\boldsymbol{c}}{|\boldsymbol{s1}|} + \frac{\boldsymbol{c}}{|\boldsymbol{s2}|} + \frac{\boldsymbol{c}-\boldsymbol{t}}{\boldsymbol{c}} \right) \qquad \textbf{Equation 2.1}$$

The Jaro similarity technique finds the matched characters in the given names. In equation 2.1, 'c' represents the number of equal characters, 't' represents half the transpositions, |s1| is the length of the first string, |s2| is the length of the second

string. The interchanging of the contiguous characters in both strings, such as 'pe' and 'ep', is called the transposition (Sun, 2015).

Initially, the Jaro-Winkler algorithm following the UNIMATCH similarity was developed in the 1970s to be used in the post-enumeration analysis by the U.S. Census Bureau. It was used to match records of different fields with uncertain name spellings. UNIMATCH matches two strings based on string length, familiar characters in strings and the number of character transpositions (Kloo, Dabkowski and Huddleston, 2019).

Another approximate matching technique is the Jaro-Winkler, in which Jaro compares the short strings, mainly names, by computing the length of the string, finding similar characters, and determining the transpositions required. In 1990, Winkler proposed a variation to the Jaro distance, which gives preference to the name prefix (Winkler, 1990, 1994, 2006). The name prefix is assigned a weight higher than the surname match weight. This Jaro distance variation is called the Jaro-Winkler distance (Gomaa and Fahmy, 2013).

$$Jaro\text{-}Winkler(s1, s2) = \text{Jaro}(s1, s2) + l * p(1 - \text{Jaro}(s1, s2)) \qquad \textbf{Equation 2.2}$$

Here, '*l*' is the longest common prefix of the two strings, where '*p*' is a scaling factor variable to adjust the matching score upwards by customarily set to a value of 0.1 and the maximum value set to 0.25; otherwise, the distance will be computed as more significant than the value 1 (Sun, 2015; Pilania and Kumaran, 2019). This enhancement helps to get a few penalties for errors like keyboard errors and errors at the end of a string (Soundex, 2015). The matching of strings is scaled between values 0 and 1, where any calculated score of two strings close to 0 represents no match, while the calculated score value close to value 1 represents a match (Pilania and Kumaran, 2019). For example, as stated (Soundex, 2015), the Jaro similarity between "ALEXANDRE" and "ALEKSANDER" is 0.85, calculated as below using equation 2.1:

During this process, match A, L, E, A, N, D, R, and E with a "1" transposition.

Simjaro = (8 / 9 + 8 / 10 + (8 – 1) / 8) / 3

= 0.85

The Jaro-Winkler similarity score of 0.90 is calculated using equation 2.2 as below:

Jaro-Winkler = 0.85 + 3 * 0.1 (1 - 0.85)

= 0.90

2.4.2.1. Issues in the Jaro-Winkler Algorithm

Jaro and Jaro-Winkler distance algorithms cannot perform well where there is a positional difference between two strings and more than the allowed change. e.g. in the strings “Alice bruce Bob” and “Bob bruce Alice”, the allowed change is six positions. However, the character ’B’ in the string “Bob” has a difference of twelve positions. So only the string “Bruce” will match between two strings, and the algorithms will not find the better match (Elmagarmid, Ipeirotis and Verykios, 2007).

The Jaro-Winkler algorithm will fail to match short names with typographical errors if other strings of different lengths have the same errors (Kloo, Dabkowski and Huddleston, 2019).

2.5. Phonetic Matching

Phonetic matching plays a significant role in matching strings. (Snae, 2007; Shah and Kumar Singh, 2014) defines phonetic matching as identifying a set of strings most likely to be similar in sound to a given keyword. The strings can be spelt using different writing styles but can be matched phonetically (Christopher Jaisunder, Ahmed and Mishra, 2017). The phonetic algorithm matching compares names with similar sounds, even those with different spellings, and it is vital for matching the names from the database. According to (A. Lait and Randell, 1996; Christopher Jaisunder, Ahmed and Mishra, 2017), a phonetic algorithm is an algorithm for indexing words by pronunciation. The complex algorithms have many different rules and exceptions due to the English

language's historical changes in pronunciation and the integration of many different language words into it. Phonetic matching evaluates the similarity of the names based on how they are pronounced without looking at the actual spelling of the names. Different researchers have carried-out many pieces of research to explore data mining methods in information retrieval. The most common phonetic matching technique is the Soundex technique, which compares and matches names based on pronunciation (Shah and Kumar Singh, 2014; Christopher Jaisunder, Ahmed and Mishra, 2017).

In Soundex, the code value is generated for similar-sounding names and the code is compared to find a match. The values are called Soundex encodings. Therefore, any search on any name in the database gets Soundex encoding rather than a direct name search. Any name match based on Soundex encoding will be retrieved from the database. According to (A. J. Lait and Randell, 1996), any matches found during a search are called positives, and the ones rejected by the search are called negatives. So, the positives relevant to the search term are called true positives, while the other searches are false positives. Searching for names in an extensive database has always been a problem. In a large database, misspelt names may be spelt differently due to human typing errors or mishearing the name. In such a situation, a fuzzy match will be enough to match the names instead of matching names exactly.

2.5.1. Soundex Algorithm

The Soundex algorithm was developed by Odell and Russell in 1918. It is a code-based matching algorithm that converts the name into a 4-letter alphanumeric code based on the sound of each word while preserving the first letter from the name (A. J. Lait and Randell, 1996). The similar-sounding names get similar code values to match the names. These codes are called Soundex encodings (Koneru, Pulla and Varol, 2016). The algorithm does not search the name directly from the database. However, it converts into Soundex encoding of 4-letter code, keeping the first letter preserved and then performing a search based on this encoding (Shah and Kumar Singh, 2014; Christopher Jaisunder, Ahmed and Mishra, 2017).

The following steps are required to create the Soundex code:

Step 1 - Retain the first letter from the name string.

Step 2 - Convert all occurrences of these letters to Zero: a, e, h, i, o, u, w, y.

Step 3 - Convert the remaining letters into numbers: b, f, p, v = 1; c, g, j, k, q, s, x, z = 2; d, t = 3; l = 4; m, n =5; r = 6

Step 4 - Remove all pairs of digits with the same code adjacent to the original name.

Step 5 - Remove all the zeros.

Step 6 - Return the code with four characters and add zeroes to the right if there are fewer than four characters.

In Step 1, to generate the Soundex code for any given name or string, the letter will be kept as the first character of the final Soundex code. It represents the initial sound of the word. After the first letter, in Step 2, all vowels “A, E, I, O, U” will be ignored. The letters “H, W, Y” are considered insignificant because these vowels and letters often do not affect the word's pronunciation meaningfully, particularly when in the middle or end of words. After this, in Step 3, each of the remaining consonants will be converted into a number according to Table 2.2. The idea behind this step is that similar-sounding consonants are grouped under the same number, reducing the variations caused by different spellings. Step 4 ensures that repeated sounds do not overly influence the code. That is why if two or more letters that convert to the same number appear next to each other, only keep the first one and remove the duplicates. Step 5 is to use the first four characters and discard the rest if there are more than four characters in the forming code. In Step 6, the first letter kept in Step 1 with the numbers from Step 3 is combined to result in a four-character alphanumeric code. If there are fewer than four characters in the code, add zeros (0) until the code is four characters long. This padding ensures that all Soundex codes are the same length, making them easy to compare. The standard Soundex algorithm defines the following groups:

Table 2.2 - Grouping of Letters and Code Assignments (Patman and Shaefer, 2001)

Letters	Code Assignment
B, F, P, V	1
C, G, J, K, Q, S, X, Z	2
D, T	3
L	4
M, N	5
R	6
H, Y, W	(omitted)
A, E, I, O, U	(omitted)

In converting a name to a Soundex code, only one digit is used for any consecutive letters, e.g. “CK” will not be assigned a code as 22 but will be “CK” = 2. Any code with more than three digits will be truncated, and any code with fewer than three consonants will be padded by zero, e.g. the string “PEEL” will get code as “P400”.

For example, using the Table 2.2, the Soundex code of the name “ALEXANDRE” is:

Step 1: A4E2A536E

Step 2: A4E2A536E

Step 3: A42536

Step 4: A425

Another example of Soundex code for the name “ALEKSANDER”. Here, it is assumed the name is misspelt:

Step 1: A4E22A53E6

Step 2: A4E2A53E6

Step 3: A42536

Step 4: A425

The Soundex code of both names is “A425”; thus, both names are easily comparable and matched (Soundex, 2015).

For the Soundex algorithm, it is worth knowing how each word is pronounced based on the sound of each English language alphabet.

The following Table 2.3 briefly describes this:

Table 2.3 - Phonetic description based on assignments of letters (Magazine, 2018)

Number	Represents the Letter	Phonetic Description
1	B, F, P, V	Labial sounds and labiodental (require particular use of the lips).
2	C, G, J, K, Q, S, X, Z	Guttural sounds (produced in the throat) and sibilant sounds (requires a hissing noise).
3	D, T	Dental-mute sounds (formed with the tip of the tongue against the teeth).
4	L	Palatal fricative or long liquid sound (produced by an extended contact of the tongue and mouth).
5	M, N	Nasal sounds (produced partly through the nose). M is labio-nasal and N is dental or lingua-nasal.
6	R	Dental fricative or short liquid sound (produced by a slight contact of the tongue and mouth).
Discard	H, W, Y	Disregard consonants H and W.
Discard	A, E, I, O, U	Disregard vowels.

The Soundex technique is commonly used for the English language. However, researchers have also modified it for other languages (Balabantaray *et al.*, 2012). The Soundex algorithm matches names that sound similar but have different spellings, e.g., “SMITH” and “SMYTH”. The names are given a phonetic code that helps to match names and reduces the issue of mistyped names (Soundex, 2015). For example, “SMITH” and “SMYTH” will get code as “S5030”, and it will be refined to a shorter code based on Soundex rules as “S530” (Christopher Jaisunder, Ahmed and Mishra, 2017). According to (Patman and Shaefer, 2001), “The computational benefits of Soundex-type algorithms are easy to see by exchanging a name for a code; all variant spellings of the name can be expected to share that same code, allowing a relatively efficient search of a small

subset of a database, versus "brute force" evaluation of every name as a potential match. Soundex keys are typically used to form an index for data implemented in relations DBMS products, allowing very fast key-based retrievals of a (theoretically) a small number of potentially matching names."

2.5.1.1. Issues of Soundex Algorithm

There are issues with the Soundex code that limit the matching of names. One of the disadvantages of the Soundex technique is that it preserves the first letter of the name and converts the rest into a code. Therefore, a name starting with a different letter will result in a different Soundex code and the match will be unsuccessful. For example, the Soundex code for "PATER" and "PAIDER" will be P360. The string "SOMERS" Soundex code S562 will match with "SUMMERS" and "SOMMARS" as the Soundex code for these is S562. It shows that the Soundex code is assigned based on the pronunciations regardless of the spellings, which can result in false matches (Patman and Shaefer, 2001).

- ***First Letter dependency*** - According to (Patman and Shaefer, 2001a; Shah and Kumar Singh, 2014), retaining the initial letter is vital for Soundex, and any name that starts with a different letter will create a mismatch because it will be considered a different Soundex code. For example, the names "Corth" and "Korth" will not match even if one of the names is in the database. It means that starting with a letter in the Soundex code is essential, and names starting with different letters will never match (Pilania and Kumaran, 2019).

- ***Transcription difference*** - The Soundex codes for Roman and non-Roman names or strings of different spelling variants do not match each other and will not be reliably retrieved from the database. The names written in different languages with different spelling are challenging to match as the form of the name may not match the other variant of the name present in the database. For example, the Russian names "Ivanov", "Ivanoff", or "Iwanow" will be difficult to match as the Soundex code will be different for these names. The Chinese name may be

written as “Hsiao” or “Xiao” and will not match due to different Soundex codes (Patman and Shaefer, 2001; Shah and Kumar Singh, 2014).

- ***Silent consonants*** - Soundex does not match the names with silent consonants, so a different code for each name is generated. Because Soundex does not identify silent consonants in the name or any name with simplified spellings with omitted consonants, for example, names “Meghburn” and “Meburn,” “Coghburn” and “Coburn,” or “Deighton” and “Dayton.” will have different codes and will not match with each other (Patman and Shaefer, 2001; Shah and Kumar Singh, 2014).

- ***Name syntax*** - In different cultures, the structure of the name is different. Usually, the structure of the name is First, Middle, and Last. It may result in inconsistency in the database in the way names are mapped. Soundex provides no means for accommodating such types of variation. Instead, it codes them as two different, non-matching names, resulting in two closely related variants not retrieving each other. For Example, “Sheikh Ali Mohamed” may be in one database. However, in another database, it may be recorded separately as first name “Sheikh”, middle name “Ali”, and last name “Mohamed”. There can be situations where the first name is recorded as “Sheikh Ali” and the last name “Mohamed”. It is difficult for Soundex to generate the matching code for such name variations. However, there can be different variations of the name that Soundex code may encounter, such as if an Arabic name, “Alhameed,” is written as “Hameed”, “Hamid,” or “Hamed,” then Soundex code will struggle to match even though all these name variations refer to the same individual because Soundex was not designed to tackle such different variations in names (Patman and Shaefer, 2001; Shah and Kumar Singh, 2014).

- ***Name Initials*** - Normally, first names in long names are written with initials, but this is not limited to only long names. Mostly, full names are substituted with initials. For example, “Mikhail Kovalchuk” and “M.Kovalchuk,” will have different Soundex codes and will not be matched to each other (Patman and Shaefer, 2001; Shah and Kumar Singh, 2014).

- ***Poor precision*** - The Soundex algorithm sometimes can match many dissimilar names in the result, and the amount of poor precision results increases with the database size. For example, the name "Criton." search will match with "Courtmanche," "Corradino," "Cartmill," and "Cortinez" (Patman and Shaefer, 2001; Shah and Kumar Singh, 2014).

- ***Noise intolerance*** - Soundex cannot overcome typographical errors because it relies on the sound of letters and does not handle common transpositions. This spelling variation will generate false matches. For example, if the name is mistyped as "Msith" in the database, it will not match "Smith" due to a transposition error. Similarly, "Hubbins" will not be matched with "Huggins," and "Hreman" will not match with "Herman" (Patman and Shaefer, 2001; Shah and Kumar Singh, 2014; Pilania and Kumaran, 2019).

- ***Long Name*** - The Soundex code combines an initial letter and three digits for any given name. Therefore, the Soundex code will ignore the name length and generate the code that may match other names that are not the same length, but the string may match based on sound. For Example, searching for the name "Rameshwar" will match with the name "Rameshwaram" even though the name may refer to two different individuals because the Soundex code will be the same for both names, and this will result in false matching (Pilania and Kumaran, 2019).

Even with all the above issues found in Soundex, the algorithm is mainly used for name matching and retrieval. The most common use of Soundex is in airline reservation systems to generate passenger name codes to avoid confusion when pronouncing the names. In informational retrieval, Soundex is used to overcome the problem of names with alternate spellings. It plays a vital role in approximate name matching due to human errors. However, Soundex can be helpful in systems that are not very sensitive to false results and can accept if results are high false positive or negative (Shah and Kumar Singh, 2014).

2.6. Comparison of Matching Algorithms

The choice of one from the following string-matching algorithms mainly depends on the nature of the error in searching the text data. Table 2.4 compares all phonetic matching algorithms based on the language specification, features, and limitations (Shah and Kumar Singh, 2014; Soundex, 2015).

Table 2.4 - Comparison of String Matching Techniques (Shah and Kumar Singh, 2014)

Algorithm	Type	Language	Usage
Soundex	Phonetic	English	Misspelt English words
Edit distance	Pattern matching	N/A	Local spelling errors
Jaro-Winkler	Pattern matching	N/A	Spelling and typewriting errors

2.7. Clustering Technique

Clustering is a widely used technique in machine learning and data mining, where the goal is to group similar data points into distinct clusters based on their similarities. Clustering is an unsupervised learning method as it does not rely on labelled data but instead discovers patterns or structures within data (Bezdek, 1981).

Clustering is significant for data mining and pattern recognition. Clustering means grouping all similar records in one group based on criteria set to compare the similarity of each string or pattern. (Jain, Murty and Flynn, 1999; Halkidi, Batistakis and Vazirgiannis, 2001; Dharmarajan and Velmurugan, 2013; Nisha and Kaur, 2015). Clustering has emerged as a popular technique for pattern recognition, image

processing, and, most recently, data mining. Clustering algorithms are increasingly required to deal with large-scale data sets containing categorical and numeric data, particularly in data mining. According to (Mamun, Aseltine and Rajasekaran, 2014), clustering refers to the record linkage problem where records are grouped based on similarity.

The representation of objects is the main feature of clustering, as objects are represented as patterns to find the similarity (Filippone *et al.*, 2008). The patterns are considered similar if they are in the same cluster but considered different if not in the same cluster. This difference should be clear and meaningful to represent patterns in the cluster (Xu and Wunsch, 2005). (Monge and Elkan, 1997) they improved the nested-loop record comparison performance by showing a transitive approach for duplicate detection. For example, if 'A' and 'B' are duplicates and 'B' and 'C' are duplicates, then 'A' and 'C' are assumed duplicates. The problem with this approach is that record matching relies on the dependency of connected components of the graph. If these connected components are accurate, then the assumption can be valid; otherwise, no relationship can be retrieved between records. So, to compute the connected components efficiently on the graph, (Monge and Elkan, 1997) used a method called union-find. This method combines records into a cluster during the union stage. In contrast, the cluster is used as a comparison representation, and the number of record comparisons gets reduced for duplicate detection. So, we can say that if 'A' is not a duplicate of 'B' already in the cluster, other members in the cluster will not be duplicates of 'A'.

2.7.1. The Types of Clustering Techniques

Many clustering algorithms are available for specific problems, such as hierarchical, k-means, mean-shift and fuzzy clustering. Each technique requires matching criteria to find the distance between different clusters. Usually, edit distance is a common technique used for this purpose.

2.7.1.1. K-Means Clustering

K-Means is a widely used partition-based clustering algorithm that aims to divide the data into K clusters, where K is a user-specified parameter. One method that has proved particular efficiency is the k-means algorithm (Jain, Murty and Flynn, 1999). (Huang, 1998) developed the k-modes algorithm by extending the standard k-means algorithm with a simple matching dissimilarity measure for categorical data and a frequency-based method to update centroids in the clustering. The numeral-only limitation of the k-means algorithm does not constrain this extended method. It has shown efficient clustering performance in real-world databases.

The algorithm works as follows:

- Randomly initialize K centroids (points representing each cluster's centre).
- Assign each data point to the nearest centroid, creating K clusters.
- Recalculate the centroids by taking the mean data points in each cluster.
- Repeat steps 2 and 3 until convergence (when the centroids no longer change significantly or after a certain number of iterations).

K-Means tries to minimize the within-cluster sum of squares (inertia) by iteratively optimizing the positions of the centroids. While K-Means is efficient and easy to implement, it may converge to a local minimum, depending on the initial placement of centroids. K-Means is widely used in customer segmentation, market analysis, image compression, and pattern recognition tasks. It is particularly effective when the clusters are well-separated and approximately spherical (Lloyd, 1982). (WONG, 1979; Lloyd, 1982) identifies some advantages and disadvantages of K-mean as below:

Advantages:

- Fast and efficient for large datasets.
- Scalable and easy to implement.
- Well-suited for convex-shaped clusters.

Disadvantages:

- Must predefine the number of clusters
- Sensitive to the initial placement of centroids.
- It converges to local optima.
- It is not suitable for clusters with irregular shapes or different sizes.

Furthermore, the fuzzy k-modes algorithm generates the fuzzy partition matrix from categorical data with the framework of the fuzzy k-means-type algorithm (Bezdek, 1981; Bezdek *et al.*, 1999) and improves on the k-modes algorithm by assigning confidence degrees to data in different clusters. In most fuzzy versions of clustering algorithms, the assigned data memberships to a cluster are fuzzy, but the centroid is not fuzzy. For example, although the fuzzy k-modes algorithm efficiently handles categorical data sets, it uses a complex centroid representation for categorical data in a cluster. This use of hard rejection of data can lead to misclassification in the region of doubt. The main focus of clustering is discovering duplicate entities in a dataset without unique identifiers. The entities in the dataset may be referenced differently from merging records in the datasets. One common approach to tackling this problem includes clustering, such as hierarchical agglomerative and k-means clustering, where each cluster represents an entity. However, a problem with many of these existing approaches is that they require the number of clusters to be set in advance (Dai, 2011). Using the Mean-Shift technique gives flexibility in the cluster. It does not require a fixed number of clusters to be defined in advance.

2.7.1.2. Mean-Shift Clustering

Mean-Shift is an iterative process that moves each point to the average point in the cluster (Yizong Cheng, 1995). Mean-Shift was originally introduced by Fukunaga and Hostetler in 1975, where the iteration process shifts points until all points are converged to estimate the “mode” to define a cluster (Fukunaga and Hostetler, 1975; Comaniciu and Meet, 1999; Carreira-Perpiñán, 2015). There is no need to predefine the number of clusters as the number of clusters is obtained automatically by finding the centre of

modes. According to (Georgescu, Shimshoni and Meer, 2003), finding the closest neighbours of data points in the area is the most expensive operation in the Mean-Shift method. The mode is the densest area where most of the data is located due to data points being moved there during the iteration process, and the process continues until all data points are moved to the mode. Because of the nature of the process, the mean shift is a sequential clustering algorithm. The second-in-line data point will be processed until the first data point is moved to the mode. Therefore, each data point process waits until the previous process is completed. In this process, the close data points require less iteration, but data points far from each other require more iteration (Li, Hu and Wu, 2007; Wu and Yang, 2007).

(Liu *et al.*, 2013) states that the Mean-Shift iteration procedure is based on two essential steps:

- The distribution of data points is based on constructing a probability density model.
- Each data point is mapped to the density model nearest to the point. Therefore, the density model is an integral part of machine learning for representing clusters.

Typically, Mean-shift is used in machine vision for image analysis, image processing, pattern recognition, and objective tracking. In pattern recognition, mode plays a vital role in applications like classification, feature extraction, image segmentation and object tracking (Comaniciu and Meer, 2002). In 1995, Cheng generalized the Mean-Shift algorithm, and in machine vision, the algorithm became popular (Ghassabeh, 2013). Mean-Shift is vital in discovering data presented in arbitrary clusters (Anand *et al.*, 2014).

The following steps are involved in working of the Mean-Shift clustering algorithm (Comaniciu and Meer, 2002):

Step 1 − First, start with the data points assigned to a cluster of their own.

Step 2 − Next, this algorithm will compute the centroids.

Step 3 − In this step, the location of new centroids will be updated.

Step 4 – The process will be iterated and moved to the higher-density region.

Step 5 – At last, it will be stopped once the centroids reach a position from which they cannot move further.

The Mean-Shift algorithm has advantages and disadvantages, as described by (Yizong Cheng, 1995; Comaniciu and Meer, 2002). The following are some advantages and disadvantages are discussed below :

Advantages:

- It does not need to make any model assumption as in K-means
- It can also model complex clusters which have nonconvex shapes.
- It only needs one bandwidth parameter, which automatically determines the number of clusters.
- There is no issue with local minima, unlike K-Means.
- No problem was generated from outliers.

Disadvantages:

- The mean-shift algorithm does not work well in the case of high dimensions, where the number of clusters changes abruptly.
- There is no direct control over defining the number of clusters.
- It cannot differentiate between meaningful and meaningless modes.

2.7.1.3. Hierarchical Clustering

Hierarchical clustering creates a tree-like structure of nested clusters called a dendrogram without requiring a predefined number of clusters. There are two main approaches to hierarchical clustering (Ward, 1963):

- Agglomerative (bottom-up): Starts with each data point as its cluster and repeatedly merges the closest pairs of clusters until only one cluster remains.
- Divisive (top-down): Starts with all data points in one cluster and recursively splits the clusters into smaller ones until each data point is in its cluster.

The choice of merging or splitting is determined by the linkage criteria, such as single linkage (distance between the closest points in each cluster), complete linkage (distance between the farthest points in each cluster), or average linkage (average distance between all points in each cluster). The dendrogram can be cut at a certain level to obtain a specific number of clusters. The algorithm has advantages and disadvantages defined by (Ward, 1963; Fionn Murtagh, 2014):

Advantages:

- It provides insights into hierarchical structures in the data.
- It is flexible and can be cut at different levels to obtain cluster granularities.

Disadvantages:

- It is Computationally expensive for large datasets.
- The choice of linkage criteria impacts the clustering result.
- The best-used areas are Biology, bioinformatics, social sciences, image segmentation, and document clustering.

2.7.1.4. Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN is a density-based algorithm that groups data points based on their density. It requires two parameters: "epsilon" (ε), which defines the neighbourhood radius around each data point, and "min Pts," which sets the minimum number of data points within ε to form a core point. DBSCAN automatically determines the number of clusters and is

robust to outliers and irregularly shaped clusters. The algorithm works as follows (Martin Ester, Hans-Peter Kriegel, Jiirg Sander, 1996; Schubert *et al.*, 2017):

Step 1 - Find all the points within distance ε of each data point. It becomes a core point if a point has at least "minPts" neighbours.

Step 2 - Expand the cluster from the core point by adding all reachable points within ε to the cluster.

Step 3 - Repeat the process for all core and non-core points within the cluster's reachability distance.

Step 4 - Points not reachable from any core points are considered noise and do not belong to any cluster.

The algorithm comes with its advantages and disadvantages:

Advantages:

- Automatically determines the number of clusters.
- Robust to outliers and noise.
- It handles clusters of arbitrary shapes and sizes.

Disadvantages:

- Sensitive to the choice of epsilon and min Pts.
- It may be difficult with datasets having significantly varying densities.
- The best-used areas are Spatial data analysis, anomaly detection, and varying-density datasets.

2.7.1.5. Gaussian Mixture Model (GMM)

GMM is a probabilistic model-based clustering algorithm that assumes data points are generated from a mixture of several Gaussian distributions (Dempster, Laird and Rubin, 1977).

The algorithm works as follows:

Step 1 - Initialize the parameters (no. of kernels) of the Gaussian components (mean, covariance, and weight).

Step 2 - Expectation step (E-step): Calculate the probability that each data point belongs to each Gaussian component.

Step 3 - Maximization step (M-step): Re-estimate the parameters of the Gaussian components based on the probabilities from the E-step.

Step 4 - Repeat steps 2 and 3 until convergence.

The data points are then assigned to clusters based on their probabilities of belonging to each Gaussian component. GMM allows soft assignments, meaning each data point can belong to multiple clusters with different probabilities.

There are the following advantages and disadvantages of this algorithm:

Advantages:

- Soft assignments of data points to clusters.
- Can capture complex data distributions.
- It handles overlapping clusters.

Disadvantages:

- Sensitive to the initialization of parameters.
- May converge to local optima.
- The best-used areas are Image segmentation, speech recognition, and pattern recognition tasks with overlapping clusters.

2.7.1.6. Spectral Clustering

Spectral Clustering is a graph-based algorithm that treats data points as nodes and clusters them based on the graph structure (Von Luxburg, 2007).

The algorithm works as follows:

Step 1 - Construct an affinity matrix that measures pairwise similarities between data points. Common choices include the Gaussian affinity and k-nearest neighbour affinity.

Step 2 - Compute the graph Laplacian matrix, which encodes the graph structure.

Step 3 - Obtain the eigenvectors or eigenvalues of the Laplacian matrix and use them to create a lower-dimensional data representation.

Step 4 - It performs K-Means or other clustering techniques on the lower-dimensional representation to identify the clusters.

Spectral Clustering is helpful for non-convex clusters and can handle data in high-dimensional spaces.

The algorithm has some advantages and disadvantages, as listed below:

Advantages:

- It handles non-convex clusters and high-dimensional data.
- Performs well with datasets having clear low-dimensional structures.

Disadvantages:

- Computationally expensive for large datasets.
- It requires parameter tuning for the affinity matrix.
- The best-used areas are Image segmentation, community detection in social networks, and dimensionality reduction tasks.

2.8. Comparison of Clustering Techniques

The following Table 2.5 provides the comparison summary of the clustering techniques discussed above. This comparison discusses each clustering technique's key points, advantages, limitations, and best-used areas.

Table 2.5 - Comparison of Clustering Techniques

Clustering Technique	Key Points	Advantages	Limitations	Best-Used Area
K-Means	Partition-based, iterative K clusters	Fast and efficient	Sensitive to initial centroids, local optima	Customer segmentation, market analysis, image compression, pattern recognition
Hierarchical	Hierarchical representation, agglomerative	Provides insights into a hierarchy	Computationally expensive linkage criteria	Biology, bioinformatics, social sciences, image segmentation, document clustering
DBSCAN	Density-based, automatic clusters	Handles arbitrary shapes, robust to noise	Sensitive to parameters, varying density	Spatial data analysis, anomaly detection, clusters with varying density
Mean Shift	Mode-seeking, automatic clusters	Handles arbitrary shapes, robust to noise	Computationally intensive, bandwidth parameter sensitivity	Image segmentation, identifying clusters with varying density, computer vision
GMM (Gaussian Mixture Model)	Probabilistic model, soft assignments	Captures complex data distributions	Sensitive to initialization, local optima	Image segmentation, speech recognition, pattern recognition with overlapping clusters
Spectral Clustering	Graph-based, handles non-convex clusters	Handles high-dimensional data	Computationally expensive parameter tuning	Image segmentation, community detection, dimensionality reduction, clusters with low-dimensional structures

2.9. Knowledge-Base

In machine learning, a knowledge base is a repository or collection of structured and organized knowledge that supports an intelligent system's learning and decision-making processes. It is a central source of information that the system can access and use to make informed decisions, solve problems, and acquire new knowledge. Knowledge bases are crucial in various AI applications, including expert systems, natural language processing, semantic search, and knowledge-based question-answering systems.

2.9.1. Types of Knowledge-Base

2.9.1.1. Rule-Based Knowledge-Base

A rule-based knowledge base consists of rules expressed as "if-then" statements. Each rule represents a piece of knowledge or a condition the system can apply to make decisions or perform actions. Rule-based systems benefit expert systems, where human experts' knowledge is represented as rules for automated decision-making. These systems are widely used in medical diagnosis, finance, and control systems (Bharambe, Jain and Jain, 2012).

2.9.1.2. Ontology-Based Knowledge-Base

An ontology formally represents knowledge that defines a specific domain's concepts, relationships, and constraints. An ontology-based knowledge base organizes knowledge using ontologies, allowing the system to reason about the domain and perform semantic searches. Ontology-based systems are standard in natural language processing, semantic web applications, and knowledge representation tasks. They enable advanced semantic search and knowledge inference (Studer, Benjamins and Fensel, 1998; Noy and McGuinness, 2001).

2.9.1.3. Case-Based Knowledge-Base

A “case-based knowledge base” stores past experiences or cases as features and outcomes. The system can use these stored cases to solve new problems by retrieving and adapting relevant solutions from past cases. Case-based reasoning is widely used in machine learning and artificial intelligence applications, especially in systems that learn from experience. It is commonly employed in medicine, finance, and customer support (Aamodt and Plaza, 1996; Watson and Marir, 2007).

2.9.1.4. Knowledge Graph

A knowledge graph is a structured representation of knowledge, typically in a graph, where entities and their relationships are interconnected. Knowledge graphs capture complex relationships between entities and provide a powerful knowledge representation and reasoning framework. They are commonly used in various applications, including search engines and knowledge-based question-answering systems (Bollacker *et al.*, 2008; Bizer, Heath and Berners-Lee, 2009).

2.9.2. Use of Knowledge-Base in Machine Learning

2.9.2.1. Decision-Making

Knowledge bases provide a set of rules, facts, and domain-specific knowledge that guide decision-making processes. They are used in expert systems and AI applications to make informed choices based on the available information and rules. Expert systems like MYCIN and DENDRAL employed rule-based knowledge bases for medical diagnosis and chemistry problem-solving (Studer, Benjamins and Fensel, 1998).

2.9.2.2. Problem-Solving

Case-based and ontology-based knowledge bases facilitate problem-solving by leveraging past experiences and domain-specific knowledge. In case-based reasoning, past cases are used to solve new problems by retrieving similar cases and adapting their solutions. While Ontology-based systems reason over the structured knowledge in ontologies to provide context-aware and domain-specific problem-solving capabilities (Studer, Benjamins and Fensel, 1998).

2.9.2.3. Semantic Search

Knowledge bases based on ontologies and knowledge graphs enable semantic search, allowing the system to understand the context and meaning of queries. They support advanced search capabilities that consider relationships between entities, leading to more relevant search results. Semantic search is essential in natural language processing and web search engines (Yerva, Miklos and Aberer, 2010).

2.9.2.4. Knowledge Representation

Knowledge bases formally represent knowledge in a structured and organized manner. They enable knowledge sharing and reasoning within intelligent systems, facilitating more efficient learning and decision-making. Knowledge representation is fundamental to many AI applications, including expert and knowledge-based systems (Islam and Inkpen, 2008).

2.10. Summary

This chapter has explored different machine-learning techniques and discussed the entity resolution and challenges in matching the string in detail. The main three matching techniques and how they match any two strings were explained. The two matching techniques, Edit Distance and Jaro-Winkler, have been discussed to address a few matching challenges. In contrast, the Soundex technique has been explored to address string matching based on phonetic matching. Next, clustering techniques required to group similar matching strings were discussed. The last section of the chapter explains how the knowledge base is valuable in machine learning and how it can be used.

CHAPTER 3

3. RESEARCH DESIGN AND METHODOLOGY

3.1. Chapter Overview

This chapter presents the methodology used for this research. The discussion will then explain the research design approach in detail, along with the foundation of the methodology. The proposed model will outlined the following sections. The following sections will detail the implementation of the proposed model. Initially, the discussion will be of different tools and libraries for computer simulation. Then, it will explain the importance of improvements in the Soundex technique and present some matching explanations of this improved Soundex technique and the standard Jaro-Winkler technique. The next chapter will be the results chapter, which will present and explain the results based on this chapter.

3.2. Foundation of Research Methods (Design)

For this research evaluation of the similarity techniques, de-identified police data has been used for record matching and pattern recognition. The search runs iteratively using a combination of attributes for approximate string matching to find the desired entity, and the results create three data subsets. These data subsets will be compared and merged to produce the matched record(s) after the computation steps on the data. This research (Nawaz and Kazemian, 2021) was published in the EANN2021 Conference held on 25th-27th June 2021, and the published research paper is attached in Appendix A.

3.2.1. Foundation of the Proposed Model

In big data, names with different name variations or incorrect spelling may be an issue for exact matching. However, finding approximate matches for an entity leads to an arduous task. Nevertheless, missing or incomplete information in the database can also lead to considerable manual work to guess the matching record (Mon, Mie and Thwin, 2013).

The following sections discuss key terminologies, techniques, and the proposed model's design.

3.2.1.1. Data Pre-processing

Data pre-processing is a process that runs on data cleaning, standardizing, and fixing incomplete information. Data pre-processing helps when matching record values for one or more particular attributes and is considered a vital part of data mining (Huang *et al.*, 2008). Data pre-processing is a crucial preparatory step in entity matching, also known as record linkage or deduplication, which aims to identify and link records referring to the same real-world entities from different data sources. It involves a series of techniques and operations to clean, standardize, and transform the data, making it more suitable for subsequent entity-matching.

An essential step is removing the unwanted characters to clean the data. It can refer to eliminating unwanted text variation affecting the matching. The variations include uppercase or lowercase, different punctuations, and extra space between text (Branting, 2003). The standardisation is to bring the text into the same format to be easily compared, such as matching “London, UK” and “City of London, UK”. Different attribute values can be matched and combined to generate new attributes for matching to handle the issue of incomplete information. A piece of incomplete information can also be called missing information, affecting the matching process where the result can be inaccurate. Therefore, data pre-processing is essential to improve data quality and

consistency, leading to more accurate and efficient entity-matching results. However, some following steps can be used for data pre-processing:

➲ *Data Pre-processing steps in Entity Matching*

- ***Data Cleaning***

Data cleaning involves identifying and correcting errors, inconsistencies, and missing values in the data. Different techniques, such as data imputation, outlier detection, and error correction, are applied to ensure high data quality. Cleaning the data reduces the chances of false matches caused by errors or inconsistencies (Herzog and Reiter, 2008; Christen, 2012).

- ***Data Standardisation***

Standardisation involves transforming data to a standard format or representation. This step includes converting data to lowercase, removing punctuation, and applying stemming or lemmatization to text fields. Standardisation ensures that variations in data representations are minimized, facilitating more accurate matching (Oracle, 2010).

- ***Tokenisation***

Tokenisation is the process of breaking text into individual tokens or words. Tokenisation is particularly important for text-based entity-matching tasks. It helps create a more structured representation of textual data, which aids in identifying standard terms between records (Bird, Klein and Loper, 2009).

- ***Feature Extraction***

Feature extraction involves selecting and transforming relevant attributes or features from the data. Features like TF-IDF (Term Frequency-Inverse Document Frequency) and n-grams are commonly used for textual data. Numerical data may require feature scaling to bring all features to a similar scale. Selecting informative features is essential for accurate entity matching (Basu, Bhattacharyya and Kim, 2010).

- ***Blocking/Blocking Key Selection***

Blocking is a technique that reduces the search space by dividing data into smaller blocks. The choice of blocking keys (attributes) affects the granularity and efficiency of entity matching. Effective blocking ensures that only potentially matching records are compared, reducing computation time (Kirsten *et al.*, 2010).

- ***Data Transformation***

Data transformation may involve encoding categorical attributes, normalization, or feature engineering. Transforming data ensures that it is in a suitable format for the entity-matching algorithms being used. The choice of transformation techniques depends on the specific data characteristics and the matching approach (Tejada, Knoblock and Minton, 2002).

➲ *Importance of Data Pre-processing in Entity Matching*

Data pre-processing plays a critical role in entity matching for the following reasons mentioned (Bhattacharya and Getoor, 2007).

- *Improved Data Quality*: Data cleaning and standardisation techniques enhance data quality by removing errors and inconsistencies and standardizing representations. Clean data leads to more accurate matching results.

- *Reduced False Matches*: Pre-processing reduces the likelihood of false positives and false negatives in entity-matching results by handling data variations and inconsistencies.

- *Efficient Computation*: Effective blocking and feature selection reduce the computational complexity of the matching process. It makes entity matching more efficient and scalable, especially for large datasets.

- *Enhanced Matching Accuracy*: Tokenisation and feature extraction techniques improve the ability to identify similarities between records, increasing the matching accuracy.

3.2.1.2. Classification of Data

Classification is typically a process in which each record pair gets classified into a match and no matching category, manually setting the threshold value for record linkage (Ashish and Toga, 2016). The probabilistic record linkage was proposed by (Fellegi and Sunter, 1969), and most record linkage systems have been based on this approach for years. However, many researchers have recently been using binary classification techniques based on machine learning and data mining because these techniques provide better results for field comparison of any record (Ravikumar and Cohen, 2004). The similarity of each field comparison is calculated between the values of 0 and 1. Many research works have been conducted, and different similarity techniques were introduced to achieve this, where the field type can influence the result (Wang and Wang, 2016).

3.2.1.3. Clustering of Data

Clustering is significant and is one of the famous techniques for pattern recognition, data mining and image processing. Clustering algorithms typically deal with large datasets, primarily numeric data (Du, 2010). K-means is one of the efficient clustering algorithms. It has a fuzzy variation called the fuzzy k-modes algorithm, which produces the fuzzy partition matrix of the fuzzy k-means-type algorithm. In this fuzzy clustering algorithm, clusters are fuzzy. However, at the same time, centroids are not fuzzy, which can lead to misclassification in the cluster. The main focus of clustering is discovering duplicate entities in a dataset without unique identifiers. The entities in the dataset may be referenced differently from merging records in the datasets. The clustering techniques that tackle this problem are hierarchical and K-means clustering. However, these methods do not address the problem of fixed clustering, which involves specifying the number of clusters in advance (Dai, 2011).

3.2.1.4. Knowledge-Base (KB)

This phase of the proposed model will be used as a reference to understand the results in graph and table format while creating the knowledge base for future searches. Any new search will be compared with the pool of records in the knowledge base. The three classifiers labelled “match, possible match and close or related match” will be used to store records in KB accordingly. Any new, similar or identical search looking for “identity match” first in KB will speed up the matching process. It will help to reduce the processing time for records to be matched or to find similarities to the records found in the knowledge base. If the searched input is found in KB, it will be compared with the central database to check for any new changes to the record. Once the record reference has been updated, it will be stored in KB for future searches. However, suppose the searched input record is not found in KB. In this case, only the central database will be used to find matches, possible matches and related match records. After such a classification, new records will added to KB for future searches.

3.2.1.5. Record Linkage

It is crucial to extract the correct information for the record based on the correct relationship. The record linkage process links the shared values of particular attributes and finds the relationship in each record. Related or similar values might appear in this record linkage process, which can be duplicate records and a step closer to entity resolution (Brizan and Tansel, 2006). In simple words, connecting different records in the dataset to find the relationship is called record linkage, which helps to find the one actual record representing the real-world entity record. However, the dataset might contain errors such as misspellings, and the vast number of records makes it difficult to correctly match all records (Basu, Bhattacharyya and Kim, 2010; Mamun, Aseltine and Rajasekaran, 2014). So, record linkage needs to handle such errors during the clustering process, and it is only possible by using the approximate matching or fuzzy matching of attributes of records. The efficient record linkage is crucial for identity resolution to find the correct real-world individual identity. Nevertheless, the record linkage performance depends on accuracy, the number of required comparisons, complexity and the time required to complete the process (Wangikar, Deshmukh and Bhirud, 2016).

The relationship between records can be found by comparing it with family members and friends, which can quickly determine the real-world target entity. The information about an entity allows an individual to be linked to other entities related to an individual entity. This information helps to create links between all possible and related entities. This linkage is called *Link analysis,* which involves connecting and comparing the information within a dataset or multiple datasets (Brizan and Tansel, 2006).

3.2.1.6. Selection of Attributes for Searching Records

It is usually straightforward to provide the search query and find the required results from the database based on the search criteria set. However, if there are typographical errors in the database, the query result will not be accurate (Brizan and Tansel, 2006).

In entity resolution, finding the results as accurately as possible is imperative. For this, it is essential to run the number of queries on the database to retrieve the matching record. It is known that each entity contains different attributes, which help differentiate one entity from another. However, appropriate matching techniques are required to search records. Usually, in a single dataset, simple attributes such as address can be used to link or associate the entity with other entities.

Similarly, a forename and surname can also allow association or link the single entities with other entities matching surnames and address information as a combination of matching criteria. Also, date of birth is another attribute that helps determine the entities' linkage. It is challenging to compare the date of birth due to the different formats, such as “YYYY.MM.DD” or “MM.DD.YYYY” or “DD.MM.YYYY”. The combination of name and date of birth will not be good enough to link or associate the entity with other entities if there are no typographical errors in the records. The date format issue has been tackled in the proposed model by converting the date of birth attribute values to a string value.

3.2.1.7. Labelling of Matching Records

It is crucial to establish criteria for entity resolution and record linkage to match the records by comparing the value of attributes of the record. Typically, the matching criteria are used to compare the attributes of one or two records to generate the result in “Match or No Match”. (Brizan and Tansel, 2006) state that Fellengi and Sunter defined the matching criteria for records to be classified as “match”, “no match”, and “needs more review”. Establishing the matching criteria is critical as most of the data in the database is text-based, and entity resolution needs matching criteria for text matching as follows:

- *Match* criteria consider a match where all the values of more than one particular attribute exactly match. It is the easy matching criteria. For example, if a record's last name, first name and date of birth are the same, it will be categorised as “match”. According to (Cohen and Richman, 2002), the matching criteria may

contain partial matches only. However, records can be an exact match only with no typographical errors or other missing details.

- *Possible match* criteria, where at least one or more than one value for a particular attribute is similar, will be considered a "possible match". In fuzzy matching, it is vital to determine a possible match to help classify the records based on the matching criteria. For example, a record contains a similar surname, date of birth, and different forename. However, a similar address will be considered a possible match.

- *Related or Close match* criteria, where any one or more values for a particular attribute are similar. It can also be called a "close match". Finding the related match helps to classify the records based on the matching criteria. For example, a record containing a similar date of birth and address but a different name will be considered a related or close match.

- *No match* criteria, where values for a particular attribute that do not match or contain any similarity will be considered "No match". For example, if a record does not contain or match the name (forename, surname), date of birth and address, it will be considered no match. Because some records may match the date of birth while the rest of the attribute values do not, it will still be considered a no-match. In the past, all researchers have included no match in the matching criteria. However, it will not be included in this proposed model to eliminate non-match records from further processing and simplify the clustering and segmentation process. The "No match" records are dropped during the initial search iteration phase from further processing.

In this research, a new algorithm has been introduced to overcome the performance shortage of the Jaro-Winkler and Soundex similarity metrics to produce the desired fuzzy matching results. This new hybrid fuzzy matching algorithm contains all the good features of Jaro-Winkler and Soundex similarity metrics discussed in the previous chapter.

In the proposed algorithm shown below in Figure 3.1, each input name is searched based on Soundex code to find similar matches pronounced similarly. During this process, Soundex encoding is applied to all the names in the database and compared to the Soundex code of the input name. The two hash values are created for all the matching names by applying the Jaro-Winkler technique on the Soundex encoding of each name. The matching value between 0 and 1 is generated by the Jaro-Winkler distance. By doing this, the aim has been to get improved results by using the Jaro-Winkler and Soundex similarity metrics to generate an aggregate score for entity matching. All names in the database will have Soundex code generated, and a Jaro-Winkler score will be generated for the target name and names in the database. The Soundex code score is generated by applying the Jaro-Winkler technique to match the database's target name and other names. Once there are two distance scores, one based on the Soundex code and another from Jaro-Winkler, each name's aggregate score is generated. Converting Soundex codes into the Jaro-Winkler score tackles most name variations and Soundex issues discussed in the previous chapter.

Typically, the Soundex is based on phonetic matching by generating the 4-digit code for the given name and matching names that sound or are pronounced similarly without worrying too much about the spelling of the name (Kessler, 2005; Shah and Kumar Singh, 2014).

However, in this research, the Soundex code generates a 6-digit code. The methodology involves the combination of two different string similarity metrics to find an approximate match for the entity by applying it during the iterative search process. The iterative process is based on the combination of different attributes with three different possible name variations. These name variations are not fixed and can be anything based on how the name is pronounced. Each iterative search generates a dataset that is later combined into one final dataset. The approximate score of names is based on aggregate calculations of Soundex's and Jaro-Winkler's approximate scores. The FuzzyWuzzy Python library is used to generate the long string address score. The Mean-Shift algorithm groups resultant records based on aggregate score and calculated age. The clusters are calculated automatically based on a matched number of records, and there is no need to provide a fixed number. After this, the results are further analysed by

segmenting the records based on further similarities and displaying the relationship in the graph to detect the identity. The results will be categorised as " Match", "Possible Match", and "Close or Related Match".

The following section explains the proposed model for identity resolution and discusses the design phases in detail, building upon the foundational techniques outlined in this section.

3.3. The Proposed Model for Identity Resolution

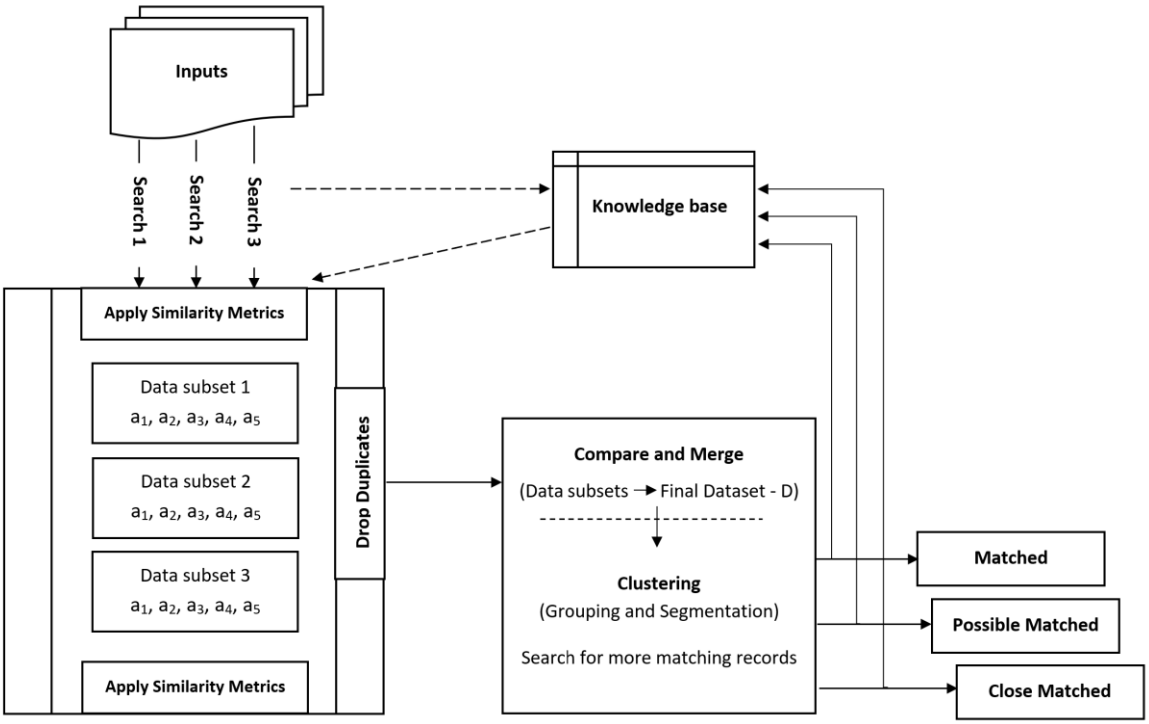

Figure 3.1 - Proposed Identity Resolution Model

3.3.1. Using Data Pre-processing

In the proposed model, the data pre-processing is simple, and the main focus is to match the field's values simply by using the similarity metrics collectively. The selected attributes are "surname", "forename", "gender", "address", "postcode", "description" and "date of birth". The input values to match can be partial, e.g., matching birth year (converted into age) instead of the complete date of birth, partial postcode, or part of the address. The proposed model converts all the values in the selected attributes to lowercase. All the numeric fields are matched as string values, such as "postcode" and "date of birth", in the records as a string. As discussed in previous sections, comparing the "date of birth" as a string benefits the search and eliminates the date format issues. Figure 3.2 below defines the used attributes:

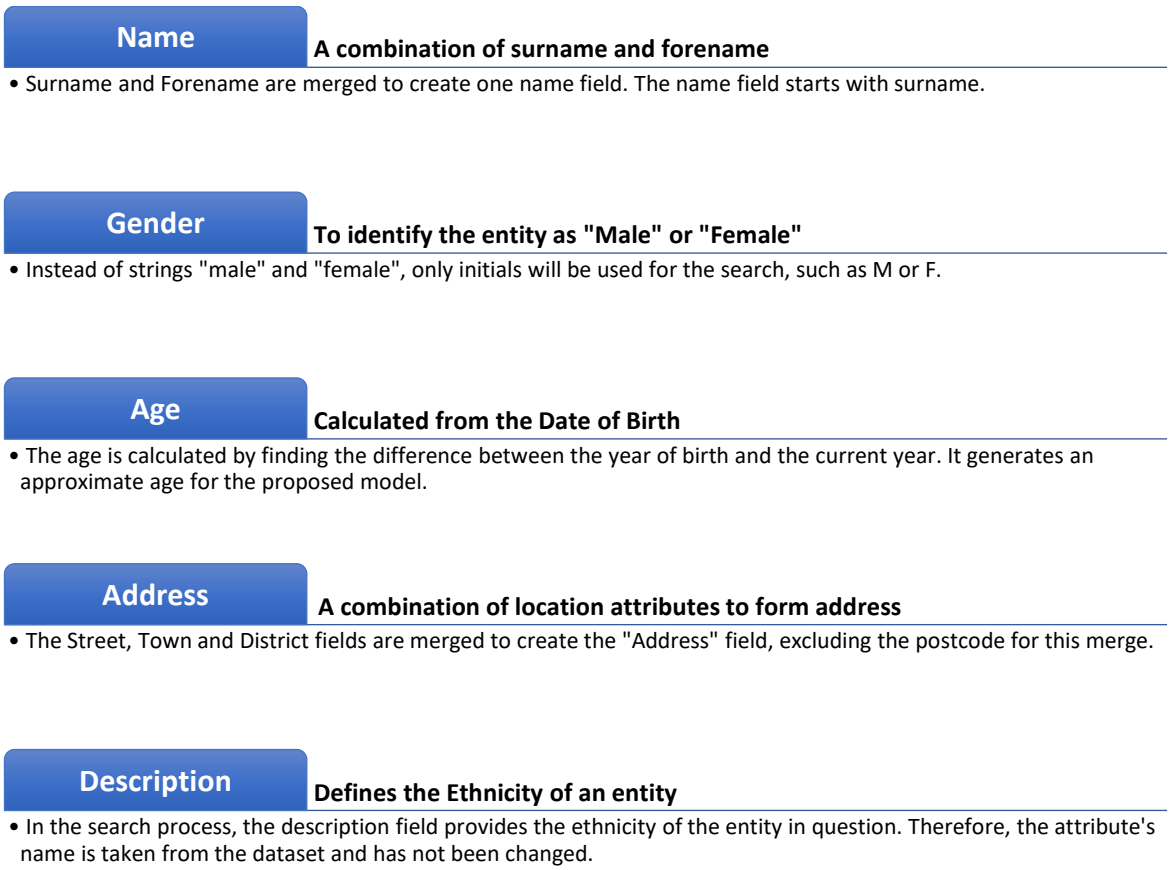

Figure 3.2 - Selection of attributes for the dataset

During this phase, the dataset splits into two datasets. The purpose of this dataset is to separate complete records from incomplete records with different attribute values.

3.3.2. Creation of Attributes Formation

- *Name* – In the policing dataset, two separate fields define entity name. The first field is a surname, a given family name of an entity. The second field is forename, the first name given by parents. For the proposed model, to simplify and generate one similarity code, the two fields, surname and forename, are combined into one field as a Name. In this Name field, the name for each entity starts with surname and then forename. The Name field will consider all the naming issues discussed about similarity matching algorithms in the previous chapter. Figure 3.3 shows the formation of the Name field.

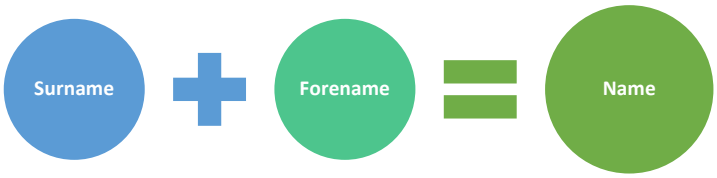

Figure 3.3 - Formation of the Name attribute for the dataset

- *Address* – In the policing dataset, multiple fields provide street, town, and district names and postcodes. All these fields are combined for the proposed model to form the address field instead of having different fields representing address. The postcode field is not combined with this Address field. The main reason is to keep the postcode separate for an approximate match based on the partially known postcode while using the Address field for approximate matching for any known address information. It gives the two sets matching flexibility based on Address and postcode. Figure 3.4 below shows the formation of the Address field.

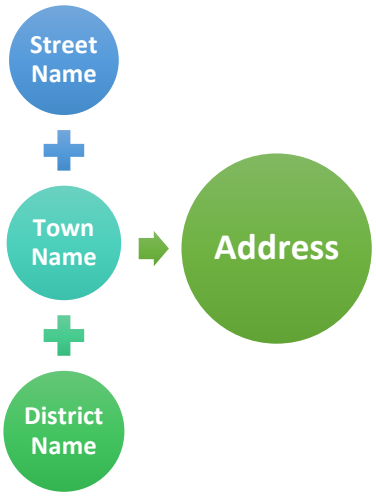

Figure 3.4 - Formation of the Address attribute for the dataset

- *Age* – For the proposed model, the full Date of Birth is required, and only approximate age is required to populate the records during the search. So, the calculation is done by extracting the birth year and subtracting it from the current year to calculate the approximate number. This resultant number is counted as Age and added to the policing datasets while keeping the date of birth in the dataset. Figure 3.5 shows the extraction of age from the Date of Birth.

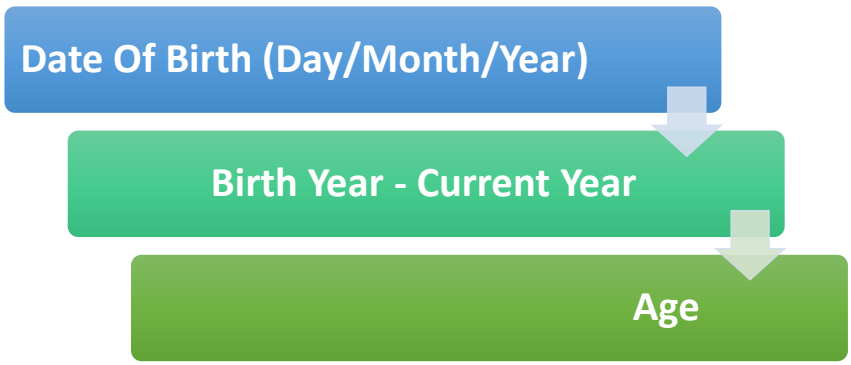

Figure 3.5 - Formation of the Age attribute for the dataset

The block diagram of the fuzzy approach for complete records shown in Figure 3.6 uses string similarity techniques in a cascaded manner, scoring the names to find an approximate name match.

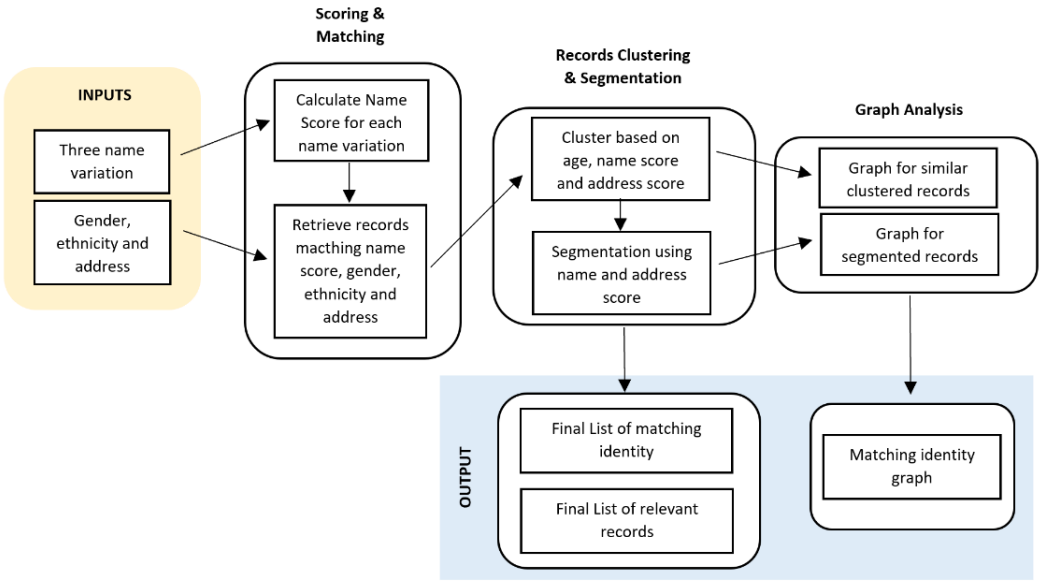

Figure 3.6 - The block diagram of a fuzzy approach to identity resolution using complete records

3.3.3. Selection of Searching Criteria

The proposed model introduces the iterative process for searching the target name in the database. The search iteration process runs in three passes, combining selected attributes to generate the search results. The search is formed in each pass with different name variations, and the result of each pass will be stored as a data subset. Once the iteration process is completed, all three data subsets will be compared for duplicate records and merged to generate one final dataset. The search process is bound to similarity metrics for matching text in case of errors.

Figure 3.1 shows the iterative process that runs in three passes and can be denoted as follows:

$$S = \{s_1, s_2, s_3\}$$

Here "S" represents the search iteration process, and s_1, s_2, and s_3 are the iterations, respectively. Each iteration process combines the selected attributes {a} to generate the

search results as data subset {d}. Each iteration uses a different name variation, and the result of each pass is stored as a data subset {d}. It can be denoted as:

$$d_i = \{a_1, a_2, a_3, a_4, a_5\}$$

Here "i" represents the data subsets as d_1, d_2, and d_3, respectively. Once the iteration process is completed, all three data subsets are compared by removing duplicate records and merging to form the final dataset "D". This can be denoted as follows:

$$D = [\{d_1, d_2, d_3\} - dup] \quad \textbf{Equation 3.1}$$

Here, "dup" represents the duplicate records.

Another process is scoring the address field strings with the help of the FuzzyWuzzy Python library's partial token function by comparing the input address during the initial search process. The scoring of addresses helps to retrieve the related records for the target name based on the matching address or part of the address. Once the matching and related records dataset is generated, it runs on the incomplete dataset to pick any matching records. Figure 3.6 shows the matching of records and different phases to generate results from the record's complete dataset. The fuzzy approach eliminates most no-match entities based on the low aggregate score compared to the threshold value set during the initial search. Finding the results as accurately as possible is essential because each record contains different attributes that help differentiate one entity from another.

3.3.4. Calculating the Aggregate Score

This fuzzy approach uses Soundex and Jaro-Winkler similarity techniques to calculate the aggregate score for name matching, as shown in Figure 3.6. The names are encoded using Soundex code, and the Jaro-Winkler score (JWscore) is calculated for each name and records are retrieved based on the matching score. Due to the mistake of the match of the Soundex code and the Jaro-Winkler score format, the Jaro-Winkler matching technique is applied to the Soundex code of each name to generate the matching score

called the Soundex score (Sscore). All these scores are added (+) together and multiplied (*) by the average value of 0.5 to get the aggregate score (AggScore) ranging between 0 and 1.

For this purpose, the Soundex algorithm has been modified by removing the retention of the first character of a name as a constant letter in the code while increasing the code length to 6 digits. It generates a numerical 6-digit Soundex code to help eliminate the Soundex first character mismatch issue. Increasing the code length helps to reduce many false-positive retrievals compared to the 4-digit code.

The conversion of name s_i and s_j to Soundex code into Jaro-Winkler score and aggregate score calculations are as follows:

$$Sscore = JWscore(s_i, s_j) \qquad \textbf{Equation 3.2}$$

The aggregate score is calculated with the following equation:

$$AggScore = (Sscore + JWscore) * 0.5 \qquad \textbf{Equation 3.3}$$

The names will be labelled based on the following criteria:

- *Possible Match* - The aggregate score is less than or equal to 1.0 and greater than or equal to 0.90. i.e. aggregate score >= 90% and aggregate score <= 100% matching score
- *Close Match* - The aggregate score is less than 0.90 and greater than or equal to 0.70. i.e. aggregate score < 90% and aggregate score >= 70% matching score
- *No Match* - The aggregate score is less than 0.70. i.e. aggregate score < 70% matching score

3.3.5. Selection of Records Comparison Criteria

Figure 3.1 shows that all three data subsets will be merged and compared for duplicate records to form one final dataset. Even though all duplicate records are removed at this stage, records will be retrieved with a low aggregate score, e.g., a score of 0.50 or 0.60 with a matching aggregate code. Any "No match" or irrelevant records will be dropped based on the name fuzzy matching score of attribute values. However, to ensure the threshold filtering is not missing any relevant records, as shown in Figure 3.6, the matching records are based on the selected attribute values in the initial search dataset that the search iteration process will produce. It will help to create the final dataset to cluster or group the records.

3.3.6. Clustering and Segmentation of records

After comparing records and acquiring the final dataset, it will be used to cluster records based on the selection criteria for the grouping as shown in Figure 3.6. The clustering will further identify matching records based on link analysis classified as "match", "possible match", and "close or related match". At this stage, the clustering does not require labelling data from a human expert to group similar records. Because the Mean-Shift clustering algorithm has been used to group similar records based on age, name aggregate score, and address score. Using the Mean-Shift algorithm gives flexibility in the clustering that does not require a fixed number of clusters to be defined. Each record will be labelled automatically with a cluster number during this clustering process. These clustered records will then be matched, compared and filtered based on the highest name and address score to create segments of records. It will ensure that similar records are linked together even in different clusters. These records will be matched for similar addresses from the initial retrieved and clustered datasets in the segmentation process, as seen in Figure 3.6. The similar segmented records will be merged into one dataset, and any relevant records will be kept separate. Any found duplicate entries for the same entity will be eliminated to reduce the number of matched records. For example, if a

record contains a similar “name” and “address” but a different “date of birth”, then in this case, it will be labelled as “possible match”. However, suppose a record contains a similar “date of birth” and “address” but a different name, then it will be labelled as a “close or related match”. The proposed model will efficiently avoid duplicate records and provide clean record linkage for a detailed resolution process.

3.3.7. Adding Graph Analysis

During the graph analysis, these segmented records will be compared with the clustered dataset to match the final identity out of all other identities. Figure 3.6 shows that the graph creation will be layer-based by using different attributes from the dataset to explore the matching records step by step visually. The first graph will be created using the entity name and cluster label for the entities, and it will visually represent all entities linked to each cluster in the clustered dataset. The second graph will use the entity name and address from the segmented dataset. The third graph will be created using the second graph data compared with the first graph to find the matched identity out of other identities, and the matched identity will be shown with associated addresses.

3.4. Tools for Implementation of the Proposed Model

The fuzzy approach utilises Soundex and Jaro-Winkler algorithms to calculate the aggregate score for names and the FuzzyWuzzy Python library using Edit-Distance partial token to score the other attributes, e.g. ethnicity description and address. The aim is to match names simply by using similarity metrics and analysing retrieved records for similarities using clustering, segmentation, and graph analysis. This fuzzy approach is implemented using Python 3.7 using PyCharm (community version) IDE, and the anonymized policing dataset is stored in MS SQL Server Express 2017. Pandas (Python data analysis library) cleans data and stores datasets retrieved during different stages. The NetworkX library is used for graph analysis and visualization.

The proposed approach requires implementation and computer programming language, tools, and machine learning packages to achieve the proposed result. The details of the tools used to implement the proposed model have been added to Appendix B.

3.5. Implementation of Soundex with Improvements

3.5.1. Standard Soundex Code Algorithm

The following is the algorithm of the standard Soundex code. It will generate a 4-character alphanumeric code for any name. The results of the standard code for each name are discussed in table format in the next section.

```
1. Define “Soundex” function for “name” to create code of “length 4”

2. Define “digit” mapping for letters A-Z as '01230120022455012623010202'

3. Initialise variables name as “soundex_code” and “first_character”

4. Convert name to uppercase and iterate through each character
5.    for character in name converted to uppercase
6.     if character is an alphabetic letter
7.      Store the first letter as the initial letter
8.     digit = digits[ASCII value of character - ASCII value of 'A']
9.      if soundex_code is empty or digit is not the same as the last digit
added:
10.     append digit to soundex_code

11. Replace the first digit with the initial letter and store in soundex_code

12. Remove all '0's from the soundex_code

13. Adjust the code to the required length 4 and remove additional characters
14. Return the result to the function Soundex
```

➲ *Standard Soundex (4-digit Code) Results*

Table 3.1 shows that the standard Soundex code is generated for the given names. The Soundex issues are added in the table below to show each name matching results accordingly.

Table 3.1 - Soundex 4-digit code for names representing Soundex issues

Soundex Issues	Names	Soundex Code
First Letter dependency	Corth , Korth	C630 , K630
Transcription difference	Ivanov , Ivanoff , Iwanow	I151 , I151 , I151
	Hsiao , Xiao	H200 , X000
Silent consonants	Meghburn , Meburn	M216 , M165
	Coghburn , Coburn	C216 , C165
	Deighton , Dayton	D235 , D350
Name syntax	Sheikh Ali Mohamed , Sheikh Ali	S245 , S240
	Sheikh Ali Mohamed , Sheikh Mohamed	S245 , S255
	Ali Mohamed , Sheikh Ali Mohamed	A455 , S245
	Alhameed , Hameed	A453 , H530
	Alhameed , Hamid	A453 , H530
Name Initials	Mikhail Kovalchuk , M.Kovalchuk,	M242 , M214
Poor precision	Criton. , Courtmanche	C635 , C635
	Criton. , Corradino	C635 , C635
	Criton. , Cartmill	C635 , C635
	Criton. , Cortinez	C635 , C635
Noise intolerance	Hreman , Herman	H655 , H655
	Hubbins , Huggins,	H152 , H252
	Msith , Smith	M230 , S530

	Smith , Smythe	S530 , S530
Long Name	Rameshwar , Rameshwaram	R526 , R526

In the standard Soundex algorithm, the code produced to match the names retains the first character from the names. The remaining alphabets in the names are converted into digits. The first letter dependency issue mismatches the names starting with different letters. If there is a typo error, then the name does not match. For example, “Corth” and “Korth” do not match names based on the standard Soundex code. The other issues of Standard Soundex code matching can be seen in the table.

3.5.2. Improved Soundex 6-digit Code Algorithm

The Soundex code algorithm has been modified by removing the first character of a name as a constant letter from the code and changing the length to 6 digits. It generates a numerical Soundex code of 6-digit code to help eliminate the Soundex first character mismatch issue. Increasing the code length helps reduce many false-positive retrievals compared to the 4-digit code. The results of this improved Soundex 6-digit code are presented in the next section. The following is the algorithm of the improved Soundex 6-digit code:

```
1. Define “Soundex” function for “name” to create code of “length 6”

2. Define “digit” mapping for letters A-Z as '01230120022455012623010202'

3. Initialise variables name as “soundex_code” and “first_character”

4. Convert name to uppercase and iterate through each character
5.    for character in name converted to uppercase
6.     if character is an alphabetic letter
7.     digit = digits[ASCII value of character - ASCII value of 'A']
8.      if soundex_code is empty or digit is not the same as the last digit
added:
9.     append digit to soundex_code

10. Remove all '0's from the soundex_code
```

```
11. Adjust the code to the required length 6 and remove additional characters
12. Return the result to the function Soundex
```

➲ *Improved Soundex 6-digit Code Results*

Table 3.2 - Improved Soundex 6-digit code for names representing Soundex issues

Soundex Issues	**Names**	**Soundex Code (Modified)**
First Letter dependency	Corth , Korth	263000 , 263000
Transcription difference	Ivanov , Ivanoff , Iwanow	151000 , 151000
	Hsiao , Xiao	200000 , 200000
Silent consonants	Meghburn , Meburn	521650 , 516500
	Coghburn , Coburn	221650 , 216500
	Deighton , Dayton	323500 , 335000
Name syntax	Sheikh Ali Mohamed , Sheikh Ali	224553 , 224000
	Sheikh Ali Mohamed , Sheikh Mohamed	224553 , 225530
	Ali Mohamed , Sheikh Ali Mohamed	455300 , 224553
	Alhameed , Hameed	453000 , 530000
	Alhameed , Hamid	453000 , 530000
Name Initials	Mikhail Kovalchuk , M.Kovalchuk,	524214 , 521422
Poor precision	Criton. , Courtmanche	263500 , 263552
	Criton. , Corradino	263500 , 263500
	Criton. , Cartmill	263500 , 263540

	Criton. , Cortinez	263500 , 263520
Noise intolerance	Hreman , Herman Hubbins , Huggins, Msith , Smith Smith , Smythe	655000 , 655000 152000 , 252000 523000 , 253000 253000 , 253000
Long Name	Rameshwar , Rameshwaram	652600 , 652650

It is worth noting that the code implementation brought different results compared to the standard Soundex code. The First letter dependency has been eliminated to fuzzy match the strings. However, most notably, the poor precision strings are not matched because the codes generated are different for each pair. It also improved the matching of long names and helped reduce the noise tolerance issue of the standard Soundex algorithm.

3.6. String Matching with 6-digit Soundex code

The improved 6-digit Soundex code shows the improvement over the standard Soundex code by addressing most of the issues discussed in previous sections. Table 3.3 provides the matching status of each pair of strings to better understand the results of the improved Soundex 6-digit code algorithm.

Table 3.3 - Improved Soundex 6-digit code for names with Algorithm Matching status

Soundex Issues	**Names**	**Soundex Code (Modified)**	**Matching Status**
First Letter dependency	Corth , Korth	263000 , 263000	Possible Match

Transcription difference	Ivanov , Ivanoff , Iwanow Hsiao , Xiao	151000 , 151000 200000 , 200000	Possible Match Close Match
Silent consonants	Meghburn , Meburn Coghburn , Coburn Deighton , Dayton	521650 , 516500 221650 , 216500 323500 , 335000	Possible Match Possible Match Close Match
Name syntax	Sheikh Ali Mohamed , Sheikh Ali Sheikh Ali Mohamed , Sheikh Mohamed Ali Mohamed , Sheikh Ali Mohamed Alhameed , Hameed Alhameed , Hamid	224553 , 224000 224553 , 225530 455300 , 224553 453000 , 530000 453000 , 530000	Close Match Possible Match Close Match Close Match Close Match
Name Initials	Mikhail Kovalchuk , M.Kovalchuk,	524214 , 521422	Close Match
Poor precision	Criton. , Courtmanche Criton. , Corradino Criton. , Cartmill Criton. , Cortinez	263500 , 263552 263500 , 263500 263500 , 263540 263500 , 263520	Close Match Close Match Close Match Close Match
Noise intolerance	Hreman , Herman Hubbins , Huggins,	655000 , 655000 152000 , 252000	Possible Match Close Match

	Msith , Smith Smith , Smythe	523000 , 253000 253000 , 253000	Close Match Possible Match
Long Name	Rameshwar , Rameshwaram	652600 , 652650	Possible Match

The matching status for most of the pairs of strings helps to fuzzy match them rather than complete mistakes compared to standard Soundex code. The poor precision is closed-matched if the matching threshold is set to 0.70. However, if the matching threshold is lower than 0.70, these pairs of strings will not be matched as close or matches. At the same time, the matching status of other Standard Soundex issues is improved.

3.7. Summary

The research methodology has been explained in this chapter. The methods used to form the proposed algorithm involve data preparation, labelling, clustering, records linkage, and data selection based on set criteria with data segmentation. Following this detail, the proposed model was explained, and each phase was presented in detail. Different tools and libraries that helped implement the proposed model were discussed in detail. The chapter's last sections discussed the implementation of matching algorithms and introduced improvements in the Soundex algorithm. At the end of the chapter, the matching of different strings was discussed using an improved Soundex algorithm. The next chapter uses these matching algorithms and the proposed model to provide the computer-simulated results.

CHAPTER 4

4. RESULTS – DATA ANALYSIS OF POLICING DATASET & COMPUTER SIMULATION

4.1. Chapter Overview

The chapter is divided into four sections, which provide the analysis and discuss the results. These results will be generated based on the proposal model in Chapter 3. The chapter starts by evaluating the matching of different language name variations using improved Soundex algorithms and the aggregate score for each name. For this reason, English, Arabic, Russian and other mixed names will be used. The matching results and the performance of each language name will be discussed. Following this, the evaluation will be conducted on the de-identified policing dataset, but before that, the policing dataset will be explained and analysed. The dataset will be evaluated using computer simulation by performing three individual searches. The results of each search will be analysed in detail, and the final section of the chapter will evaluate each search result's performance measures. The measures of success for this research on identity resolution would be based on several key criteria, reflecting the proposed approach's effectiveness, efficiency, and impact. It will include assessing the effectiveness of the aggregate matching score to retrieve records, flexibility and adaptability of the proposed model to handle the variation of data, precision, recall and accuracy of the matching names and quality of the clustering of records matching.

4.2. Evaluation of Names Variations

It is significant for entity matching in an extensive database to match names of different languages. At the same time, identify no match from the results. The classification of results is needed to do this, and the aggregate score algorithm can be applied to achieve this. The aim is to find matches as true positives and reduce false positives and false negatives where needed. For this purpose, a confusion matrix will be applied to evaluate the performance of the aggregate score algorithm. Therefore, some true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) calculations will help to measure the matching outcome and provide the precision, recall and accuracy of the results. The results will be placed in the following four categories. They will help to calculate the precision, recall and accuracy of the overall results:

- ➲ True positives (TP): When the match is a true match detected by the algorithm. It means the entity name correctly matches the actual name.

- ➲ True negative (TN): When the non-match is a non-match detected by the algorithm. It means the entity name is a true non-match with the actual name and detected correctly.

- ➲ False positives (FP): When the algorithm detects the match as a true match with a non-match. It means a false match is found that should not be matched with the actual name.

- ➲ False negatives (FN). When the algorithm detects a non-match with a match, it should be found as a match but detected incorrectly.

To assess the algorithm's overall performance through a single value measurement, precision, recall, and accuracy will be derived from the four values in the confusion matrix. Precision provides the measure of the accuracy of the positive predictions. It is the ratio of correctly predicted positive observations to the total predicted positives. Recall measures the ability of the model to capture all the relevant instances of a class. It is the ratio of correctly predicted positive observations to the actual positives.

Accuracy provides the overall correctness of the model. It is the ratio of correctly predicted instances to the total number of instances.

The following sections are the computer simulation results to compare names and find similarities. It shows the performances of the proposed algorithm by generating these matching results. Therefore, before implementing the proposed algorithm on the policing dataset, it is worth discussing and evaluating the results of different name matching using improved Soundex code using the proposed algorithm.

4.2.1. Evaluation of English Names

In Figure 4.1, the 17 English names are selected from the previous work by (Winkler, 1994) to evaluate the names' similarities. These names are fed into the proposed matching algorithm and are compared here using modified Soundex code with Jaro-Winkler similarity metrics.

Names matching codes & scores:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
0	SHACKLEFORD	224163	SHACKELFORD	224163	1.00	0.98	0.99	Possible Match
1	DUNNINGHAM	355250	CUNNIGHAM	252500	0.69	0.90	0.79	Close Match
2	NICHLESON	524250	NICHULSON	524250	1.00	0.96	0.98	Possible Match
3	MASSEY	520000	MASSIE	520000	1.00	0.93	0.97	Possible Match
4	JONES	252000	JOHNSON	252500	0.92	0.83	0.88	Close Match
5	ABROMS	165200	ABRAMS	165200	1.00	0.92	0.96	Possible Match
6	HARDIN	635000	MARTINEZ	563520	0.65	0.72	0.69	No Match
7	ITMAN	355000	SMITH	253000	0.82	0.47	0.65	No Match
8	JERALDINE	264350	GERALDINE	264350	1.00	0.93	0.97	Possible Match
9	MARHTA	563000	MARTHA	563000	1.00	0.96	0.98	Possible Match
10	MICHELLE	524000	MICHAEL	524000	1.00	0.92	0.96	Possible Match
11	JULIES	242000	JULIUS	242000	1.00	0.93	0.97	Possible Match
12	TANYA	350000	TONYA	350000	1.00	0.88	0.94	Possible Match
13	DWAYNE	350000	DUANE	350000	1.00	0.84	0.92	Possible Match
14	SEAN	250000	SUSAN	225000	0.90	0.80	0.85	Close Match
15	JON	250000	JOHN	250000	1.00	0.93	0.97	Possible Match
16	JON	250000	JAN	250000	1.00	0.80	0.90	Possible Match

Figure 4.1 - English names variation with the proposed algorithm

All these names have different scores listed, but to clarify the results further, the results are separated into three sections, as shown below in Figure 4.2.

Names with Aggregate score higher than JW Score and different Soundex Codes:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
4	JONES	252000	JOHNSON	252500	0.92	0.83	0.88	Close Match
7	ITMAN	355000	SMITH	253000	0.82	0.47	0.65	No Match
14	SEAN	250000	SUSAN	225000	0.90	0.80	0.85	Close Match

Names with Aggregate score Less than JW Score and different Soundex Codes:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
1	DUNNINGHAM	355250	CUNNIGHAM	252500	0.69	0.90	0.79	Close Match
6	HARDIN	635000	MARTINEZ	563520	0.65	0.72	0.69	No Match

Names with equal Aggregate & JW score and different Soundex Codes:

Empty DataFrame

Figure 4.2 - English names breakdown of matching with and without the same Soundex code

The first three names in Figure 4.2 have higher aggregate scores compared to the Jaro Winkler Score. At the same time, they have different Soundex codes. Jaro-Winkler scored lower than Soundex and the aggregate score for these three names. However, looking at these names closely, it can be observed that these “3” names are different and should not be matched with a higher score. The aggregate score generated for the names matches them as a close match for two names. The last two names in Figure 4.2 have an aggregate score lower than the Jaro-Winkler score. The first name is labelled as a close match as the names are similar yet different. In comparison, the second name is labelled as no match. It is lower than the Jaro-Winkler score because the names are not similar.

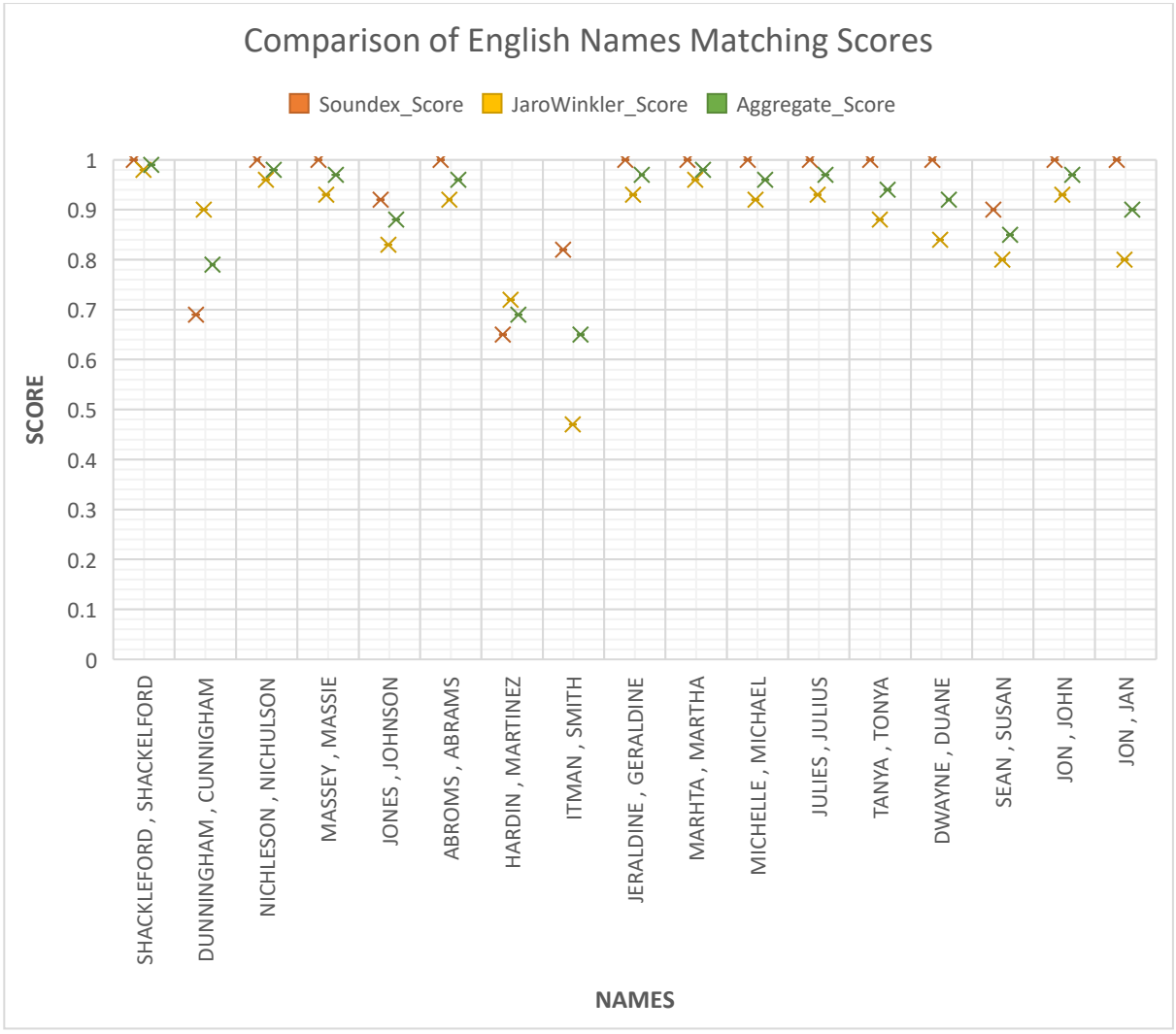

Figure 4.3 - Matching Scores Comparison of English Names

The score comparison graph in Figure 4.3 shows the name match Soundex, Jaro-Winkler, and Aggregate score. Based on the fuzzy string matching requirements, the aggregate score approximately matches the names instead of giving the value of 1 to show a 100% match. Therefore, in this instance, this is the correct way to match these names to find the best match.

Names matching codes & scores:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
0	SHACKLEFORD	224163	SHACKELFORD	224163	1.00	0.98	0.99	Possible Match
9	MARHTA	563000	MARTHA	563000	1.00	0.96	0.98	Possible Match
2	NICHLESON	524250	NICHULSON	524250	1.00	0.96	0.98	Possible Match
15	JON	250000	JOHN	250000	1.00	0.93	0.97	Possible Match
11	JULIES	242000	JULIUS	242000	1.00	0.93	0.97	Possible Match
8	JERALDINE	264350	GERALDINE	264350	1.00	0.93	0.97	Possible Match
3	MASSEY	520000	MASSIE	520000	1.00	0.93	0.97	Possible Match
5	ABROMS	165200	ABRAMS	165200	1.00	0.92	0.96	Possible Match
10	MICHELLE	524000	MICHAEL	524000	1.00	0.92	0.96	Possible Match
12	TANYA	350000	TONYA	350000	1.00	0.88	0.94	Possible Match
13	DWAYNE	350000	DUANE	350000	1.00	0.84	0.92	Possible Match
16	JON	250000	JAN	250000	1.00	0.80	0.90	Possible Match
4	JONES	252000	JOHNSON	252500	0.92	0.83	0.88	Close Match
14	SEAN	250000	SUSAN	225000	0.90	0.80	0.85	Close Match
1	DUNNINGHAM	355250	CUNNIGHAM	252500	0.69	0.90	0.79	Close Match
6	HARDIN	635000	MARTINEZ	563520	0.65	0.72	0.69	No Match
7	ITMAN	355000	SMITH	253000	0.82	0.47	0.65	No Match

Figure 4.4 – Performance of Matching Scores of English Names

Figure 4.4 is colour-coded based on the Figure 4.5 confusion matrix values for a better understanding. Based on these values, the precision, recall and accuracy have been calculated as shown below:

Figure 4.5 – Confusion Matrix for English Names Comparison

The Precision is calculated as TP / (TP + FP), and the result is 0.79. The Recall is calculated as TP / (TP + FN), and the result is 0.92. The accuracy is calculated as TP+TN / TP+TN+FP+FN, and the result is 0.76. Figure 4.6 below shows the overall result of the comparison of algorithm performance.

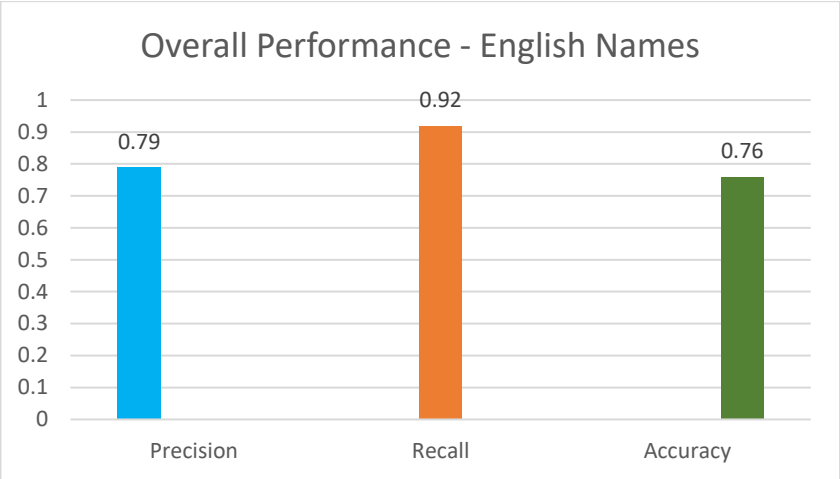

Figure 4.6 – Overall Performance Evaluation of English Names

4.2.2. Evaluation of Selection of Mixed Names

Figure 4.7 lists the random names of different languages for similarity matching. These 22 names are fed into the proposed matching algorithm, which combines English, Arabic, and Russian names and abbreviations. These names are compared here using modified Soundex code and Jaro-Winkler similarity metrics.

Names matching codes & scores:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
0	Abdal-Rachid	134623	Abdur-rashide	136230	0.91	0.80	0.85	Close Match
1	Catherine	236500	Katherine	236500	1.00	0.93	0.97	Possible Match
2	ADELINE	345000	LINE	450000	0.89	0.46	0.68	No Match
3	MCMANUS	525520	MACMANUS	525520	1.00	0.90	0.95	Possible Match
4	WILL SMITH	425300	WILL SMITH YOUNG	425352	0.87	0.93	0.90	Possible Match
5	SMIT	253000	S.M.I.T	253000	1.00	0.75	0.88	Close Match
6	WILL SMITH	425300	WILL-SMITH	425300	1.00	0.96	0.98	Possible Match
7	Mikhail Kovalchuk	524214	M.Kovalchuk,	521422	0.83	0.69	0.76	Close Match
8	Ivanov	151000	Iwanow	500000	0.78	0.80	0.79	Close Match
9	Ivanoff	151000	Iwanow	500000	0.78	0.77	0.78	Close Match
10	Ivanoff	151000	Ivanov	151000	1.00	0.91	0.96	Possible Match
11	BOBBY	110000	BOB	110000	1.00	0.91	0.96	Possible Match
12	ROB	610000	ROBBIN	615000	0.91	0.88	0.90	Close Match
13	Sinclair	252460	St. Clair	232460	0.90	0.75	0.82	Close Match
14	MAXIME	525000	MAXIMIEN	525500	0.92	0.95	0.94	Possible Match
15	Christina	262350	Tina	350000	0.44	0.45	0.45	No Match
16	Pooh	100000	Puh	100000	1.00	0.75	0.88	Close Match
17	Incorporated	526163	Inc.	520000	0.56	0.67	0.61	No Match
18	Korth	263000	Corth	263000	1.00	0.87	0.94	Possible Match
19	Meghburn	521650	Meburn,	516500	0.90	0.90	0.90	Possible Match
20	Herman	655000	Hreman	655000	1.00	0.95	0.97	Possible Match
21	Smith	253000	Smythe	253000	1.00	0.86	0.93	Possible Match

Figure 4.7 - Mixed name variation with the proposed algorithm

All names have different scores listed, but to clarify the results further, the results are separated into three sections, as shown below in Figure 4.8.

Names with Aggregate score higher than JW Score and different Soundex Codes:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
0	Abdal-Rachid	134623	Abdur-rashide	136230	0.91	0.80	0.85	Close Match
2	ADELINE	345000	LINE	450000	0.89	0.46	0.68	No Match
7	Mikhail Kovalchuk	524214	M.Kovalchuk,	521422	0.83	0.69	0.76	Close Match
9	Ivanoff	151000	Iwanow	500000	0.78	0.77	0.78	Close Match
12	ROB	610000	ROBBIN	615000	0.91	0.88	0.90	Close Match
13	Sinclair	252460	St. Clair	232460	0.90	0.75	0.82	Close Match

Names with Aggregate score Less than JW Score and different Soundex Codes:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
4	WILL SMITH	425300	WILL SMITH YOUNG	425352	0.87	0.93	0.90	Possible Match
8	Ivanov	151000	Iwanow	500000	0.78	0.80	0.79	Close Match
14	MAXIME	525000	MAXIMIEN	525500	0.92	0.95	0.94	Possible Match
17	Incorporated	526163	Inc.	520000	0.56	0.67	0.61	No Match

Names with equal Aggregate & JW score and different Soundex Codes:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
15	Christina	262350	Tina	350000	0.44	0.45	0.45	No Match
19	Meghburn	521650	Meburn,	516500	0.90	0.90	0.90	Possible Match

Figure 4.8 - Mixed names breakdown of matching with and without the same Soundex code

The first six names are in Figure 4.8, and they have higher aggregate scores than the Jaro-Winkler score with different Soundex codes. For these six names, Jaro-Winkler scored lower than Soundex and aggregate score. It can be observed that “Adeline” and “Line” are different strings and should not be matched with a higher score. The other “5” names are matched as close matches with aggregate scores. The middle four names in Figure 4.8. have an aggregate score lower than the Jaro-Winkler score. The two names are labelled as Possible matches because these names are similar. One name is labelled as close matched while another is labelled as no match with a lower score than the Jaro-Winkler score, but this should match as this is an abbreviated word. The last two names got similar Soundex, Jaro-Winkler, and aggregate scores. One name is labelled as no match, and it is correct to be a no match because names are different. The other name is labelled as a possible match. It is good to be picked up with a better score because the

name can be mistyped or misspelt in the dataset. However, based on fuzzy matching, the algorithm will pick it as a possible match.

The score comparison graph in Figure 4.9 below shows the name match Soundex, Jaro-Winkler and aggregate score. Based on the fuzzy matching requirements, the aggregate score approximately matches the names instead of giving the value of 1 to show a 100% match. In this instance, that is the correct way to match these names.

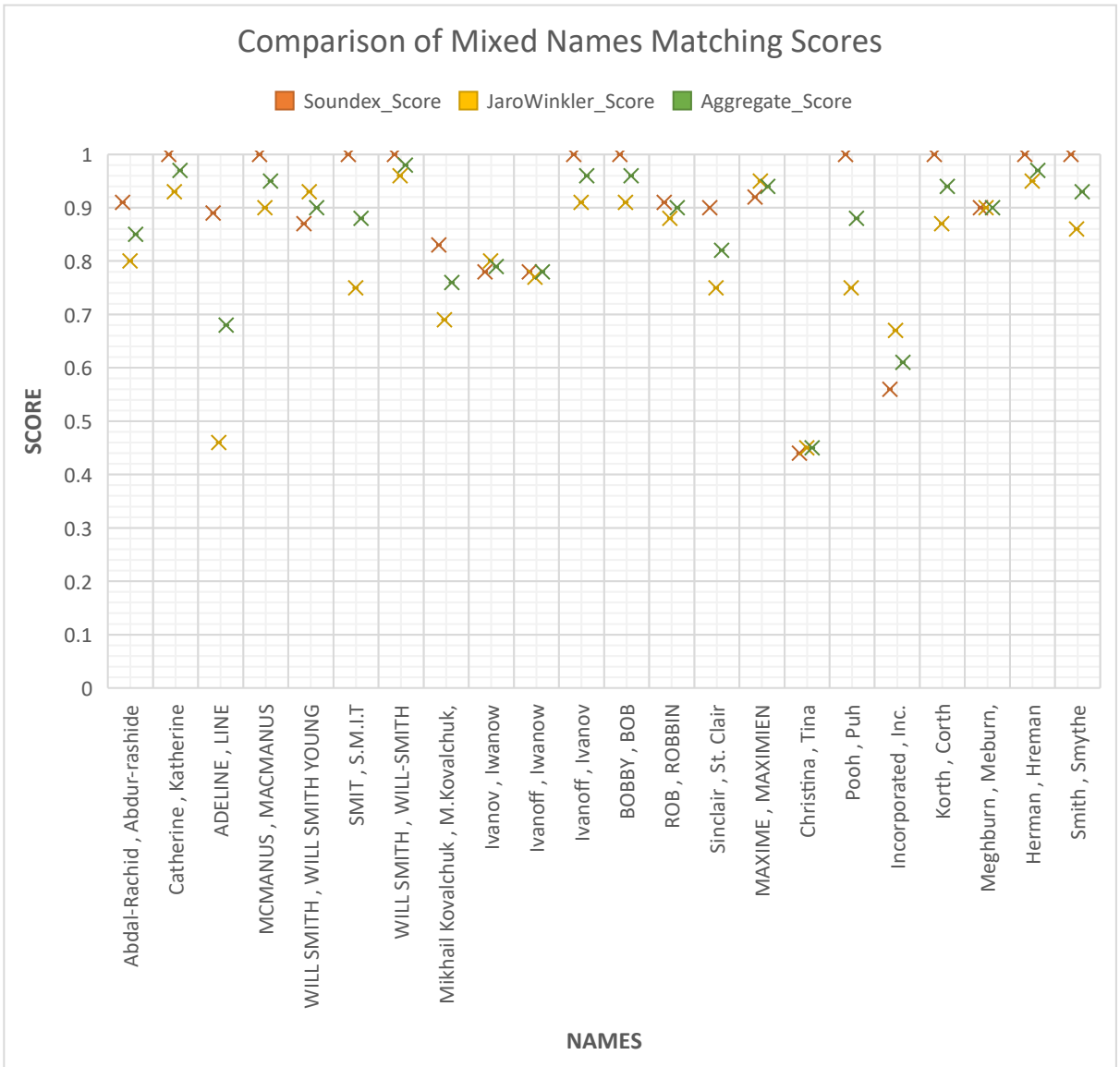

Figure 4.9 - Matching Scores Comparison of Mixed Names

Names matching codes & scores:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
6	WILL SMITH	425300	WILL-SMITH	425300	1.00	0.96	0.98	Possible Match
20	Herman	655000	Hreman	655000	1.00	0.95	0.97	Possible Match
1	Catherine	236500	Katherine	236500	1.00	0.93	0.97	Possible Match
11	BOBBY	110000	BOB	110000	1.00	0.91	0.96	Possible Match
10	Ivanoff	151000	Ivanov	151000	1.00	0.91	0.96	Possible Match
3	MCMANUS	525520	MACMANUS	525520	1.00	0.90	0.95	Possible Match
18	Korth	263000	Corth	263000	1.00	0.87	0.94	Possible Match
14	MAXIME	525000	MAXIMIEN	525500	0.92	0.95	0.94	Possible Match
21	Smith	253000	Smythe	253000	1.00	0.86	0.93	Possible Match
12	ROB	610000	ROBBIN	615000	0.91	0.88	0.90	Close Match
4	WILL SMITH	425300	WILL SMITH YOUNG	425352	0.87	0.93	0.90	Possible Match
19	Meghburn	521650	Meburn,	516500	0.90	0.90	0.90	Possible Match
5	SMIT	253000	S.M.I.T	253000	1.00	0.75	0.88	Close Match
16	Pooh	100000	Puh	100000	1.00	0.75	0.88	Close Match
0	Abdal-Rachid	134623	Abdur-rashide	136230	0.91	0.80	0.85	Close Match
13	Sinclair	252460	St. Clair	232460	0.90	0.75	0.82	Close Match
8	Ivanov	151000	Iwanow	500000	0.78	0.80	0.79	Close Match
9	Ivanoff	151000	Iwanow	500000	0.78	0.77	0.78	Close Match
7	Mikhail Kovalchuk	524214	M.Kovalchuk,	521422	0.83	0.69	0.76	Close Match
2	ADELINE	345000	LINE	450000	0.89	0.46	0.68	No Match
17	Incorporated	526163	Inc.	520000	0.56	0.67	0.61	No Match
15	Christina	262350	Tina	350000	0.44	0.45	0.45	No Match

Figure 4.10 – Performance of Matching Scores of Mixed Names

Figure 4.10 is colour-coded based on the Figure 4.11 confusion matrix values for a better understanding. Based on these values, the precision, recall and accuracy have been calculated as shown below:

Figure 4.11 – Confusion Matrix for Mixed Names Comparison

The Precision is calculated as TP / (TP + FP), and the result is 0.95. while the Recall is calculated as TP / (TP + FN), and the result is 0.95. The accuracy is calculated as TP+TN / TP+TN+FP+FN, and the result is 0.90. Figure 4.12 below shows the overall result of the comparison of algorithm performance.

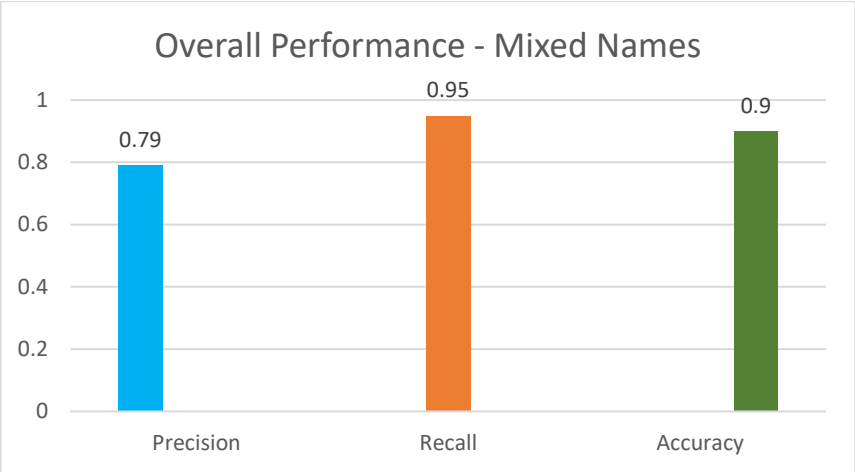

Figure 4.12 – Overall Performance Evaluation of Mixed Names

4.2.3. Evaluation of Arabic Names

In Figure 4.13, Arabic names are selected and fed into the proposed matching algorithm. These 15 names are taken to compare using modified Soundex code and Jaro-Winkler similarity metrics.

Names matching codes & scores:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
0	Abdal-Rachid	134623	Abdur-rashid	136230	0.91	0.82	0.86	Close Match
1	KADER	236000	QADIR	236000	1.00	0.73	0.86	Close Match
2	ABD AL-RAHMAN	134655	ABDULREHMAAN	134655	1.00	0.84	0.92	Possible Match
3	ABD AL-AZIZ	134220	ABDELAZIZ	134220	1.00	0.88	0.94	Possible Match
4	Hussein	250000	HUSAYN	250000	1.00	0.44	0.72	Close Match
5	MOHAMED	553000	MUHAMMAD	553000	1.00	0.80	0.90	Possible Match
6	ABU AL-FADL	141340	AFAZL	124000	0.67	0.62	0.65	No Match
7	BRAHIM	165000	ABRAHIM	165000	1.00	0.87	0.94	Possible Match
8	DAUD	330000	DAWOOD	330000	1.00	0.80	0.90	Possible Match
9	HABIBULLAH	114000	HABIB	110000	0.91	0.90	0.91	Possible Match
10	IZZ UD DIN	235000	AZIZ UD-DEEN	223500	0.90	0.78	0.84	Close Match
11	JAMAL AL-DIN	254435	KAMAL UD-DIN	254350	0.92	0.80	0.86	Close Match
12	NOUREDDINE	563500	NUR AD-DIN	563500	1.00	0.82	0.91	Possible Match
13	ZAAHIR	260000	ZAHEER	260000	1.00	0.82	0.91	Possible Match
14	ZAKARIA	226000	ZAKARIYYA	226000	1.00	0.96	0.98	Possible Match

Figure 4.13 - Arabic name variation with the proposed algorithm

All names have different scores listed, but to clarify the results further, the results are separated into three sections, as shown below in Figure 4.14.

Names with Aggregate score higher than JW Score and different Soundex Codes:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
0	Abdal-Rachid	134623	Abdur-rashid	136230	0.91	0.82	0.86	Close Match
6	ABU AL-FADL	141340	AFAZL	124000	0.67	0.62	0.65	No Match
9	HABIBULLAH	114000	HABIB	110000	0.91	0.90	0.91	Possible Match
10	IZZ UD DIN	235000	AZIZ UD-DEEN	223500	0.90	0.78	0.84	Close Match
11	JAMAL AL-DIN	254435	KAMAL UD-DIN	254350	0.92	0.80	0.86	Close Match

Names with Aggregate score Less than JW Score and different Soundex Codes:
Empty DataFrame

Names with equal Aggregate & JW score and different Soundex Codes:
Empty DataFrame

Figure 4.14 - Arabic names breakdown of matching with and without the same Soundex code

The names in Figure 4.14 have higher aggregate scores than the Jaro-Winkler score. At the same time, they have different Soundex codes. Three names are labelled as close matches, and Jaro-Winkler has scored lower than the aggregate. Only one name is labelled as no match as it has been scored lower. It can be observed that this name is a variation of another name and should match as a possible or close match. It should be like the other name, where names are labelled as possible matches. The aggregate score generated for the name is higher than Jaro-Winkler and equivalent to the Soundex code.

The score comparison graph in Figure 4.15 below shows the name match Soundex, Jaro-Winkler, and Aggregate score. Based on the fuzzy matching requirements, the aggregate score approximately matches the names instead of giving the value of 1 to show a 100% match in this instance; that is the correct way to match these names.

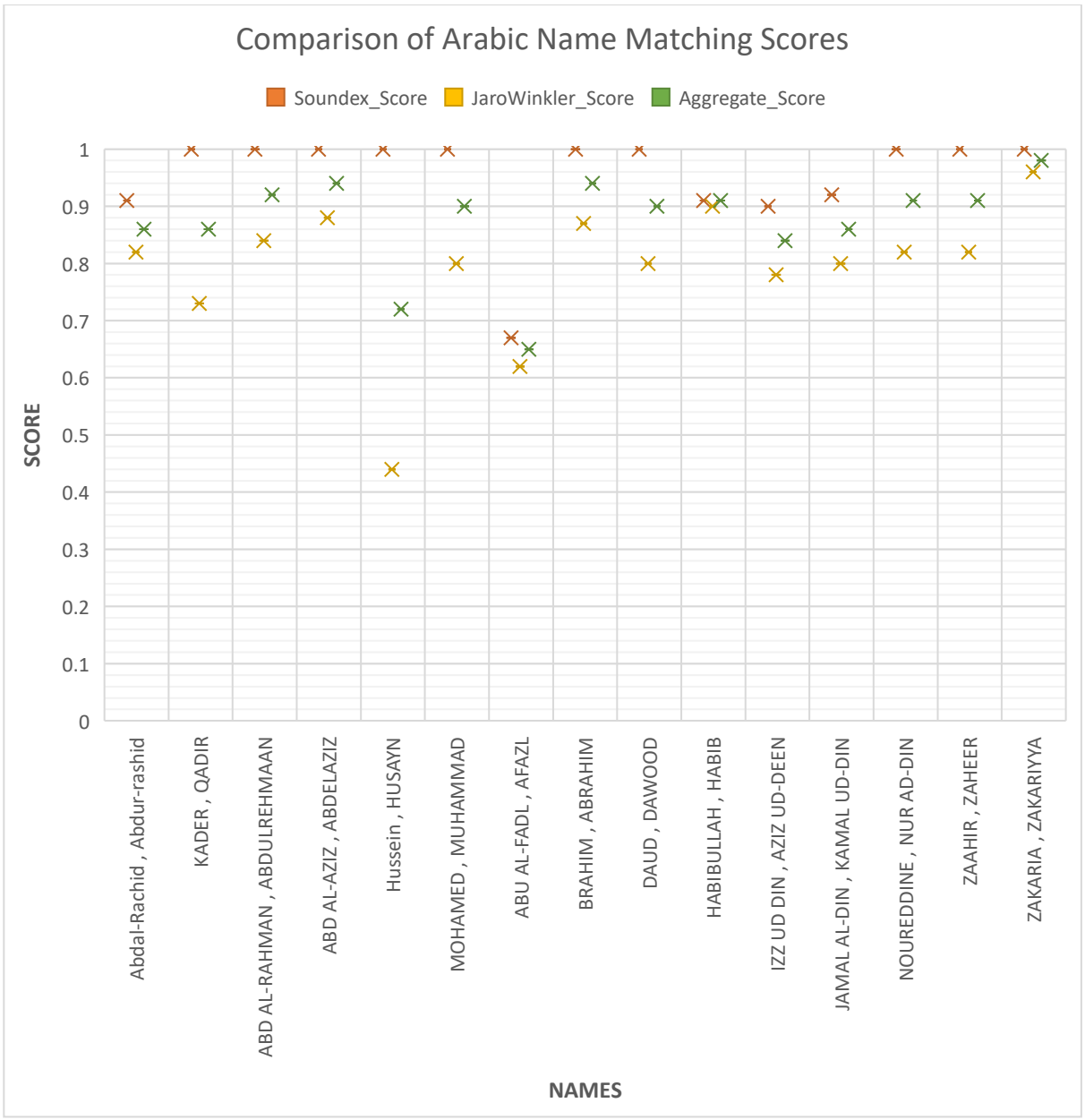

Figure 4.15 - Matching Scores Comparison of Arabic Names

Names matching codes & scores:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
14	ZAKARIA	226000	ZAKARIYYA	226000	1.00	0.96	0.98	Possible Match
3	ABD AL-AZIZ	134220	ABDELAZIZ	134220	1.00	0.88	0.94	Possible Match
7	BRAHIM	165000	ABRAHIM	165000	1.00	0.87	0.94	Possible Match
2	ABD AL-RAHMAN	134655	ABDULREHMAAN	134655	1.00	0.84	0.92	Possible Match
9	HABIBULLAH	114000	HABIB	110000	0.91	0.90	0.91	Possible Match
12	NOUREDDINE	563500	NUR AD-DIN	563500	1.00	0.82	0.91	Possible Match
13	ZAAHIR	260000	ZAHEER	260000	1.00	0.82	0.91	Possible Match
5	MOHAMED	553000	MUHAMMAD	553000	1.00	0.80	0.90	Possible Match
8	DAUD	330000	DAWOOD	330000	1.00	0.80	0.90	Possible Match
0	Abdal-Rachid	134623	Abdur-rashid	136230	0.91	0.82	0.86	Close Match
1	KADER	236000	QADIR	236000	1.00	0.73	0.86	Close Match
11	JAMAL AL-DIN	254435	KAMAL UD-DIN	254350	0.92	0.80	0.86	Close Match
10	IZZ UD DIN	235000	AZIZ UD-DEEN	223500	0.90	0.78	0.84	Close Match
4	Hussein	250000	HUSAYN	250000	1.00	0.44	0.72	Close Match
6	ABU AL-FADL	141340	AFAZL	124000	0.67	0.62	0.65	No Match

Figure 4.16 – Performance of Matching Scores of Arabic Names

Figure 4.16 is colour-coded based on the Figure 4.17 confusion matrix values for a better understanding. Based on these values, the precision, recall and accuracy have been calculated as shown below:

Figure 4.17 – Confusion Matrix for Arabic Names Comparison

The Precision is calculated as TP / (TP + FP), and the result is 1.0. while the Recall is calculated as TP / (TP + FN), and the result is 0.93. The accuracy is calculated as TP+TN / TP+TN+FP+FN, and the result is 1.0. Figure 4.18 below shows the overall result of the comparison of algorithm performance.

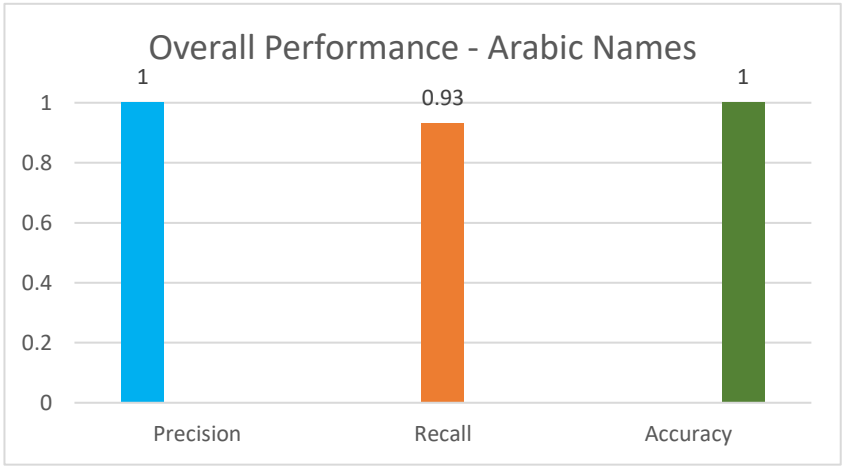

Figure 4.18 – Overall Performance Evaluation of Arabic Names

4.2.4. Evaluation of Russian Names

In Figure 4.19, Russian names are selected and fed into the proposed matching algorithm. These 9 names are compared here using modified Soundex code and Jaro-Winkler similarity metrics.

Names matching codes & scores:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
0	EKATERINA	236500	YEKATERINA	236500	1.00	0.97	0.98	Possible Match
1	ELENA	450000	YELENA	450000	1.00	0.94	0.97	Possible Match
2	JEKATERINA	223650	YEKATERINA	236500	0.90	0.93	0.92	Possible Match
3	LIOUBA	410000	LYUBA	410000	1.00	0.84	0.92	Possible Match
4	NASTASYA	523200	ANASTASIYA	523200	1.00	0.77	0.89	Close Match
5	MAX	520000	MAKS	520000	1.00	0.78	0.89	Close Match
6	Mikhail Kovalchuk	524214	M.Kovalchuk	521422	0.83	0.74	0.78	Close Match
7	Ivanov	151000	Iwanow	500000	0.78	0.80	0.79	Close Match
8	Ivanoff	151000	Ivanov	151000	1.00	0.91	0.96	Possible Match

Figure 4.19 - Russian name variation with the proposed algorithm

These results are separated into three sections to clarify the results further, for all the names have different Soudex codes, as shown below in Figure 4.20.

Names with Aggregate score higher than JW Score and different Soundex Codes:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
6	Mikhail Kovalchuk	524214	M.Kovalchuk	521422	0.83	0.74	0.78	Close Match

Names with Aggregate score Less than JW Score and different Soundex Codes:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
2	JEKATERINA	223650	YEKATERINA	236500	0.90	0.93	0.92	Possible Match
7	Ivanov	151000	Iwanow	500000	0.78	0.80	0.79	Close Match

Names with equal Aggregate & JW score and different Soundex Codes:

Empty DataFrame

Figure 4.20 - Russian names breakdown of matching with and without the same Soundex code

The first name in Figure 4.20 has a higher aggregate score than the Jaro Winkler Score, while it has different Soundex codes and has been labelled as a close match. However, the name Jaro-Winkler scored lower than Soundex and aggregate score. By close observation, the name is the same but with Full name and initials in the second variation.

The last two names in Figure 4.20 have an aggregate score lower than the Jaro-Winkler score. The first name is labelled as a possible match as names are similar but are possibly similar or different names. While the second name is labelled as a close match and has a score lower than the Jaro-Winkler score, it is still the same or a similar name to be matched.

The score comparison graph in Figure 4.21 below shows the name match Soundex, Jaro-Winkler and Aggregate score. Based on the fuzzy matching requirements, the aggregate score approximately matches the names instead of giving the value of 1 to show a 100% match in this instance; that is the correct way to match these names.

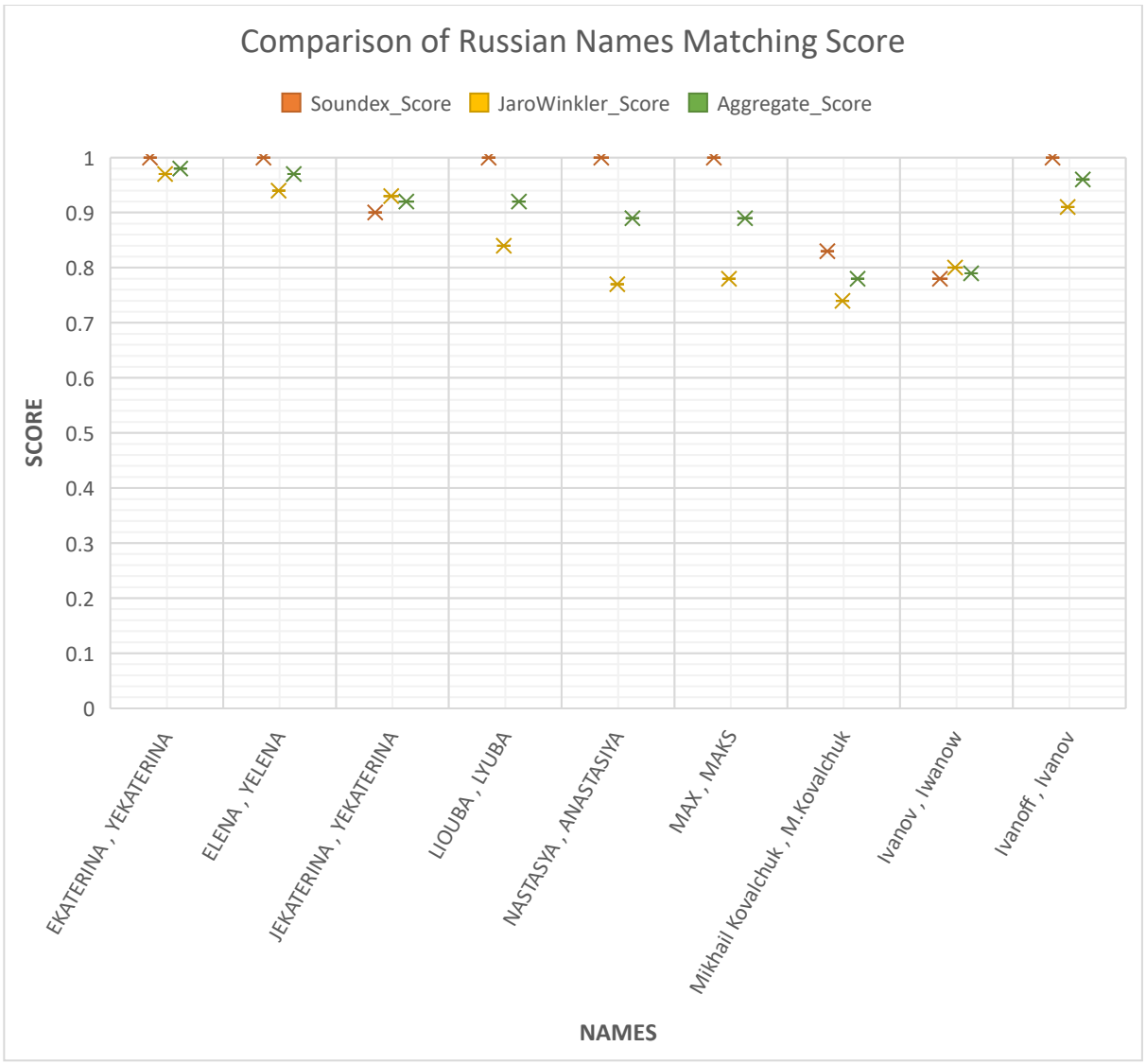

Figure 4.21 - Matching Scores Comparison of Russian Names

Names matching codes & scores:

	Name1	Soundex1	Name2	Soundex2	Soundex-Jaro Score	Jaro-Winkler Score	Aggregate score	Status
0	EKATERINA	236500	YEKATERINA	236500	1.00	0.97	0.98	Possible Match
1	ELENA	450000	YELENA	450000	1.00	0.94	0.97	Possible Match
8	Ivanoff	151000	Ivanov	151000	1.00	0.91	0.96	Possible Match
2	JEKATERINA	223650	YEKATERINA	236500	0.90	0.93	0.92	Possible Match
3	LIOUBA	410000	LYUBA	410000	1.00	0.84	0.92	Possible Match
4	NASTASYA	523200	ANASTASIYA	523200	1.00	0.77	0.89	Close Match
5	MAX	520000	MAKS	520000	1.00	0.78	0.89	Close Match
7	Ivanov	151000	Iwanow	500000	0.78	0.80	0.79	Close Match
6	Mikhail Kovalchuk	524214	M.Kovalchuk	521422	0.83	0.74	0.78	Close Match

Figure 4.22 – Performance of Matching Scores of Russian Names

For better understanding, figure 4.22 is colour-coded based on the Figure 4.23 confusion matrix values. Based on these values, the precision, recall and accuracy have been calculated as shown below:

Figure 4.23 – Confusion Matrix for Russian Names Comparison

The Precision is calculated as TP / (TP + FP), and the result is 1.0. while the Recall is calculated as TP / (TP + FN), and the result is 1.0. The accuracy is calculated as TP+TN / TP+TN+FP+FN, and the result is 1.0. Figure 4.24 below shows the overall result of the comparison of algorithm performance.

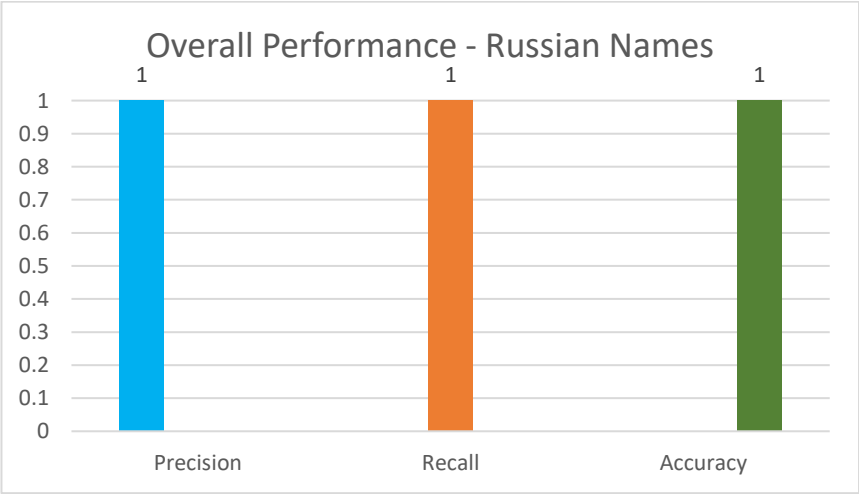

Figure 4.24 – Overall Performance Evaluation of Russian Names

4.3. Policing Dataset

This section will explore and analyse the de-identified police dataset. The proposed algorithm in Chapter 3 has been applied to this dataset. The details will be discussed later in this chapter. The policing dataset comprises the following fields about each entity. The names of each field listed here are precisely the same as in the dataset.

- **Crime_Ref:** This is a computer-generated unique reference number, which is continuous.
- **nominal_ref:** This is a computer-generated unique reference number, which is continuous.
- **surname:** Surname or family name.
- **forename :** First name(s).
- **sex:** Gender is "M" for Male or "F" for Female.
- **date_of_birth :** This is in dd/mm/yyyy format. A time aspect of 00:00:00 is attached to this field, which is not utilised.
- **ea_desc:** Police Identity Codes as outlined below or written text:
  - IC1 – White person, northern European/northern America type.
  - IC2 – Mediterranean European/Hispanic.
  - IC3 – African/Afro-Caribbean person.
  - IC4 – Indian, Pakistani, Nepalese, Maldivian, Sri Lankan, Bangladeshi, or other (South) Asian persons.
  - IC5 – Chinese, Japanese, or South-East Asian person.
  - IC6 – Middle Eastern, Arabic.
  - IC0, IC7 or IC9 – Origin unknown.
- **role_type :**
  - victims **(VICT)**;
  - defendants **(DEFE);**
  - persons who are probably responsible for an offence **(PROB);**
  - persons are known to be responsible for an offence **(RESP);**
  - suspects for committing an offence **(SUSP)**

- victims **(VICA).**

➲ **street_name:** The street name of the Nominal's home address that is manually entered. House number is excluded.

➲ **district_name:** District of Nominal's home address, which is manually entered.

➲ **town_name:** Town of Nominal's home address, which is manually entered.

➲ **Postcode_sector1:** The postcode of Nominal's home address is manually entered and encompasses 3,000 residences.

➲ **Beat_number1:** Computer allocated**.** It is a Local Policing Unit (LPU) sub-area. For example, E125 is Beat 25 in E1 LPU.

➲ **grid_ref_northing1:** This is a six-digit Ordnance Survey Grid Reference, which is a 1km block.

➲ **grid_ref_easting1:** This is a six-digit Ordnance Survey Grid Reference, which is a 1km block.

4.4. Policing Dataset Analysis

The data pre-processing phase discussed in Chapter 3 eliminates missing values from the dataset for each record. In the dataset, four different fields represent the address. So, the three fields are merged as one address field while keeping the postcode field separate. Similarly, the surname and forename are merged as the name field. The records defined as 'Unknown' gender rather than 'M' for males and 'F' for females are considered missing values. Overall, a total of 430,293 missing values are removed from the dataset. Therefore, after data cleaning, the dataset still has 715,919 records. All attribute values are converted to lowercase for standard matching.

In this fuzzy approach, name, gender, ethnicity, and address attributes are used to start the initial search of records. There are a total of 1146212 records in the dataset.

Table 4.1 - Analysis of Policing Dataset

Column Name	Total Records	Missing Records
Crime_Ref	1146212	0
Offence	1145418	794
HOMC_Code	1077337	68875
HOOC_Code	1077341	68871
nominal_ref	1146212	0
Surname	1146212	0
Forename	1145364	848
Gender	1146212	0
date_of_birth	1136404	9808
role_type	1146212	0
street_name	1137798	8414
town_name	1104356	41856
district_name	1002364	143848
postcode	1039866	106346
ea_desc	1126280	19932
grid_ref_northing	1146212	0
grid_ref_easting	1146212	0
beat_number	1146212	0

4.4.1. Multiple Nominal References for the same individual

When joining column Nominal_Ref to assign nominals to crimes and ascertaining their network activities, it would not obtain a complete picture of all individuals as many entities have multiple Nominal_Refs presented in the dataset.

Based on Nominal_Ref, Surname, Forename, dob and count Crime_Ref, there are 697773 records where flagged duplicate surnames, forenames, and dobs, and 12295 records are found in the dataset. There are 6032 individuals, each with 5 or fewer different Nominal_Refs. For example, Surname = BECK & Forename = JAUNETTE & dob

= 16/08/64, it retrieves 17 crimes over the 3 Nominal_Refs, joined the 6032 to the using Nominals ref to retrieve the number of crimes for each Surname, Forename, dob shows 27363 records. For example, Surname = HASKIN, Forename = ZIED dob = 19/11/99, by using the combination, 5 Nominal_Ref has 38 crime records.

4.4.2. Multiple DOBs for the same individual

Omitting Nominal_Refs and only using Surname, Forename and Date of Birth (DOB) as a compound key eliminates the issue above. However, many nominal records have the same names, but some have different DOBs. It can account for common names such as John Smith. However, there could be data errors, such as recording the UK vs international dates (01/12/1945 is the same as 12/01/1945). Aggregate on Surname, Forename, dob, and with flagged duplicates to ID all records with the same Surname and Forename retrieves 309518 records. To check the different dob associated with the same details, e.g. Surname = ABBIDAH, Forename = FAROS, retrieves 12 records where 10 are related to the same person across 3 different dobs judged by the addresses. Entities comprise the complete list of all persons associated with the crime data set – a victim, offender, witness, or suspect. Each entity is identified under a separate role.

4.5. Evaluation of De-identified Policing Dataset

The three names are input as desired in searches 1, 2 and 3 to find the corresponding match in the database. Figure 4.25 shows the start of the iteration process by providing three different name variations of search 1. The input name may or may not be the correct spelling of the name, but this is what is known or being guessed for searching. This search process produced records found in the database, as shown in Figure 4.26 and Figure 4.27.

➲ **Search 1** – The target name is Bech Jaunette

```
Please type the 1st variation of name : back junete
Please type the 2nd variation of name : bach junette
Please type the 3rd variation of name : beck jaunete
Please type the gender (m/f) : f
Please type approx. age : 49
Please type the description : white
Please type the address : town and close

Found records for 1st variation of name :  95
Found records for 2nd variation of name :  61
Found records for 3rd variation of name :  95
Length of Dataset by merging three name variation datasets  :  173
Length of Dataset without duplicate records :  173
ClusterID [0 1 2 3]
```

Figure 4.25 - Search 1: Number of found records per search stage

Final Records By Address :

												0
name	address	nominal_ref	age	date_of_birth	postcode	FullNameScore	address_score	Cluster	gender	ea_desc	role_type	
BECH JAUNETTE	TOWN END CLOSE YAULE TETHERDALE	118137890Q	59	1964-08-16	B66 3TY	0.7	0.83	1	F	WHITE SKINNED EUROPEAN	DEFE	1
		201060383H	59	1964-08-16	B66 3TY	0.7	0.83	1	F	WHITE SKINNED EUROPEAN	DEFE	5
											VICT	1

Related Records found by name :

												0
name	address	nominal_ref	age	date_of_birth	postcode	FullNameScore	address_score	Cluster	gender	ea_desc	role_type	
BECH JAUNETTE	BREEZE LANE YAULE TETHERDALE	96346618M	59	1964-08-16	B66 4PR	0.7	0.29	1	F	WHITE SKINNED EUROPEAN	DEFE	1
	BROWNEY CROFT YAULE TETHERDALE	96346618M	59	1964-08-16	B66 4PR	0.7	0.27	1	F	WHITE SKINNED EUROPEAN	DEFE	1
	FARTHINGDALE CLOSE CARSINGTON TEIGN HILL	201060383H	59	1964-08-16	B11 3AF	0.7	0.53	1	F	WHITE SKINNED EUROPEAN	VICT	1
	HESSLE TERRACE CARSINGTON THACKFORD	118137890Q	59	1964-08-16	B33 8RS	0.7	0.33	1	F	WHITE SKINNED EUROPEAN	DEFE	1
	KNAVESMIRE YAULE TETHERDALE	96346618M	59	1964-08-16	B66 4PR	0.7	0.24	1	F	WHITE SKINNED EUROPEAN	DEFE	1
	ROMAN AVENUE YAULE TETHERDALE	96346618M	59	1964-08-16	B66 4PR	0.7	0.28	1	F	WHITE SKINNED EUROPEAN	DEFE	3
	SERVIA GARDENS CARSINGTON DRAYFORD COMMON	85267154M	31	1992-05-26	B35 7LB	0.7	0.29	0	F	WHITE SKINNED EUROPEAN	VICT	1

Figure 4.26 - Search 1: Found records based on matching addresses and related records

Max value Add Score 0.83

												0
name	address	nominal_ref	age	date_of_birth	postcode	FullNameScore	address_score	Cluster	gender	ea_desc	role_type	
BECH JAUNETTE	BREEZE LANE YAULE TETHERDALE	96346618M	59	1964-08-16	B66 4PR	0.7	0.29	1	F	WHITE SKINNED EUROPEAN	DEFE	1
	BROWNEY CROFT YAULE TETHERDALE	96346618M	59	1964-08-16	B66 4PR	0.7	0.27	1	F	WHITE SKINNED EUROPEAN	DEFE	1
	FARTHINGDALE CLOSE CARSINGTON TEIGN HILL	201060383H	59	1964-08-16	B11 3AF	0.7	0.53	1	F	WHITE SKINNED EUROPEAN	VICT	1
	HESSLE TERRACE CARSINGTON THACKFORD	118137890Q	59	1964-08-16	B33 8RS	0.7	0.33	1	F	WHITE SKINNED EUROPEAN	DEFE	1
	KNAVESMIRE YAULE TETHERDALE	96346618M	59	1964-08-16	B66 4PR	0.7	0.24	1	F	WHITE SKINNED EUROPEAN	DEFE	1
	ROMAN AVENUE YAULE TETHERDALE	96346618M	59	1964-08-16	B66 4PR	0.7	0.28	1	F	WHITE SKINNED EUROPEAN	DEFE	3
	SERVIA GARDENS CARSINGTON DRAYFORD COMMON	85267154M	31	1992-05-26	B35 7LB	0.7	0.29	0	F	WHITE SKINNED EUROPEAN	VICT	1
	TOWN END CLOSE YAULE TETHERDALE	118137890Q	59	1964-08-16	B66 3TY	0.7	0.83	1	F	WHITE SKINNED EUROPEAN	DEFE	1
		201060383H	59	1964-08-16	B66 3TY	0.7	0.83	1	F	WHITE SKINNED EUROPEAN	DEFE	5
											VICT	1

Figure 4.27 - Search 1: Searching and filtering of records based on address score

Based on the inputs for Search 1 in Figure 4.25, the initial search and matching criteria produced three datasets. The first variation of the name retrieved 95 records, the second retrieved 61, and the third retrieved 95. A resultant dataset of 173 records is generated after merging all three datasets and removing duplicate records. The records are input into the clustering algorithm to calculate the number of clusters based on the records found with matching addresses, names, and ages. Figure 4.28. shows the clustering of the records.

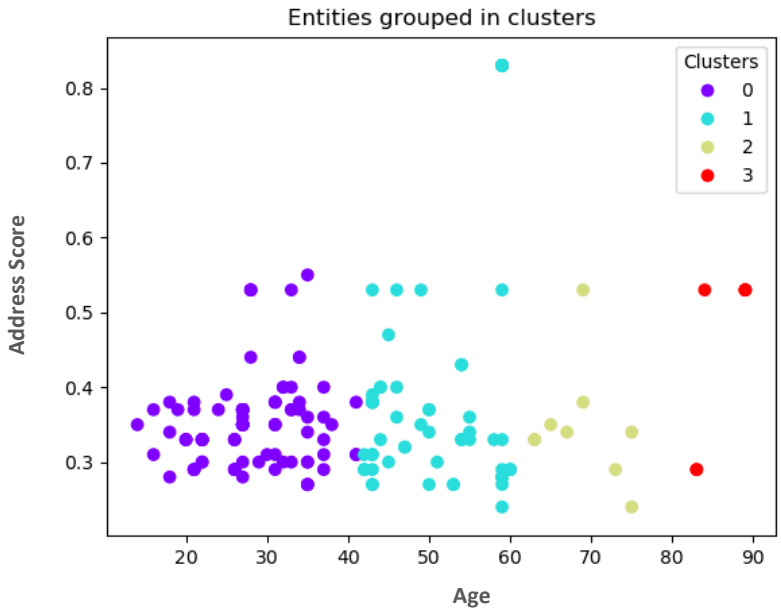

Figure 4.28 - Search 1: Clustered records

The dataset produced from the search results is fed into the clustering algorithm, and clusters of the records are created based on age, name aggregate score, and address score. The records are grouped into four clusters based on the similarity score of the attributes involved using the Mean-Shift clustering technique. Some of the records in clusters have low scores at the y-axis (address score). However, these records are still required for the next stage to ensure not to ignore any related or close matches. After the clustering phase, the segmentation picked the address with the highest score of 0.83. Figure 4.26 shows that 7 records, displayed in the last column, were retrieved matching the highest address and name aggregate scores. In comparison, 9 related records were retrieved with similar names with low address scores and a different date of birth. Therefore, after merging these retrieved records, the final dataset has been obtained containing 16 records, displayed in the last column, with a combination of the same name, two different date of birth, and different addresses, as shown in Figure 4.27.

Figure 4.29 below presents the graphical analysis of the clustered records, where blue dots represent individual entities by name, and the orange dots denote the clusters. There are 4 clusters in Figure 4.29, with different entities linked to each cluster. The two small clusters at the top right and bottom right in the graph are simple clusters of entities and have no similarities with other clusters. However, the two clusters in the middle left of the graph have some similar entities linked to each cluster. The three entities linked to each cluster are shown in Figure 4.29 below.

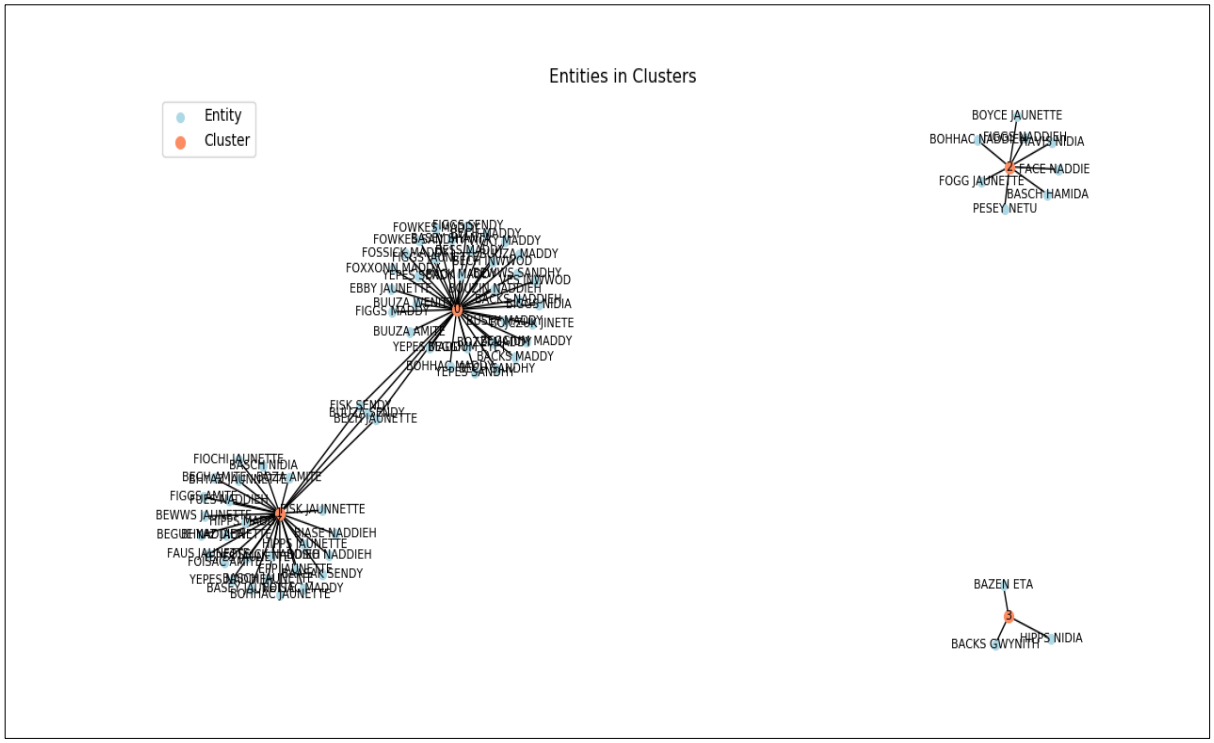

Figure 4.29 - Graph analysis, the suspect identified as "Bech Jaunette" clustered

Figure 4.30 further explores the entity linkage details in the graph analysis, where the red dot represents the matched entity. In contrast, the orange dot represents the clusters. So, one of the three entities linked to two clusters from Figure 4.29 is matched with the target search entity and represented with a red dot in Figure 4.30. This entity is linked to different addresses, as shown in Figure 4.30. Therefore, the suspect was identified out of other entities, with a red dot associated with different addresses.

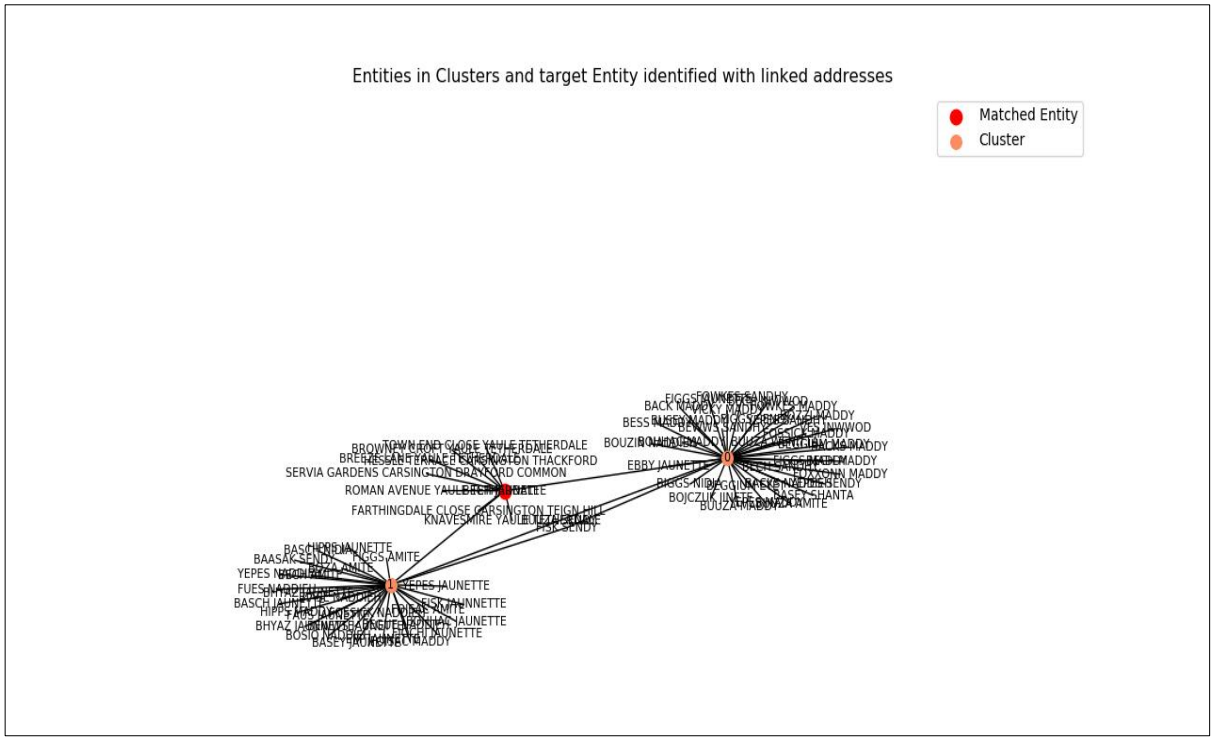

Figure 4.30 - Graph analysis, the suspect identified as “Bech Jaunette” highlighted Red in Clusters associated with different addresses.

However, this matched suspect entity in Figure 4.30 requires further clarity on the address to be easily readable, as shown in Figure 4.31. Here, the red dot represents the suspect entity, and the black arrows show the particular entity's link with different addresses. The grey dot represents each address in Figure 4.31.

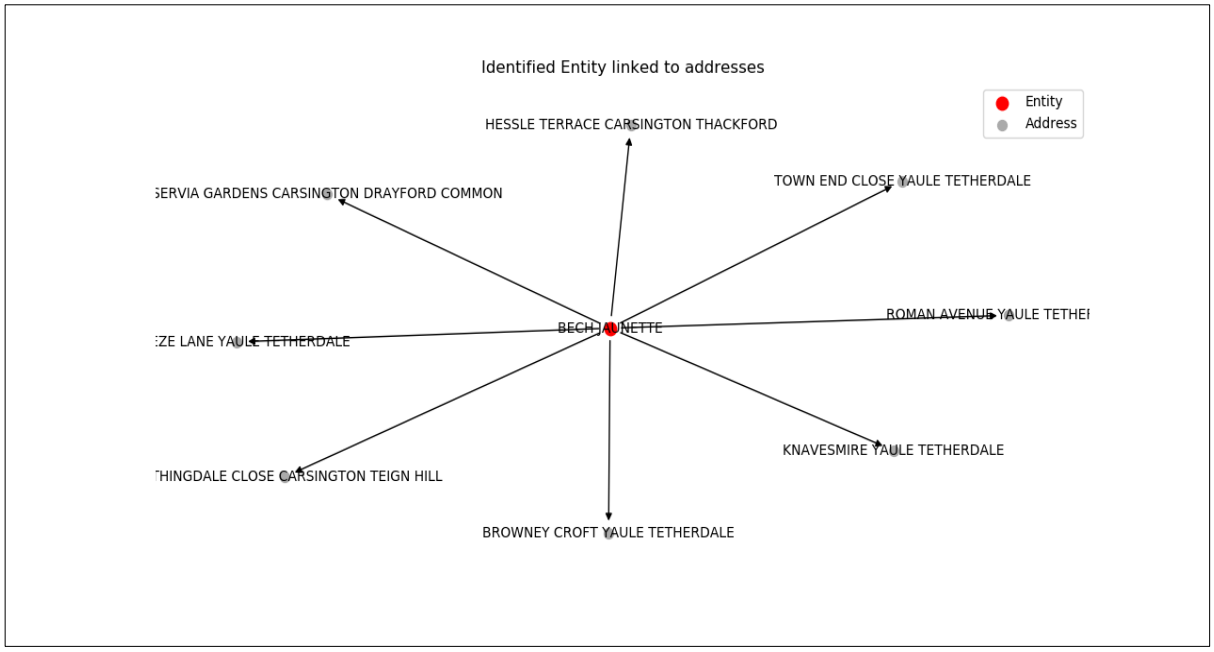

Figure 4.31 - Graph analysis, the suspect identified as "Bech Jaunette" highlighted Red and associated with different addresses.

➲ **Search 2** – The target name is Abbidah Faroz

```
Please type the 1st variation of name : abidah feros
Please type the 2nd variation of name : abbidha firoz
Please type the 3rd variation of name : abiddah farose
Please type the gender (m/f) : f
Please type approx. age : 44
Please type the description : white
Please type the address : close carsington

Found records for 1st variation of name :  41
Found records for 2nd variation of name :  41
Found records for 3rd variation of name :  41
Length of Dataset by merging three name variation datasets  :  85
Length of Dataset without duplicate records :  85
ClusterID [0 1 2 3]
```

Figure 4.32 - Search 2: Number of found records per search stage

Based on the inputs for Search 2 in Figure 4.32, the initial search and matching criteria produced three datasets. All three name variations retrieved 41 records, and the resultant dataset of 85 records was generated after merging all three datasets and removing any duplicate records. All these retrieved records have some similarities or are entirely different, but this will be distinguished later in the next stage.

Final Records By Address :

												0
name	address	nominal_ref	age	date_of_birth	postcode	FullNameScore	address_score	Cluster	gender	ea_desc	role_type	
ABBIDAH FAROS	ELGOOD CLOSE CARSINGTON ARBORFIELD	114726519Y	46	1977-02-25	B32 2QN	0.50	1.0	0	F	WHITE SKINNED EUROPEAN	VICT	1
						0.69	1.0	0	F	WHITE SKINNED EUROPEAN	VICT	1
BOTY FAROS	BERESFORD CLOSE CARSINGTON LOWER DRAYFORD	871186422T	53	1970-02-28	B36 0TY	0.50	1.0	1	F	WHITE SKINNED EUROPEAN	VICT	1
						0.70	1.0	1	F	WHITE SKINNED EUROPEAN	VICT	1
	HOLBETON CLOSE CARSINGTON FORTHINGTON	97898721E	42	1981-02-25	B23 6JN	0.50	1.0	0	F	WHITE SKINNED EUROPEAN	VICT	1
						0.70	1.0	0	F	WHITE SKINNED EUROPEAN	VICT	1

Related Records found by name :

												0
name	address	nominal_ref	age	date_of_birth	postcode	FullNameScore	address_score	Cluster	gender	ea_desc	role_type	
ABBIDAH FAROS	BRANDEARTH HEY CARSINGTON ARBORFIELD	1012878080T	46	1977-02-02	B32 2QN	0.50	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	2
						0.69	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	2
		114726519Y	46	1977-02-25	B32 2QN	0.50	0.77	0	F	WHITE SKINNED EUROPEAN	DEFE	2
											VICT	3
						0.69	0.77	0	F	WHITE SKINNED EUROPEAN	DEFE	2
											VICT	3
		155151271G	46	1977-02-17	B32 2QN	0.50	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	1
						0.69	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	1
	CHILD'S ERCALL LANE CARSINGTON BARBWORTH	10131383640	47	1976-02-18	B25 8UT	0.50	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	1
						0.69	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	1
	PRIMROSE AVENUE DEWMAPLE BLACKFORD	86197283B	53	1970-03-18	CV5 9JZ	0.50	0.32	1	F	WHITE SKINNED EUROPEAN	VICT	1
						0.69	0.32	1	F	WHITE SKINNED EUROPEAN	VICT	1
BOTY FAROS	ATHENS GARDENS CARSINGTON GARNET GREEN	13369907L	23	2000-11-15	B28 8LZ	0.50	0.77	3	F	WHITE SKINNED EUROPEAN	DEFE	1
						0.70	0.77	3	F	WHITE SKINNED EUROPEAN	DEFE	1

Figure 4.33 - Search 2: Found records based on matching addresses and related records

Max value Add Score 1.0

												0
name	address	nominal_ref	age	date_of_birth	postcode	FullNameScore	address_score	Cluster	gender	ea_desc	role_type	
ABBIDAH FAROS	BRANDEARTH HEY CARSINGTON ARBORFIELD	1012878080T	46	1977-02-02	B32 2QN	0.50	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	2
						0.69	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	2
		114726519Y	46	1977-02-25	B32 2QN	0.50	0.77	0	F	WHITE SKINNED EUROPEAN	DEFE	2
											VICT	3
						0.69	0.77	0	F	WHITE SKINNED EUROPEAN	DEFE	2
											VICT	3
		155151271G	46	1977-02-17	B32 2QN	0.50	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	1
						0.69	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	1
	CHILD'S ERCALL LANE CARSINGTON BARBWORTH	10131383640	47	1976-02-18	B25 8UT	0.50	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	1
						0.69	0.77	0	F	WHITE SKINNED EUROPEAN	VICT	1
	ELGOOD CLOSE CARSINGTON ARBORFIELD	114726519Y	46	1977-02-25	B32 2QN	0.50	1.00	0	F	WHITE SKINNED EUROPEAN	VICT	1
						0.69	1.00	0	F	WHITE SKINNED EUROPEAN	VICT	1
	PRIMROSE AVENUE DEWMAPLE BLACKFORD	86197283B	53	1970-03-18	CV5 9JZ	0.50	0.32	1	F	WHITE SKINNED EUROPEAN	VICT	1
						0.69	0.32	1	F	WHITE SKINNED EUROPEAN	VICT	1
BOTY FAROS	ATHENS GARDENS CARSINGTON GARNET GREEN	13369907L	23	2000-11-15	B28 8LZ	0.50	0.77	3	F	WHITE SKINNED EUROPEAN	DEFE	1
						0.70	0.77	3	F	WHITE SKINNED EUROPEAN	DEFE	1
	BERESFORD CLOSE CARSINGTON LOWER DRAYFORD	871186422T	53	1970-02-28	B36 0TY	0.50	1.00	1	F	WHITE SKINNED EUROPEAN	VICT	1
						0.70	1.00	1	F	WHITE SKINNED EUROPEAN	VICT	1
	HOLBETON CLOSE CARSINGTON FORTHINGTON	97898721E	42	1981-02-25	B23 6JN	0.50	1.00	0	F	WHITE SKINNED EUROPEAN	VICT	1
						0.70	1.00	0	F	WHITE SKINNED EUROPEAN	VICT	1

Figure 4.34 - Search 2: Searching and filtering of records based on address score

Applying the resultant dataset to the clustering phase generated 4 clusters based on age, name aggregate score, and address score. Figure 4.35 shows that some cluster records have low scores at the y-axis (address score). These records are required for the next stage to ensure that any related or close matches are not ignored. After the clustering phase, the segmentation picked the address with the highest score of 1.0. In Figure 4.33, 6 records were retrieved with the highest address and different name aggregate scores.

In comparison, 20 other related records were retrieved with a similar name, low address score, and 5 different dates of birth. So, the final dataset retrieved 22 records with a combination of the same name, 6 different dates of birth, and different addresses. In comparison, it contains 6 records with a different name, 3 different dates of birth, and a high address matching score, possibly having similar addresses as shown in Figure 4.34.

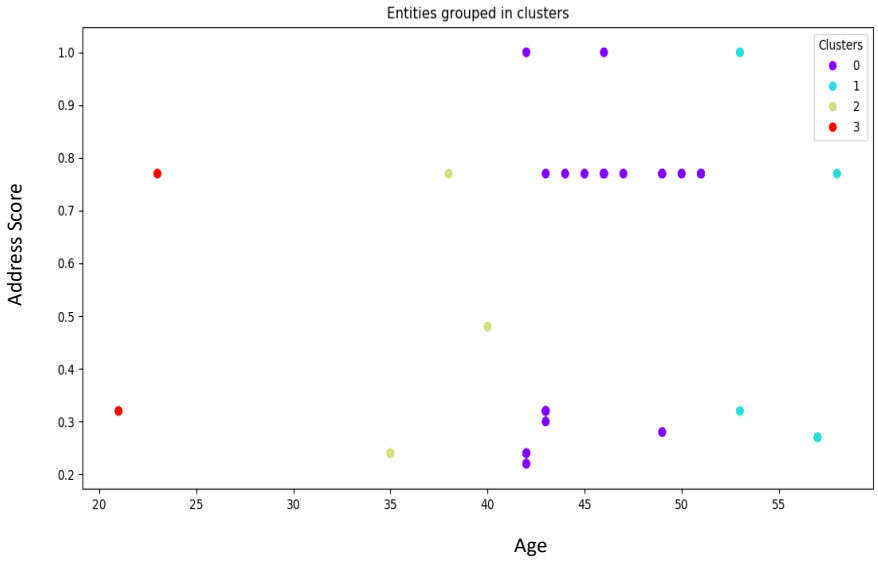

Figure 4.35 - Search 2: Clustered records

Figure 4.36 shows a graph analysis of the entities added to different clusters. The blue dot represents the entity, while the orange represents each cluster, showing that different entities are linked to each cluster. However, some entities are linked to other clusters too. These are familiar entities found in multiple clusters due to some

similarities. In the graph of Figure 4.36, entities are linked between clusters 0, 1, 2 and 3.

Meanwhile, clusters 1 and 2 have a common entity but are not linked to other clusters. Two similar entities are linked to clusters 1 and 3. Therefore, the main clusters to focus on here are “0 and 1”, “0 and 2”, and “0 and 3”. Further graph analysis is required to find the matched entity from these entities.

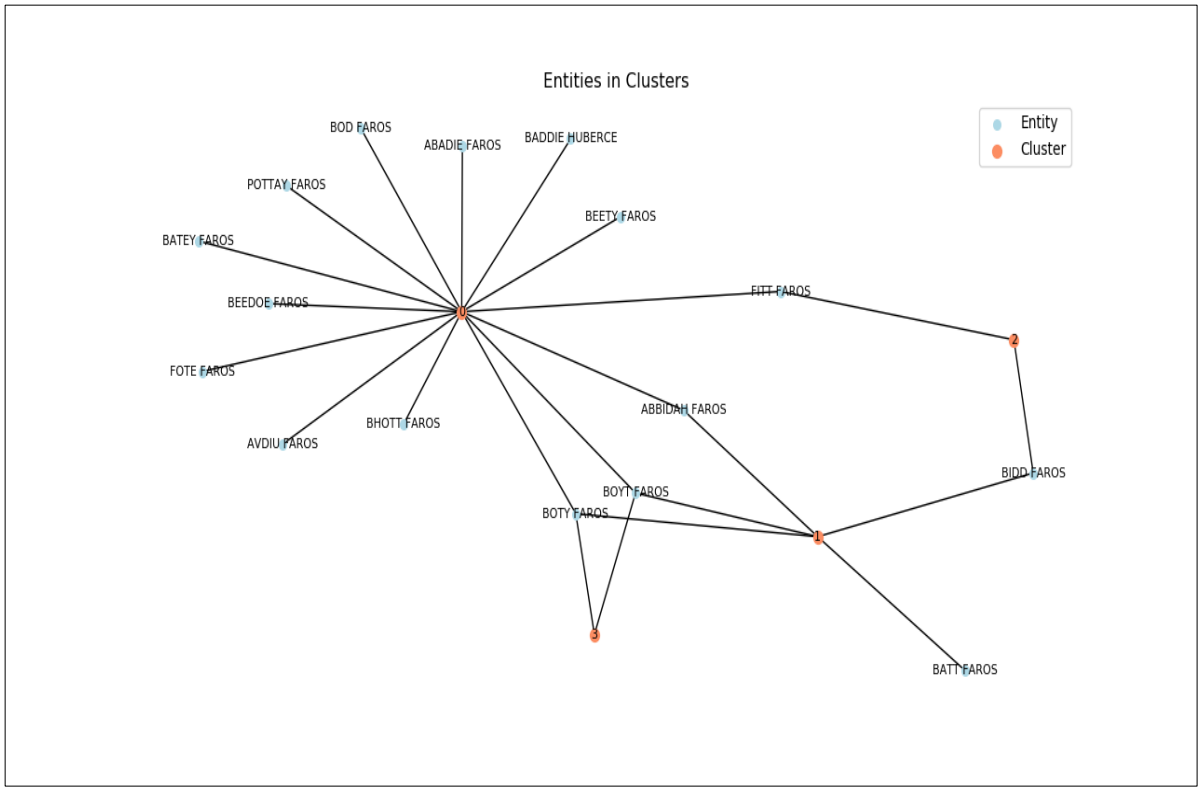

Figure 4.36 - Graph analysis, the suspect identified as “Abbidah Faroz” clustered

Figure 4.37 shows further analysis of the identified clusters. The entities in clusters are linked to different addresses. In Figure 4.37, orange dots represent the clusters, while the red dots represent the match entities. The match entities are also linked to addresses. At the same time, the entities are shown as linked to other clusters, as discussed before. However, this graph is not easily interpretable with respect to the addresses where the entities are linked.

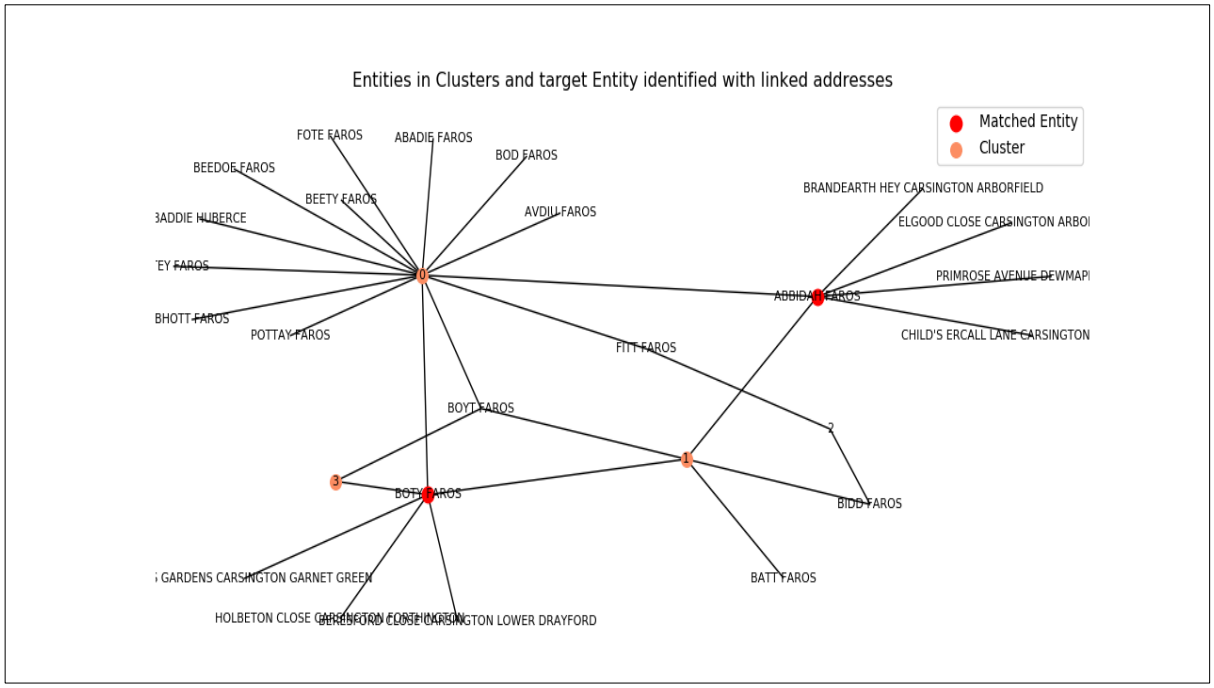

Figure 4.37 - Graph analysis, the suspect identified as "Abbidah Faroz" highlighted Red and Red in Clusters associated with different addresses.

Therefore, to clarify these details, Figure 4.38 shows the matched entities with different addresses.

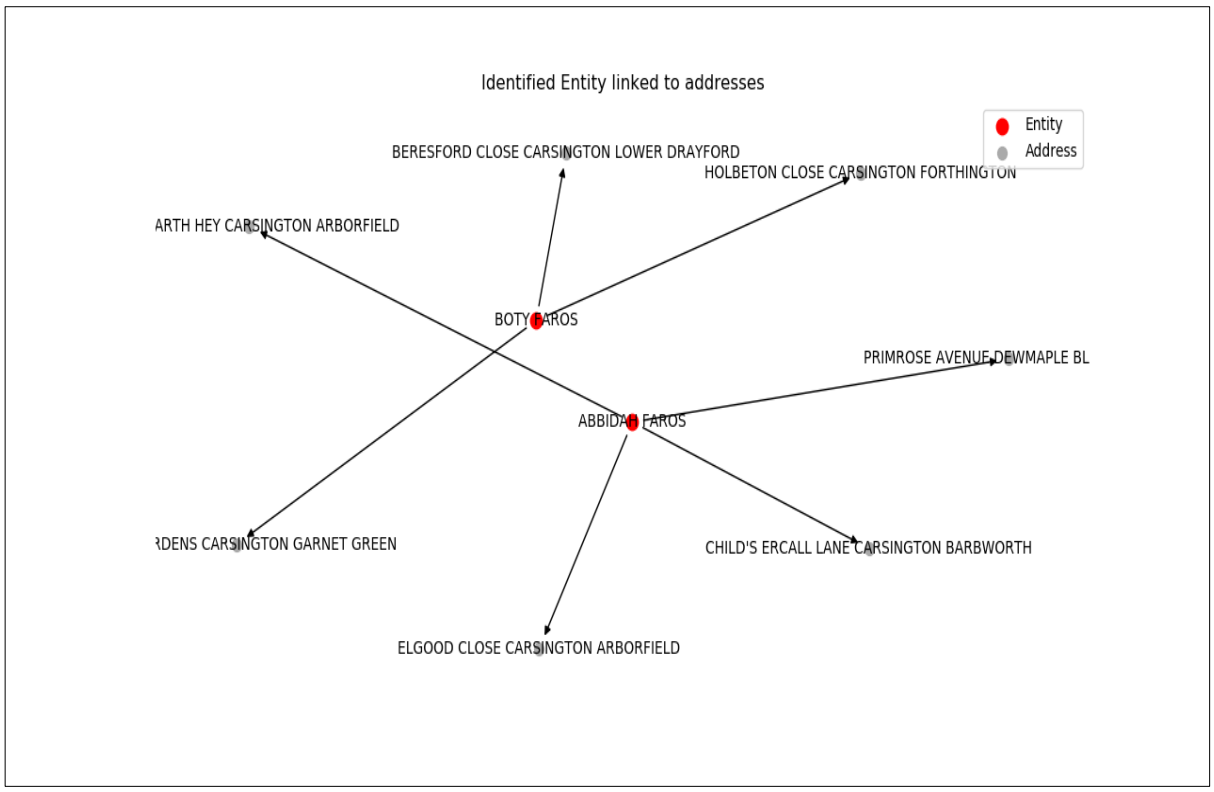

Figure 4.38 - Search 2: Graph analysis, the suspect identified as "Abbidah Faroz" highlighted Red and associated with different addresses.

In Figure 4.38, the red dots represent the matched entities. These entities are linked to addresses represented by black arrows, where grey dots represent the address. The results of the graph analysis show there are two matched entities. One suspect entity can be easily identified in terms of the search for the target entity. Therefore, “Abbidah Faros” is the matched suspect entity. However, the other suspect entity, “Boty Faros”, is either a linked related suspect or a false negative matched entity.

- **Search 3** – The target name is Haskin Zeid

```
Please type the 1st variation of name : huskin zaid
Please type the 2nd variation of name : haskin zayed
Please type the 3rd variation of name : hasken zeid
Please type the gender (m/f) : m
Please type approx. age : 30
Please type the description : white
Please type the address : carsington drampton

Found records for 1st variation of name :  204
Found records for 2nd variation of name :  204
Found records for 3rd variation of name :  204
Length of Dataset by merging three name variation datasets  :  208
Length of Dataset without duplicate records :  208
ClusterID [0 1 2 3]
```

Figure 4.39 - Search 3: Number of found records per search stage

Based on the inputs for search 3 in Figure 4.39, the initial search and matching criteria produced three datasets. All three name variations retrieved 204 records, and the resultant dataset of 208 records was generated after merging all three datasets and removing duplicate records. All these retrieved records have some similarities or are entirely different, but this will be differentiated later in the next stage.

Final Records By Address :

												0
name	address	nominal_ref	age	date_of_birth	postcode	FullNameScore	address_score	Cluster	gender	ea_desc	role_type	
HASKIN ZIED	BASING STREET CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	BITTACY RISE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	CLEVEDON ROAD CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	COCK BANK CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	ELECTRIC PARADE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	FORD LANE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	KELMSCOTT CRESCENT CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	KNOWLE GROVE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	MOULE RISE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	8
	SCOTIA DRIVE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 7UA	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	SORREL GARDENS CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	WENTWORTH GROVE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	WOODNEWTONS ROAD CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	WOODTHORPE CRESCENT CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
KEYNES UTT	ROCKHALL ROAD CARSINGTON DRAMPTON	921272734C	30	1993-05-25	B37 7UR	0.7	1.0	0	M	WHITE SKINNED EUROPEAN	DEFE	1
											VICT	1

Related Records found by name :

												0
name	address	nominal_ref	age	date_of_birth	postcode	FullNameScore	address_score	Cluster	gender	ea_desc	role_type	
HASKIN ZIED	BURNT ASH HILL CARSINGTON TARLINGTON	55100010B	24	1999-11-19	B26 1LY	0.7	0.69	0	M	WHITE SKINNED EUROPEAN	DEFE	2
	QUORN ROAD CARSINGTON IVERSONS HEATH	381368636J	24	1999-11-19	B19 3UL	0.7	0.69	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	REMNANT STREET CARSINGTON UPPEREND GREEN	55100010B	24	1999-11-19	B33 0AN	0.7	0.69	0	M	WHITE SKINNED EUROPEAN	DEFE	1

Figure 4.40 – Search 3: Found records based on matching addresses and related records

Max value Add Score 1.0

												0
name	address	nominal_ref	age	date_of_birth	postcode	FullNameScore	address_score	Cluster	gender	ea_desc	role_type	
HASKIN ZIED	BASING STREET CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	BITTACY RISE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	BURNT ASH HILL CARSINGTON TARLINGTON	55100010B	24	1999-11-19	B26 1LY	0.7	0.69	0	M	WHITE SKINNED EUROPEAN	DEFE	2
	CLEVEDON ROAD CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	COCK BANK CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	ELECTRIC PARADE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	FORD LANE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	KELMSCOTT CRESCENT CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	KNOWLE GROVE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	MOULE RISE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	8
	QUORN ROAD CARSINGTON IVERSONS HEATH	381368636J	24	1999-11-19	B19 3UL	0.7	0.69	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	REMNANT STREET CARSINGTON UPPEREND GREEN	55100010B	24	1999-11-19	B33 0AN	0.7	0.69	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	SCOTIA DRIVE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 7UA	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	SORREL GARDENS CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	WENTWORTH GROVE CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	WOODNEWTONS ROAD CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
	WOODTHORPE CRESCENT CARSINGTON DRAMPTON	55100010B	24	1999-11-19	B37 5EW	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
KEYNES UTT	ROCKHALL ROAD CARSINGTON DRAMPTON	921272734C	30	1993-05-25	B37 7UR	0.7	1.00	0	M	WHITE SKINNED EUROPEAN	DEFE	1
											VICT	1

Figure 4.41 - Search 3: Searching and filtering of records based on address score

The dataset, produced from the search results, is fed into the clustering algorithm, and 4 clusters of records are created by the Mean-Shift clustering algorithm based on age, name aggregate score, and address score, as shown in Figure 4.42. There are records in clusters with low scores at the y-axis (address score). However, these records are

required for the next stage to ensure that any related or close matches are not ignored. After the clustering phase, the segmentation picked the address with the highest score of 1.0. Figure 4.40 shows a total of 21 records, numbers displayed in the last column, were retrieved with the highest address and name aggregate scores of 0.70 and the same date of birth. However, the 2 other records, numbers displayed in the last column, were retrieved with different names, high address scores, and dates of birth. At the same time, 4 records were found with related records with matching names, a low address score of 0.69 and the exact date of birth, the same as 21 records found during the initial search. Therefore, the final dataset containing 27 records, numbers displayed in the last column, and the combination of the two different names, dates of birth, and addresses was obtained, as shown in Figure 4.41.

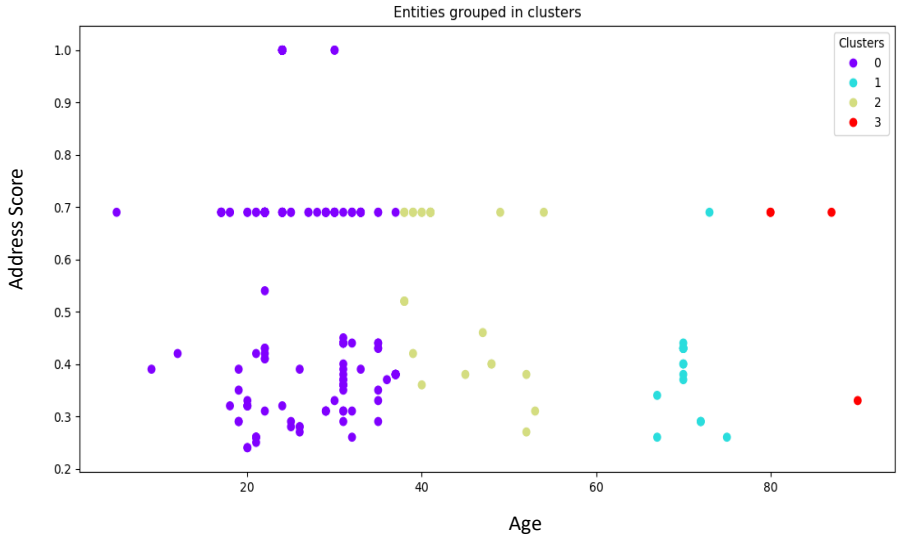

Figure 4.42 - Search 3: Clustered records

Figure 4.43 below is the graph analysis of the clustered records. The blue dots represent the Entity and the individual's name, and the orange dots represent the clusters. The 4 clusters contain different entities linked to each cluster. The small cluster at the top left of the graph is a simple cluster with no similarities. However, the two clusters on the bottom left of the graph have some similar entities linked to each cluster. One entity is linked to clusters 1 and 2, while the other is linked to clusters 0 and 1.

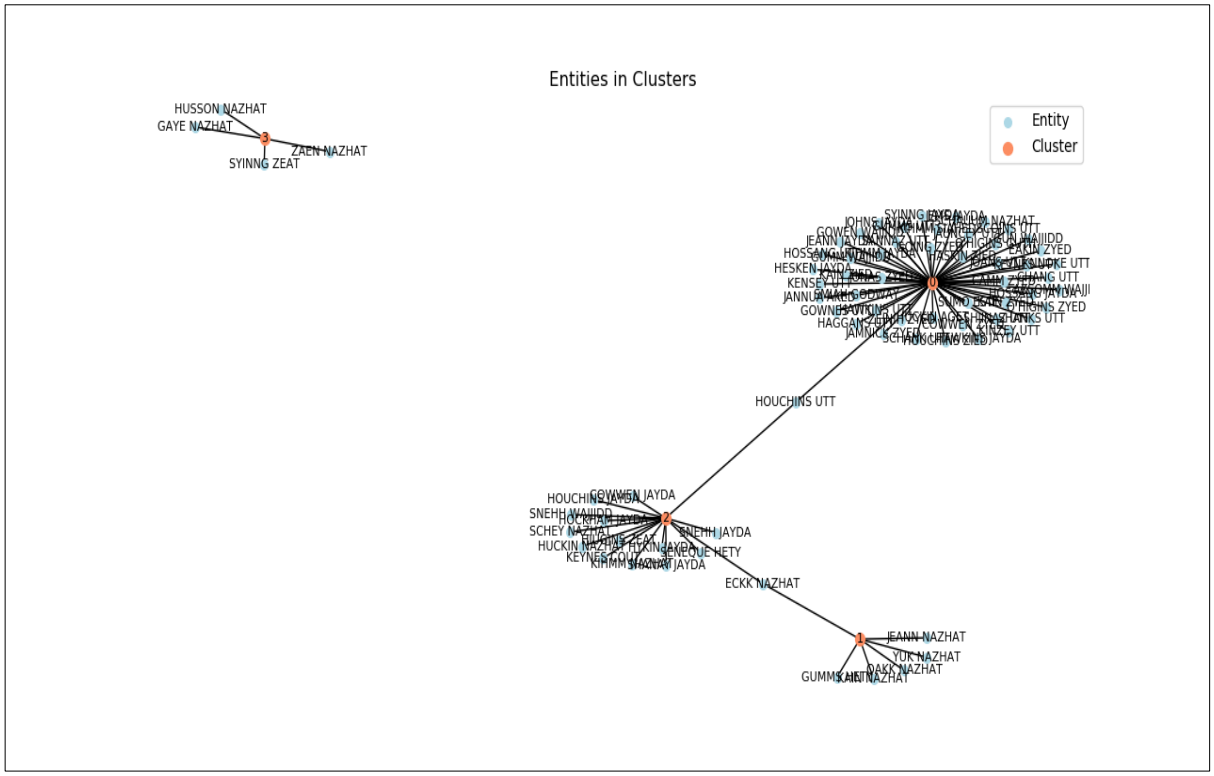

Figure 4.43 - Graph analysis, the suspect identified as "Haskin Zeid" clustered

Figure 4.44 further explores the entity linkage details in the graph analysis, where the red dot represents the matched entity. In contrast, the orange dot represents the clusters. Here, two entities linked to two clusters are matched with the target search entity and represented with a red dot in Figure 4.44. However, one entity is an entirely different name while getting a high address match score. It could be a close match or a related match retrieved during the search process. Therefore, the suspect has been identified out of other entities, represented with a red dot associated with different addresses matching the exact search name.

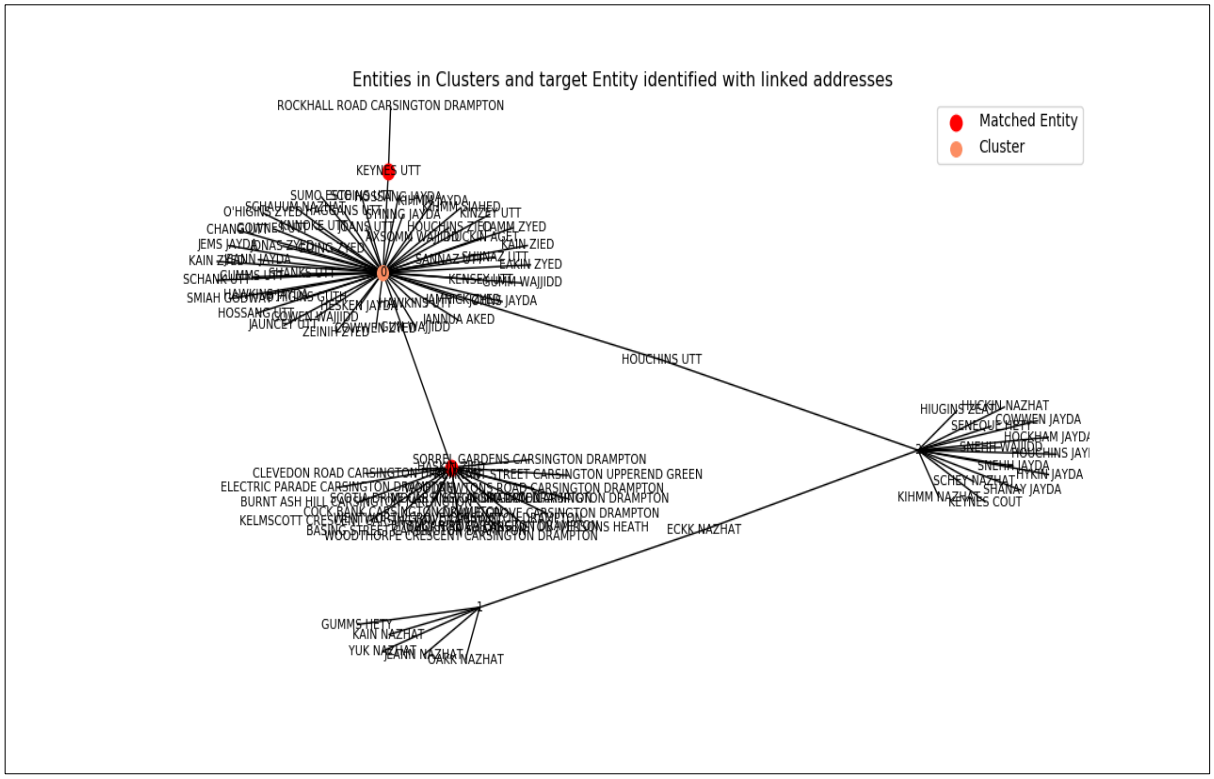

Figure 4.44 - Graph analysis, the suspect identified as “Haskin Zeid” highlighted Red and other Red in Clusters associated with different addresses.

However, this matched suspect entity in Figure 4.44 requires further clarity on the address to be easily readable, as shown in Figure 4.45. The red dot represents the suspect entity, and the black arrows show the particular entity's link with different addresses. In contrast, the grey dot represents each address. Therefore, “Haskin Zied” is the matched suspect entity. The other suspect entity, “Keynes Utt”, is either a linked suspect or a false negative matched entity.

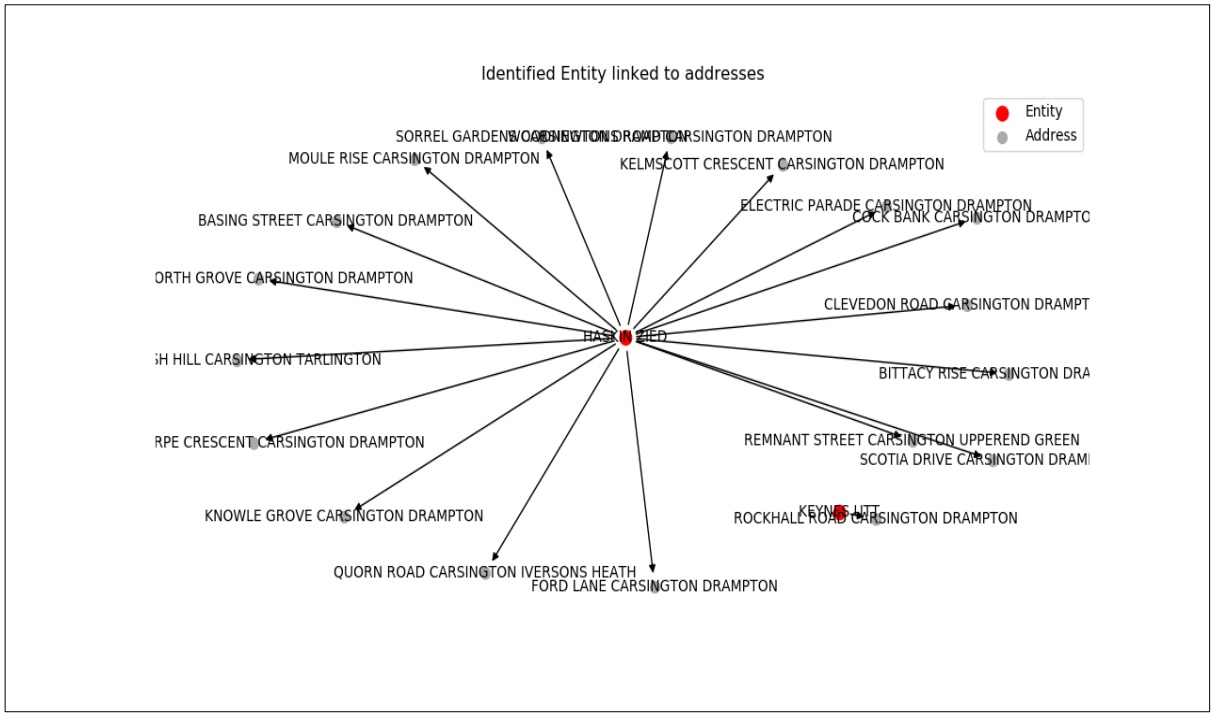

Figure 4.45 - Search 3: Graph analysis, the suspect identified as "Haskin Zied" highlighted Red and associated with different addresses.

4.6. Policing Dataset Results Performance Analysis

After the search results from the database on different scenarios above, it is apparent that both similarity techniques mentioned are performing up to some extent to find a match for the given name but have issues matching the string.

To further discuss, we can see the matching scores in Table 4.2 to compare the matching results. In Table 4.2. the target string is converted into the 6-digit Soundex code to compare and get the scores and the aggregate score.

Table 4.2 - Results Performance Comparison of Fuzzy Matching Scores

Search	Target String	Target String 6-digit Soundex Code	Input String	6-digit Soundex Code	Soundex-JW Score	Jaro-Winkler Score	Aggregate Score
1	Bech Jaunette	122530	Back Junete	125300	0.91	0.84	**0.88**
			Bach Junette	122530	1.00	0.91	**0.96**
			Beck Jaunete	125300	0.91	0.94	**0.93**
2	Abbidah Faros	131620	Abidah Feros	131620	1.00	0.94	**0.97**
			Abbidha Firoz	131620	1.00	0.92	**0.96**
			Abiddah Farose	131620	1.00	0.94	**0.97**
3	Haskin Zeid	252300	Huskin Zaid	252300	1.00	0.89	**0.95**
			Haskin Zayed	252300	1.00	0.95	**0.97**
			Hasken Zeid	252300	1.00	0.92	**0.96**

The graph below shows the aggregate score against modified Soundex and Jaro-Winkler scores.

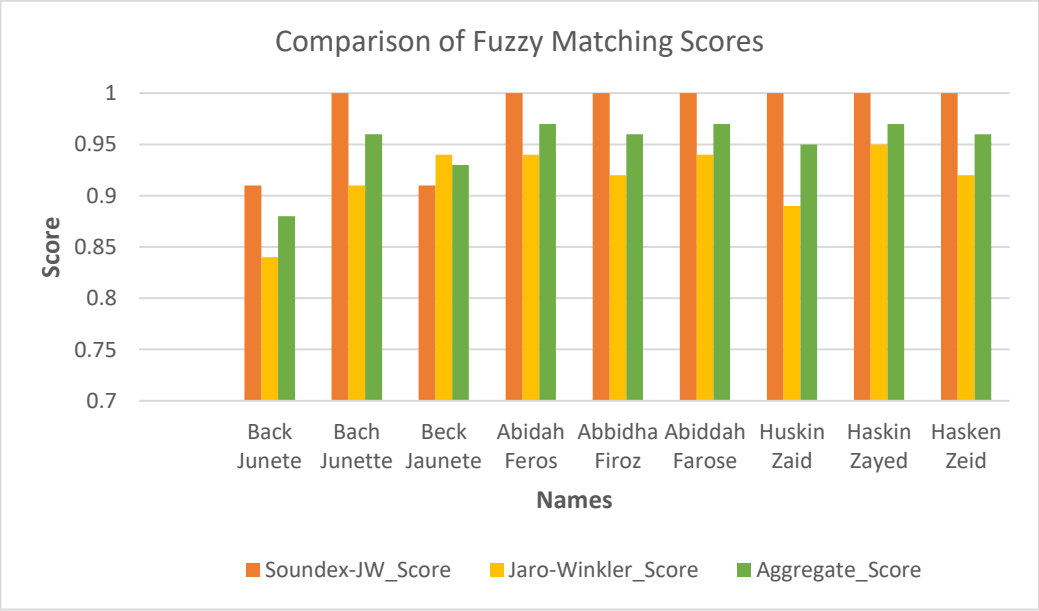

Figure 4.46 - Results comparison of Names Fuzzy Matching Scores

From Figure 4.46, it can be seen that the selected similarity metrics are suitable for matching names. However, an individual technique alone is unsuitable for generating accurate required results to match the searched string. Therefore, combining Soundex and Jaro_Winkler techniques is better for generating the aggregate score for name matching. This aggregate score method fits the purpose of doing fuzzy matching of strings. For long strings, edit distance is suitable for the approximate matching of strings. It is about scoring the strings to help cluster similar records that have been retrieved. The input strings Soundex-JW scores are the same or similar results scores matching names from the database.

In contrast, the Jaro-Winkler matching scores are different for input strings. However, the aggregate score provides the fuzzy score. It shows better results, showing that the names are not exact matches but are only fuzzy. The results show that the matching performance of the Jaro_Winkler is good. However, the aggregate score gives an even better fuzzy score to match strings in fuzzy matching.

The records are grouped using the clustering technique based on the aggregate score of strings. However, evaluating if the clusters are good or bad created is necessary. It can be achieved by applying the Silhouette Coefficient or score for clusters.

4.6.1. Silhouette Coefficient

The Silhouette Coefficient or Silhouette score is a metric used to calculate the goodness of a clustering technique (Rousseeuw, 1987). The value ranges from -1 to 1.

Where:

- 1: This means clusters are well apart from each other and distinguished.
- 0: This means clusters are indifferent, or we can say that the distance between clusters is not significant.
- -1: This means clusters are assigned in the wrong way.

The formula to calculate the Silhouette score is:

Silhouette Score = (b – a) / max (a, b)

Here:

- a = An average intra-cluster distance, i.e., the average distance between each point within a cluster.
- b = An average inter-cluster distance, i.e., the average distance between all clusters.

The clustering results generated by the three search results in evaluating the de-identified dataset can be measured using the Silhouette Coefficient method. Below are the clustering figures from the three search results. The Silhouette Coefficient score is calculated for each clustering search result.

Search 1: Figure 4.47 shows the clustering of entities where 4 clusters are generated by putting entities into clusters. For easy understanding, these clusters are represented using different colours in Figure 4.47. The Silhouette Coefficient score is applied below to find out the quality or performance of the clustering of entities.

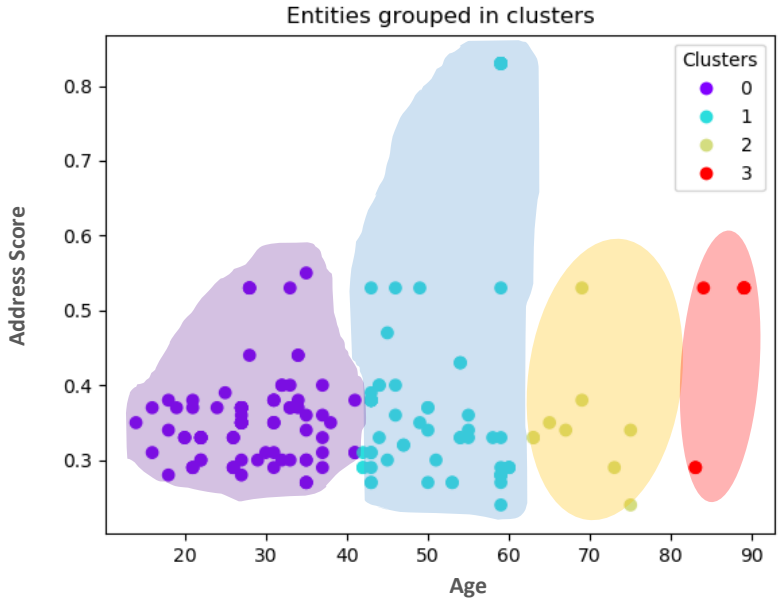

Figure 4.47 - Search 1: Highlighted Clustered records

Silhouette Score(n =): 4," " ,0.556

It calculates that there are 4 clusters, and the score is 0.556. The round-off score is 0.56 for this clustering of records, which has a positive value close to 1. Therefore, it shows that the clusters are well apart and that the clustering quality of the records is good.

Search 2: Figure 4.48 shows the clustering of entities generated in search 2 during the results evaluation of the dataset. There are 4 clusters generated, and each cluster is represented with a different colour for easy understanding in Figure 4.48. To measure the cluster quality, the Silhouette Score is applied to calculate the clustering score.

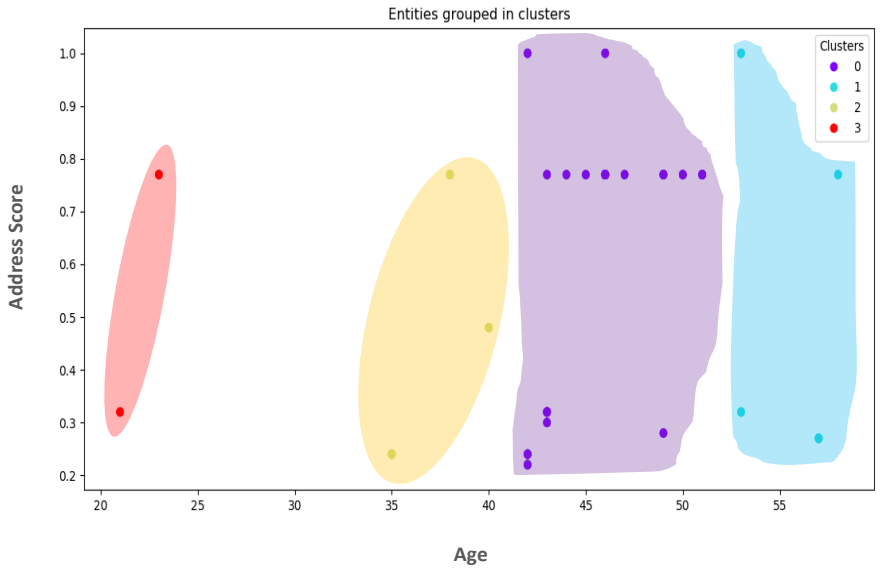

Figure 4.48 - Search 2: Highlighted Clustered records

Silhouette Score(n =): 4," " ,0.647

The Silhouette Score calculates 4 clusters with a score of 0.647. After round-off, the score is 0.65 for this clustering of the records. The score is close to 1, and again, it shows that the clusters are dense and well apart for search 2. Therefore, it means the clusters are of good quality.

Search 3: Figure 4.49 shows clustered entities generated during search 3. The entities are grouped into 4 clusters and are represented with different colours

for easy understanding, as shown in Figure 4.49. The Silhouette coefficient is applied to measure the performance of the clusters.

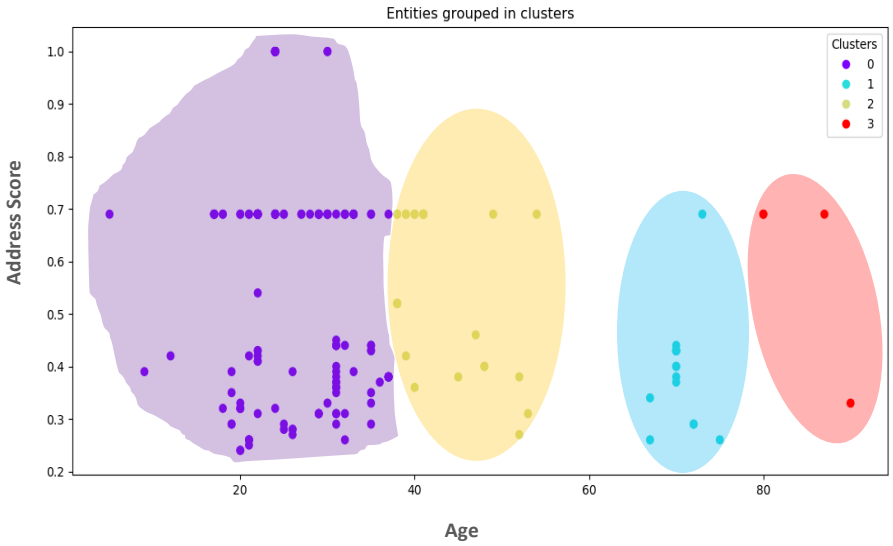

Figure 4.49 - Search 3: Highlighted Clustered records

Silhouette Score(n =): 4," " ,0.571

The Silhouette Score is calculated as 0.57 with a total of 4 clusters. The score is close to 1, so the clusters are well apart and dense. Therefore, the clustering quality of search 3 records is good.

4.7. Summary

The improved Soundex algorithm was applied to the names collected from three languages: English, Arabic, and Russian. It was also applied to mixed language names of different name variations. The computer-simulated results show that each name-matching score matches the names based on the aggregate score. The results of name scores are compared and analysed based on Soundex, Jaro-Winkler and aggregate scores. These matching results provided good fuzzy matching, where using only a single matching technique will not match names correctly. The proposed model was then applied to the de-identified policing data. The results were generated for three different target individual's names. Each search was named search1, search2, and search3, and

each search identified the target identity. These identity-matching results were presented in tabular and visualised in graph format for easy identification. The results showed that the model efficiently identified the target individual. The Silhouette Coefficient was applied to the clustering of records for each of the three searches for clustering performance measures. The overall identification and clustering performance are promising to match any identity.

CHAPTER 5

5. CONCLUSION

This chapter concludes by summarising the findings related to the critical research questions and discussing the value and contribution to knowledge. Research limitations will be highlighted, and avenues for future work will be explored.

This research successfully addressed fundamental questions: How can the desired identity be extracted from a raw dataset? How can records be matched to derive meaningful information? Furthermore, how can relationships between different identities be intelligently established using pattern recognition? The research evaluated various string-matching techniques and determined that a combination of Soundex, Jaro-Winkler, and edit distance matching techniques was most effective.

Notably, modifications were made to the original Soundex technique to generate a six-digit numeric code, which does not retain the first character of the name but instead produces a purely numerical code. This code, combined with the Jaro-Winkler score, generates an aggregate score. This approach enables the extraction of matching names based on the aggregate score, facilitating the retrieval of related records from the dataset. It aids in clustering these retrieved records and performing graph analysis to identify potential target identities and associated links.

The research aims to identify and evaluate techniques for improved identity resolution. Consequently, using an unsupervised machine learning approach, different string similarity techniques were analysed to retrieve matching entity records and identify related links. These techniques were cascaded within the framework to produce an aggregate name score. Based on this score, names could be categorised as “match”, "possible match", and "close or related match” in cases where string matching was applied. However, the framework utilised an iterative search to combine three spelling variations of given names, enabling the retrieval of data related to each fuzzy-matched

name from the dataset. After obtaining the three datasets, they were merged into a final dataset, from which duplicate records were removed.

Similar matching name records from the final dataset were grouped to refine the results further. The Means-Shift clustering technique was employed to cluster matching records based on the name "aggregate score" and "age" attributes. Notably, the Mean-Shift algorithm used within this framework automatically determines the number of clusters, providing a more dynamic approach by allowing the number of clusters to be adjusted based on the dataset size. Once clustering was complete, the records were labelled with a cluster number. Using NetworkX, graph analysis was conducted, linking all entities based on selected attributes in a layered approach across multiple graphs. This method effectively identified the suspect entity and displayed links to different addresses associated with the same entity. The results indicate that this fuzzy matching approach is effective in retrieving suspect entities and related records, aiding in identity matching. These links are presented graphically, with detailed records stored in a table format for ease of review. The entire matching process is automated, requiring minimal human interaction and providing fuzzy attribute inputs to the framework. It enables results to be generated even with limited information about a suspect entity.

According to the comprehensive literature review:

- No existing unsupervised machine learning framework automatically adjusts and fine-tunes results based on input, utilising various similarity metrics in a cascaded manner for record retrieval.
- Existing approaches do not enable record matching and retrieval without training data samples or record de-duplication.
- Current methods do not employ different similarity metrics at various stages to produce optimal record linkage and relationship analysis results.
- The literature does not address the use of clustering techniques to group records without a fixed number of clusters, thus facilitating record segmentation for identity resolution.

Therefore, this framework for identity resolution incorporates intelligence in matching raw information by employing a suitable algorithm with pattern recognition capabilities

that mimic the human brain's ability for fuzzy matching. In June 2021, this research was presented and published in a Springer conference paper and is also available in Appendix A. The research has practical implications, particularly for law enforcement agencies, as this framework can expedite investigations with minimal available information about a suspect.

➲ Contributions to Knowledge Summary:

This research makes several significant contributions to identity resolution and data science. Firstly, the hybrid approach combining unsupervised machine learning techniques with traditional string-matching algorithms represents a novel advancement in the field. Developing and implementing a modified Soundex algorithm to produce a 6-digit numeric code, coupled with Jaro-Winkler and edit distance and aggregate score methods, introduced a more refined and efficient approach to record matching.

Moreover, the research provides valuable insights into applying clustering techniques in handling large datasets within a hybrid framework. The use of the Mean-Shift algorithm, which dynamically adjusts the number of clusters based on the dataset size, showcases the ability of this framework to process and analyse big data more effectively. This contribution is especially relevant to big data applications, where the ability to cluster large volumes of records dynamically can lead to more accurate and efficient data analysis. The framework's capacity to handle complex datasets with minimal human intervention further underscores its potential for real-world applications, particularly in fields requiring high data accuracy and reliability.

In conclusion, this research not only advances the theoretical understanding of identity resolution techniques but also offers practical solutions for addressing the challenges associated with big data and unsupervised learning. By leveraging a hybrid model and innovative clustering method, the framework developed in this study provides a robust tool for identity resolution in various domains, including law enforcement and beyond. Additionally, the implications of this research extend to consumer behaviour analysis in business contexts. Companies increasingly require a 360-degree view of consumers, encompassing their activities across multiple devices, apps, and pre-account interactions. The framework's ability to intelligently resolve identities by linking

disparate data points offers businesses a powerful means to achieve this comprehensive view. This, in turn, opens up opportunities for more sophisticated personalisation strategies, where businesses can tailor their offerings based on a holistic understanding of consumer behaviour. By accurately matching records and identifying links between different consumer profiles, businesses can deliver highly targeted and relevant experiences, enhancing customer engagement and loyalty. Thus, the identity resolution framework developed in this study holds significant potential for applications in consumer analytics, enabling businesses to navigate the complexities of modern digital behaviour with greater precision and insight.

However, this research has some limitations. The analysis was conducted using a limited de-identified policing dataset, comprising only a single database table, with other relational database tables absent. This limitation hinders efficient record linkage and determination of true identities. Moreover, the research did not focus on addressing missing information within records. As a result, records with incomplete information were separated into different datasets, and the framework was applied only to records with complete information. This approach restricts the scope for utilising other attributes within the dataset. Future studies could further explore methods to fully use all available information.

5.1. Future work

This research has demonstrated that the proposed framework can produce matched records that can be utilised to identify individuals. While the framework shows significant potential benefits for law enforcement agencies in identity resolution, its applications extend beyond this domain. For instance, it could be employed within the financial sector to detect fraud by identifying individuals attempting to manipulate institutions through different identities.

Nonetheless, there are several opportunities for future research to enhance and extend the current framework. The following areas are proposed for further investigation:

I. Introduction of a Weighting System for Attribute Matching:

One of the key improvements that could be made is incorporating a weighting system into the current matching process. While the existing framework generates an aggregate score based on the similarity of strings, adding a weighting mechanism would allow for more nuanced differentiation between matches. For each attribute, a corresponding weighting score could be assigned alongside the aggregate score, enabling the framework to prioritise higher-confidence matches. For example, when two strings receive similar aggregate scores, the weighting score would enable the framework to distinguish between them based on predefined criteria, such as the significance of particular attributes.

II. Development of Criteria-Based Weighting Scores:

Further to introducing a weighting system, future studies could explore the development of specific criteria for determining weighting scores. For instance, the number of common characters between two strings could be a determinant, with higher scores assigned to matches with greater character similarity. Additionally, the framework could be refined to ensure that strings are matched within the same language, as cross-linguistic matches often result in inaccuracies. For example, English-language strings should not be matched with those in other languages. In cases where such mismatches occur, the weighting system would promptly flag them. This refinement addresses a common issue in string-matching techniques, where linguistic differences are often overlooked, leading to erroneous matches.

III. Addressing Missing Data Through Machine Learning:

An essential area for future research is handling incomplete records within the dataset. The current framework relies on complete records, but many datasets contain records with missing information in one or more attributes. Future studies should investigate the use of data harvesting and data regeneration techniques, particularly those employing machine learning algorithms, to fill these gaps. By comparing incomplete records within the dataset and cross-referencing them with external data sources, such as social media profiles, it would be possible to enrich the dataset with additional

information. This approach would ensure that all records are fully populated, thereby improving the accuracy and efficacy of the matching algorithm. Furthermore, this would minimise the risk of overlooking critical data, enhancing the framework's overall capacity for identity resolution.

IV. Enhancement of the Knowledge Base with a Relational Database Structure:

Future work should also refine how matched records are stored within the framework's knowledge base. Rather than merely categorising records as "matched," "possible match," or "close or related match," it would be advantageous to implement a relational database structure. It would allow for storing additional harvested and regenerated information for each matched record, facilitating a more organised and sophisticated retrieval system. By incorporating a relational database, the knowledge base could dynamically evolve over time, updating records with new information obtained through ongoing data harvesting processes. It would lead to a more comprehensive and flexible knowledge base, enhancing the framework's reliability and accuracy in future identity resolutions.

V. Exploring Hybrid Models and Big Data Clustering for Enhanced Identity Resolution:

Moreover, future research should investigate applying hybrid models and advanced clustering techniques for big data. While this research has demonstrated the effectiveness of clustering methods such as the Mean-Shift technique in grouping similar records, further studies could explore their performance on larger datasets and in more complex scenarios. By integrating clustering methods within a hybrid model combining unsupervised learning with other machine learning approaches, future research could enhance the scalability and robustness of the identity resolution framework. Additionally, optimising these clustering techniques for real-time processing of large-scale data would be particularly valuable in domains where rapid identity resolution is critical, such as the financial services industry and online platforms. It could enable the development of more sophisticated, accurate, and scalable identity resolution systems.

REFERENCES

Aamodt, A. and Plaza, E. (1996) 'Case-based reasoning: Foundational issues, methodological variations, and system approaches', *Artificial Intelligence Communications*, 7(1), pp. 39–59.

Adderley, R., 2015. Director A E Solutions (BI).

Aiken, V.C.F. *et al.* (2019) 'Record linkage for farm-level data analytics: Comparison of deterministic, stochastic and machine learning methods', *Computers and Electronics in Agriculture*, 163(June), p. 104857. Available at: https://doi.org/10.1016/j.compag.2019.104857.

Albrecht W, Albrecht C, A, Z.M. (2008) 'Fraud Examination', p. 282.

Al-khamaiseh, K. and Alshagarin, S. (2014) 'A Survey of String Matching Algorithms', 4(7), pp. 144–156.

Altowim, Y., Kalashnikov, D. V. and Mehrotra, S. (2018) 'ProgressER: Adaptive progressive approach to relational entity resolution', *ACM Transactions on Knowledge Discovery from Data*, 12(3). Available at: https://doi.org/10.1145/3154410.

Anand, S. *et al.* (2014) 'Semi-supervised kernel mean shift clustering', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6), pp. 1201–1215. Available at: https://doi.org/10.1109/TPAMI.2013.190.

Ananthakrishna, R., Chaudhuri, S. and Ganti, V. (2002) 'Eliminating Fuzzy Duplicates in Data Warehouses', *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 586–597.

Ashish, N. and Toga, A.W. (2016) 'Name Similarity for Composite Element Name Matching'.

Balabantaray, R.C. *et al.* (2012) 'An Automatic Approximate Matching Technique Based on Phonetic Encoding for Odia Query', *IJCSI, Vol9, No3*, 9(3), pp. 439–444.

Bär, D. *et al.* (2012) 'Ukp: Computing semantic textual similarity by combining multiple content similarity measures', in *First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*. Montreal, Canada: Association for Computational Linguistics, pp. 435–440.

Barkay, N. and Rein, E.D. (2015) 'Achieving cyber identity resolution via electronic warfare techniques', in *RSA Conference, Singapore*.

Barrón-Cedeño, A. *et al.* (2010) 'Plagiarism Detection across Distant Language Pairs', *In Proceedings of the 23rd International Conference on Computational Linguistics*, (August), pp. 37–45.

Bartunov, S. *et al.* (2012) 'Joint Link-Attribute User Identity Resolution in Online Social Networks Categories and Subject Descriptors', *The Sixth SNA-KDD Workshop Proceedings* [Preprint].

Basu, J.K., Bhattacharyya, D. and Kim, T. (2010) 'Use of Artificial Neural Network in Pattern Recognition', *International Journal of Software Engineering and its Applications*, 4(2), pp. 23–34.

Baxter, R., Christen, P. and Churches, T. (2003) 'A Comparison of Fast Blocking Methods for Record Linkage', *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Workshop*, pp. 25–27.

Bezdek, J.C. (1981) *Pattern recognition with fuzzy objective function algorithms*.

Bezdek, J.C. *et al.* (1999) *Fuzzy models and algorithms for pattern recognition and image processing*, *Springer*.

Bharambe, D., Jain, S. and Jain, A. (2012) 'A Survey : Detection of Duplicate Record', *International Journal of Emerging Technology and Advanced Engineering*, 2(11).

Bhattacharya, I. and Getoor, L. (2006) 'Entity Resolution in Graphs', *Mining Graph Data*, pp. 311–344. Available at: https://doi.org/10.1002/9780470073049.ch13.

Bhattacharya, I. and Getoor, L. (2007) 'Collective entity resolution in relational data', *ACM Transactions on Knowledge Discovery from Data*, 1(1), pp. 5-es. Available at: https://doi.org/10.1145/1217299.1217304.

Bilenko, M. *et al.* (2003) 'Adaptative name matching in information integration', *IEEE Intelligent Systems*, 18(5).

Bilenko, M. and Mooney, R.J. (2003) 'Adaptive duplicate detection using learnable string similarity measures', *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 39–48. Available at: https://doi.org/10.1145/956755.956759.

Bird, S., Klein, E. and Loper, E. (2009) *Natural Language Processing with Python*. O'Reilly Media, Inc.

Bizer, C., Heath, T. and Berners-Lee, T. (2009) 'Linked data - The story so far', *International Journal on Semantic Web and Information Systems*, 5(3), pp. 1–22. Available at: https://doi.org/10.4018/jswis.2009081901.

Bollacker, K. *et al.* (2008) 'Freebase: A collaboratively created graph database for structuring human knowledge', *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 1247–1249. Available at: https://doi.org/10.1145/1376616.1376746.

Bonzanini, M. (2017) *Fuzzy String Matching in Python.* Available at: https://marcobonzanini.com/2015/02/25/fuzzy-string-matching-in-python/ (Accessed: 8 May 2017).

Boongoen, T. and Shen, Q. (2009) 'Intelligent hybrid approach to false identity detection', in *Proceedings of the 12th International Conference on Artificial Intelligence and Law - ICAIL '09*. New York, New York, USA: ACM Press, p. 147. Available at: https://doi.org/10.1145/1568234.1568251.

Branting, L.K. (2003) 'A comparative evaluation of name-matching algorithms', *Proceedings of the 9th international conference on Artificial intelligence and law - ICAIL '03*, p. 224. Available at: https://doi.org/10.1145/1047788.1047837.

Brizan, D.G. and Tansel, A.U. (2006) 'A Survey of Entity Resolution and Record Linkage Methodologies', *Communications of the IIMA*, 6(3), pp. 41–50.

Brown, D.E. and Hagen, S. (2002) 'Data association methods with applications to law enforcement', *Decision Support Systems*, 34(4), pp. 369–378. Available at: https://doi.org/10.1016/S0167-9236(02)00064-7.

Buscaldi, D. *et al.* (2012) 'IRIT : Textual Similarity Combining Conceptual Similarity with an N-Gram Comparison Method', in *First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*. Montreal, Canada: Association for Computational Linguistics, pp. 552–556.

Carreira-Perpiñán, M.Á. (2015) 'A review of mean-shift algorithms for clustering', pp. 1–28. Available at: http://arxiv.org/abs/1503.00687.

Chaudhuri, S. *et al.* (2003) 'Robust and efficient fuzzy match for online data cleaning', in *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data*. New York, NY, USA: ACM, pp. 313–324. Available at: https://doi.org/10.1145/872794.872796.

Christen, P. (2007) 'A two-step classification approach to unsupervised record linkage', (Clarke), pp. 111–119.

Christen, P. (2012) 'Concepts and Techniques for Record Linkage ', *Springer* [Preprint].

Christen, P., Vatsalan, D. and Wang, Q. (2016) 'Efficient entity resolution with adaptive and interactive training data selection', *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2016-Janua, pp. 727–732. Available at: https://doi.org/10.1109/ICDM.2015.63.

Christopher Jaisunder, G., Ahmed, I. and Mishra, R.K. (2017) 'Need for Customized Soundex based Algorithm on Indian Names for Phonetic Matching', *Global Journal of Enterprise Information System*, 8(2), p. 30. Available at: https://doi.org/10.18311/gjeis/2016/7658.

Christopher M. Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Chung, C.T. *et al.* (2014) 'Person Identification between Different Online Social Networks', in *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. IEEE, pp. 94–101. Available at: https://doi.org/10.1109/WI-IAT.2014.21.

Cochinwala, M. *et al.* (2001) 'Efficient data reconciliation', *Information Sciences*, 137(1–4), pp. 1–15. Available at: https://doi.org/10.1016/S0020-0255(00)00070-0.

Cohen, W.W. (1998) 'Integration of heterogeneous databases without common domains using queries based on textual similarity', in *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, pp. 201–212. Available at: https://doi.org/10.1145/276305.276323.

Cohen, W.W., Ravikumar, P. and Fienberg, S.E. (2003) 'A comparison of string metrics for matching names and records', *KDD Workshop on Data Cleaning and Object Consolidation*, 3, pp. 73–78. Available at: https://doi.org/citeulike-article-id:964346.

Cohen, W.W. and Richman, J. (2002) 'Learning to match and cluster large high-dimensional data sets for data integration', *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 475–480. Available at: https://doi.org/10.1145/775107.775116.

Cohn, D., Atlas, L. and Ladner, R. (1994) 'improving Generaliztion with active learning', *Machine Learning*, 15(2), pp. 201–221.

Comaniciu, D. and Meer, P. (2002) 'Mean shift: A robust approach toward feature space analysis', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), pp. 603–619. Available at: https://doi.org/10.1109/34.1000236.

Comaniciu, D. and Meet, P. (1999) 'Mean shift analysis and applications', *Proceedings of the IEEE International Conference on Computer Vision*, 2(3), pp. 1197–1203. Available at: https://doi.org/10.1109/iccv.1999.790416.

Culotta, A. and McCallum, A. (2005) 'Joint deduplication of multiple record types in relational data', *International Conference on Information and Knowledge Management, Proceedings*, pp. 257–258. Available at: https://doi.org/10.1145/1099554.1099615.

Dai, A.M. (2011) 'The Grouped Author-Topic Model for Unsupervised Entity Resolution', *Artificial Neural Networks and Machine Learning – ICANN 2011*, 6791(May 2014). Available at: https://doi.org/10.1007/978-3-642-21735-7.

Datacamp.com (no date) *Fuzzy String Matching in Python Tutorial | DataCamp*. Available at: https://www.datacamp.com/tutorial/fuzzy-string-python (Accessed: 14 March 2019).

Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977) ' Maximum Likelihood from Incomplete Data Via the EM Algorithm ', *Journal of the Royal Statistical Society: Series B (Methodological)*, pp. 1–22. Available at: https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

Dey, D., Mookerjee, V.S. and Liu, D. (2011) 'Efficient techniques for online record linkage', *IEEE Transactions on Knowledge and Data Engineering*, 23(3), pp. 373–387. Available at: https://doi.org/10.1109/TKDE.2010.134.

Dharmarajan, A. and Velmurugan, T. (2013) 'Applications of partition based clustering algorithms: A survey', *2013 IEEE International Conference on Computational Intelligence and Computing Research, IEEE ICCIC 2013* [Preprint]. Available at: https://doi.org/10.1109/ICCIC.2013.6724235.

Docs.python.org (2017) *1. Whetting Your Appetite — Python 2.7.13 documentation.* Available at: https://docs.python.org/2.7/tutorial/appetite.html (Accessed: 8 May 2017).

Draisbach, U. and Naumann, F. (2011) 'A generalization of blocking and windowing algorithms for duplicate detection', *Proceedings - 2011 International Conference on Data and Knowledge Engineering, ICDKE 2011*, pp. 18–24. Available at: https://doi.org/10.1109/ICDKE.2011.6053920.

Du, K.L. (2010) 'Clustering: A neural network approach', *Neural Networks*, 23(1), pp. 89–107. Available at: https://doi.org/10.1016/j.neunet.2009.08.007.

Dun & Bradstreet (2013) 'Identity Resolution Enables Informed Decision Making', (January), pp. 1–6. Available at: https://www.dnb.com/content/dam/english/dnb-solutions/identity_resolution_enables_informed_decision_making_2013_01.pdf.

Duncan, J. *et al.* (2015) 'Building an Ontology for Identity Resolution in Healthcare and Public Health', *Online Journal of Public Health Informatics*, 7(2), pp. 1–17. Available at: https://doi.org/10.5210/ojphi.v7i2.6010.

Edwards, M. *et al.* (2016) 'Sampling Labelled Profile Data for Identity Resolution', pp. 540–547.

Elfeky, M.G., Verykios, V.S. and Elmagarmid, A.K. (2002) 'TAILOR: a record linkage toolbox', *Proceedings 18th International Conference on Data Engineering*, pp. 17–28. Available at: https://doi.org/10.1109/ICDE.2002.994694.

Elmagarmid, A.K., Ipeirotis, P.G. and Verykios, V.S. (2007) 'Duplicate record detection: A survey', *IEEE Transactions on Knowledge and Data Engineering*, 19(1), pp. 1–16. Available at: https://doi.org/10.1109/TKDE.2007.250581.

Fellegi, I.P. and Sunter, A.B. (1969) 'A Theory for Record Linkage', *Source Journal of the American Statistical Association*, 64(328), pp. 1183–1210. Available at: https://doi.org/10.1080/01621459.1969.10501049.

Filippone, M. *et al.* (2008) 'A survey of kernel and spectral methods for clustering', *Pattern Recognition*, 41(1), pp. 176–190. Available at: https://doi.org/10.1016/j.patcog.2007.05.018.

Fionn Murtagh, P.L. (2014) 'Feature Relevance in Ward's Hierarchical Clustering Using the Lp Norm', *Journal of Classification*, 32, pp. 274–295. Available at: https://doi.org/10.1007/s00357-014-9161-z.

Fukunaga, K. and Hostetler, L.D. (1975) 'The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition', *IEEE Transactions on Information Theory*, 21(1), pp. 32–40. Available at: https://doi.org/10.1109/TIT.1975.1055330.

Georgescu, Shimshoni and Meer (2003) 'Mean shift based clustering in high dimensions: a texture classification example', in *Proceedings Ninth IEEE International Conference on Computer Vision*. IEEE, pp. 456–463 vol.1. Available at: https://doi.org/10.1109/ICCV.2003.1238382.

Ghahramani, Z. (2015) 'Probabilistic machine learning and artificial intelligence', *Nature*, 521(7553), pp. 452–459. Available at: https://doi.org/10.1038/nature14541.

Ghassabeh, Y.A. (2013) 'On the convergence of the mean shift algorithm in the one-dimensional space', *Pattern Recognition Letters*, 34(12), pp. 1423–1427. Available at: https://doi.org/10.1016/j.patrec.2013.05.004.

Godby, J. *et al.* (2009) 'Who's who in your digital collection? Developing a tool for name disambiguation and identity resolution', *Proceedings of the Chicago Colloquium on Digital Humanities and Computer Science*, pp. 1–17. Available at: https://letterpress.uchicago.edu/index.php/jdhcs/article/view/58.

Gomaa, W. and Fahmy, A. (2013) 'A survey of text similarity approaches', *International Journal of Computer Applications*, 68(13), pp. 13–18. Available at: https://doi.org/10.5120/11638-7118.

Gravano, L., Ipeirotis, P.G., Jagadish, H.V., *et al.* (2001) 'Approximate String Joins in a Database (Almost) for Free', *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases*, pp. 491–500.

Gravano, L., Ipeirotis, P.G., Jagadish, H. v., *et al.* (2001) 'Using q-grams in a DBMS for Approximate String Processing', *IEEE Data Eng. Bull.*, 24(4), pp. 28–34. Available at: https://doi.org/10.1.1.14.6009.

Gravano, L. *et al.* (2003) 'Text joins for data cleansing and integration in an RDBMS', in *Proceedings of IEEE 9th International Conference on Data Engineering*. IEEE, pp. 729–731. Available at: https://doi.org/10.1109/ICDE.2003.1260850.

Hagberg, A.A., Schult, D.A. and Swart, P.J. (2008) 'Exploring network structure, dynamics, and function using NetworkX', *7th Python in Science Conference (SciPy 2008)*, (SciPy), pp. 11–15.

Halkidi, M., Batistakis, Y. and Vazirgiannis, M. (2001) 'On clustering validation techniques', *Journal of Intelligent Information Systems*, 17(2–3), pp. 107–145. Available at: https://doi.org/10.1023/A:1012801612483.

Hanneman, R.A. and Riddle, M. (2005) *Introduction to social network methods*, *Introduction to social network methods*. University of California, Riverside.

Hasan, S.S. and Ahmed, F. (2015) 'Approximate String Matching Algorithms : A Brief Survey and Comparison', 120(8), p. 8887.

Hastie, T., Tibshirani, R. and Friedman, J. (2009) *The Elements of Statistical Learning*. 2nd Editio. Springer.

Hedayati, A. (2012) 'An analysis of identity theft : Motives, related frauds, techniques and prevention', *Journal of Law and Conflict Resolution*, 4(January), pp. 1–12. Available at: https://doi.org/10.5897/JLCR11.044.

Hernandez, M.A. and Stolfo, S.J. (1998) 'Real-World Data Is Dirty-Data Cleansing and the Merge or Purge Problem', *Data Mining and Knowledge Discovery*, 2(1), pp. 9–37. Available at: https://doi.org/10.1023/A.

Herzog, T. and Reiter, J. (2008) *Data Quality and Record Linkage Techniques Data Quality and Record Linkage Techniques . Thomas N. Herzog , Fritz J. Scheuren , and William E. Winkler . New York : Springer , 2007 . ISBN 978-0-387-69502-0 . xiii + 227 pp. $44.95 (P).*, *Journal of the American Statistical Association*. Available at: https://doi.org/10.1198/jasa.2008.s229.

Huang, L. *et al.* (2008) 'Duplicate Records Cleansing with Length Filtering and Dynamic Weighting', *2008 Fourth International Conference on Semantics, Knowledge and Grid*, (2003), pp. 95–102. Available at: https://doi.org/10.1109/SKG.2008.88.

Huang, Z. (1998) 'Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. Data Mining and Knowledge Discovery 2, 283-304', *Data Mining and Knowledge Discovery*, 2(3), pp. 283–304.

Islam, A. and Inkpen, D. (2008) 'Semantic text similarity using corpus-based word similarity and string similarity', *ACM Transactions on Knowledge Discovery from Data*, 2(2), pp. 1–25. Available at: https://doi.org/10.1145/1376815.1376819.

Jain, a. K., Murty, M.N. and Flynn, P.J. (1999) 'Data clustering: a review', *ACM Computing Surveys*, 31(3), pp. 264–323. Available at: https://doi.org/10.1145/331499.331504.

Jain, A.K., Ross, A. and Prabhakar, S. (2004) 'An Introduction to Biometric Recognition', *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1), pp. 4–20. Available at: https://doi.org/10.1109/TCSVT.2003.818349.

Jaro, M.A. (1989) 'Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida', *Journal of the American Statistical Association*, 84(406), pp. 414–420. Available at: https://doi.org/10.1080/01621459.1989.10478785.

Jetbrains.com. (2017) *1 Help :: Meet PyCharm.* Available at: https://www.jetbrains.com/help/pycharm/2017.1/meet-pycharm.html (Accessed: 8 May 2017).

John S. Pistole (2003) *FBI — Fraudulent Identification Documents and the Implications for Homeland Security*, *Before the House Select Committee On Homeland Security Washington DC*. Available at: https://archives.fbi.gov/archives/news/testimony/fraudulent-identification-

documents-and-the-implications-for-homeland-security (Accessed: 10 December 2015).

Jonas, J. (2006) 'Threat and Fraud Intelligence, Las Vegas Style', *Security & Privacy, IEEE*, 4(6), pp. 28–34. Available at: https://doi.org/AA7908A8-D2AE-4C76-848D-421866FB1018.

Jurek, A. *et al.* (2017) 'A novel ensemble learning approach to unsupervised record linkage', *Information Systems*, 71, pp. 40–54. Available at: https://doi.org/10.1016/j.is.2017.06.006.

Kessler, B. (2005) 'Phonetic comparison algorithms', *Transactions of the Philological Society*, 103(2), pp. 243–260. Available at: https://doi.org/10.1111/j.1467-968X.2005.00153.x.

Kirsten, T. *et al.* (2010) 'Data Partitioning for Parallel Entity Matching', *Strategies*, 3(2), p. 11.

Kloo, I., Dabkowski, M.F. and Huddleston, S.H. (2019) 'Improving Record Linkage for Counter-Threat Finance Intelligence with Dynamic Jaro-Winkler Thresholds', *Proceedings - Winter Simulation Conference*, 2019-Decem, pp. 2467–2478. Available at: https://doi.org/10.1109/WSC40007.2019.9004945.

Kohonen, T. (1990) 'The Self-organizing Map', 78(9), pp. 1464–1480.

Koneru, K., Pulla, V.S.V. and Varol, C. (2016) 'Performance Evaluation of Phonetic Matching Algorithms on English Words and Street Names - Comparison and Correlation', (Data), pp. 57–64. Available at: https://doi.org/10.5220/0005926300570064.

Kopcke, H. and Rahm, E. (2010) 'Frameworks for entity matching: A comparison', *Data and Knowledge Engineering*, 69(2), pp. 197–210. Available at: https://doi.org/10.1016/j.datak.2009.10.003.

Kukich, K. (1992) 'Technique for automatically correcting words in text', *ACM Computing Surveys*, 24(4), pp. 377–439. Available at: https://doi.org/10.1145/146370.146380.

Lait, A. and Randell, B. (1996) 'An assessment of name matching algorithms', *Technical Report Series-University of Newcastle upon Tyne*, pp. 1–32. Available at: http://homepages.cs.ncl.ac.uk/brian.randell/Genealogy/NameMatching.pdf.

Lait, A.J. and Randell, B. (1996) 'An Assessment of Name Matching Algorithms Department of Computing Science University of Newcastle upon Tyne Abstract 1 . Name Variations 2 . Current Name-Matching Methods', pp. 1–32.

Li, J. and Wang, A. (2015) 'A framework of identity resolution: evaluating identity attributes and matching algorithms', *Security Informatics*, 4(1), p. 6. Available at: https://doi.org/10.1186/s13388-015-0021-0.

Li, J. and Wang, G.A. (2011) 'Criminal Identity Resolution Using Social Behavior and Relationship Attributes', pp. 173–175.

Li, X., Hu, Z. and Wu, F. (2007) 'A note on the convergence of the mean shift', *Pattern Recognition*, 40(6), pp. 1756–1762. Available at: https://doi.org/10.1016/j.patcog.2006.10.016.

Liao, X. and Zhao, Z. (2019) 'Unsupervised approaches for textual semantic annotation, a survey', *ACM Computing Surveys*, 52(4). Available at: https://doi.org/10.1145/3324473.

Lim, E.-P. *et al.* (1993) 'Entity identification in database integration', *Proceedings of IEEE 9th International Conference on Data Engineering*, pp. 294–301. Available at: https://doi.org/10.1109/ICDE.1993.344053.

Lisbach, B. and Meyer, V. (2013) 'Name Matching and Identity Matching BT - Linguistic Identity Matching', in B. Lisbach and V. Meyer (eds). Wiesbaden: Springer Fachmedien Wiesbaden, pp. 172–192. Available at: https://doi.org/10.1007/978-3-8348-2095-2_12.

Liu, Y. *et al.* (2013) 'Dynamics of a mean-shift-like algorithm and its applications on clustering', *Information Processing Letters*, 113(1–2), pp. 8–16. Available at: https://doi.org/10.1016/j.ipl.2012.10.002.

Lloyd, S.P. (1982) 'Least Squares Quantization in PCM', *IEEE Transactions on Information Theory*, 28(2), pp. 129–137. Available at: https://doi.org/10.1109/TIT.1982.1056489.

Von Luxburg, U. (2007) 'A tutorial on spectral clustering', *Statistics and Computing*, 17(4), pp. 395–416. Available at: https://doi.org/10.1007/s11222-007-9033-z.

Mamun, A. Al, Aseltine, R. and Rajasekaran, S. (2014) 'Poster: Efficient record linkage techniques', *2014 IEEE 4th International Conference on Computational Advances in Bio and Medical Sciences, ICCABS 2014*, p. 4673. Available at: https://doi.org/10.1109/ICCABS.2014.6863930.

Martin Ester, Hans-Peter Kriegel, Jiirg Sander, X.X. (1996) 'A Density-Based Algorithm for Discovering Clusters', *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 226–231. Available at: https://doi.org/10.1016/B978-0-444-64165-6.03005-6.

McCallum, A., Nigam, K. and Ungar, L.H. (2000) 'Efficient clustering of high-dimensional data sets with application to reference matching', in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, pp. 169–178. Available at: https://doi.org/10.1145/347090.347123.

McCallum-Bayliss, H. (2004) 'Identity resolution in a global environment: Fishing for people in a sea of names', *IT Professional*, 6(6), pp. 21–26. Available at: https://doi.org/10.1109/MITP.2004.81.

McKinney, W. (2013) *Python for Data Analysis*. 1st Editio. O'Reilly Media, Inc.

M.E.J.Newman (2010) *Networks: An Introduction*. Oxford University Press.

Mon, A.C., Mie, M. and Thwin, S. (2013) 'Effective Blocking for Combining Multiple Entity Resolution Systems', 2(04), pp. 126–136.

Monge, A.E. and Elkan, C.P. (1996) 'The field matching problem: Algorithms and applications', *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, (Slaven 1992), pp. 267–270. Available at: https://doi.org/10.1.1.23.9685.

Monge, A.E. and Elkan, C.P. (1997) 'An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records', *Proceedings of the SIGMOD 1997 workshop on research issues on data mining and knowledge discovery*, pp. 23–29.

Naseem, T. and Hussain, S. (2007) 'A novel approach for ranking spelling error corrections for Urdu', *Language Resources and Evaluation*, 41(2), pp. 117–128. Available at: https://doi.org/10.1007/s10579-007-9028-6.

Nawaz, A. and Kazemian, H. (2021) 'A Fuzzy Approach to Identity Resolution', *Proceedings of the 22nd Engineering Applications of Neural Networks Conference EANN 2021*, 3, pp. 307–318. doi: 10.1007/978-3-030-80568-5_26.

Needleman, S.B. and Wunsch, C.D. (1970) 'A general method applicable to the search for similiarities in the amino acid sequence of two proteins', *Journal of molecular biology*, 48(3), pp. 443–453.

Niblett, G. (2015) 'Identity', *ITNOW*, 57(3), p. 23. Available at: https://doi.org/10.1093/itnow/bwv066.

Nisha and Kaur, P.J. (2015) 'A Survey of Clustering Techniques and Algorithms', in *2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. New Delhi: IEEE, pp. 304–307.

Noy, N.F. and McGuinness, D.L. (2001) 'Ontology Development 101: A Guide to Creating Your First Ontology', *Stanford Knowledge Systems Laboratory*, (January 2001), p. 25. Available at: https://doi.org/10.1016/j.artmed.2004.01.014.

Oracle (2010) 'Identity Resolution and Data Quality Algorithms for Master Person Index', *An Oracle White Paper*, (August), pp. 1–16.

Pandas.pydata.org. (2017) *pandas: powerful Python data analysis toolkit — pandas 0.20.1 documentation.* Available at: http://pandas.pydata.org/pandas-docs/stable/ (Accessed: 8 May 2017).

Papadakis, G. *et al.* (2011) 'To Compare or Not to Compare: Making Entity Resolution More Efficient', *Proceedings of the International Workshop on Semantic Web Information Management*, pp. 3:1--3:7. Available at: https://doi.org/10.1145/1999299.1999302.

Pasula, H. *et al.* (2003) 'Identity uncertainty and citation matching', *Advances in Neural Information Processing Systems*, pp. 1425–1432. Available at: https://doi.org/10.1.1.15.8644.

Patman, F. and Shaefer, L. (2001) 'Is Soundex good enough for you? On the hidden risks of Soundex-based name searching', *Language Analysis Systems, Inc.*, pp. 1–31. Available at: http://www.surnamestudies.org/teaching/soundex.pdf.

Pedneault, S. *et al.* (2012) *Forensic Accounting and Fraud Investigation*. 3rd Editio, *John Wiley & Sons*. 3rd Editio. The CPE Store, Inc.

Phillips, M., Amirhosseini, M.H. and Kazemian, H.B. (2020) 'A Rule and Graph-Based Approach for Targeted Identity Resolution on Policing Data', *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020*, pp. 2077–2083. Available at: https://doi.org/10.1109/SSCI47803.2020.9308182.

Pilania, A. and Kumaran, G.M.M. (2019) 'Comparative study of name matching algorithms', *Proceedings of the 2019 6th International Conference on Computing for Sustainable Global Development, INDIACom 2019*, pp. 1174–1178.

Ravikumar, P.D. and Cohen, W.W. (2004) 'A Hierarchical Graphical Model for Record Linkage', in *Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence*. Arlington, Virginia, United States: AUAI Press, pp. 454–461. Available at: https://doi.org/http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.3.3138.

Ristad, E.S. and N.yianilos, P. (1998) 'Learning string-edit distance', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5), pp. 522–532. Available at: https://doi.org/10.1109/34.682181.

Rousseeuw, P.J. (1987) 'Silhouettes: A graphical aid to the interpretation and validation of cluster analysis', *Journal of Computational and Applied Mathematics*, 20(C), pp. 53–65. Available at: https://doi.org/10.1016/0377-0427(87)90125-7.

Sarawagi, S. *et al.* (2002) 'Alias: An active learning led interactive deduplication system', *VLDB '02 Proceedings of the 28th international conference on Very Large Data Bases*, pp. 1103–1106.

Sarawagi, S. and Bhamidipaty, A. (2002) 'Interactive Deduplication using Active Learning', *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 269–278. Available at: https://doi.org/10.1145/775047.775087.

Sayers, A. *et al.* (2016) 'Probabilistic record linkage', *International Journal of Epidemiology*, 45(3), pp. 954–964. Available at: https://doi.org/10.1093/ije/dyv322.

Schubert, E. *et al.* (2017) 'Why and How You Should (Still) Use DBSCAN', *ACM Transactions on Database Systems*, 42(3), pp. 1–21.

Schult, D. (1943) 'LA-UR- EXPLORING NETWORK STRUCTU Exploring network structure , dynamics , and function using NetworkX', 836.

Shah, R. and Kumar Singh, D. (2014) 'Analysis and Comparative Study on Phonetic Matching Techniques', *International Journal of Computer Applications*, 87(9), pp. 14–17. Available at: https://doi.org/10.5120/15236-3771.

Singla, P. and Domingos, P. (2004) 'Multi-Relational Record Linkage', *Proceedings of the 3rd KDD Workshop on Multi-Relational Data Mining*, pp. 31–48.

Smith, T.F. and Waterman, M.S. (1981) 'Identification of Common Molecular Subsequences', *J. Mol. Biol.*, 147, pp. 195–197. Available at: https://doi.org/10.1016/0022-2836(81)90087-5.

Snae, C. (2007) 'A Comparison and Analysis of Name Matching Algorithms', *Engineering and Technology*, 1(1), pp. 252–257. Available at: https://waset.org/publications/8664/a-comparison-and-analysis-of-name-matching-algorithms-.

Soltani, R. and Abhari, A. (2013) 'Identity matching in social media platforms', *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2013 International Symposium on*, pp. 64–70.

Soundex, D. (2015) *Databularium*. Available at: databularium.com/en/2015/08/22/approximate-string-matching-algorithms/ 1/10 (Accessed: 20 September 2005).

Studer, R., Benjamins, V.R. and Fensel, D. (1998) 'Knowledge Engineering: Principles and methods', *Data and Knowledge Engineering*, 25(1–2), pp. 161–197. Available at: https://doi.org/10.1016/S0169-023X(97)00056-6.

Studiawan, H., Payne, C. and Sohel, F. (2017) 'Graph clustering and anomaly detection of access control log for forensic purposes', *Digital Investigation*, 21(May), pp. 76–87. Available at: https://doi.org/10.1016/j.diin.2017.05.001.

Sun, S. (2013) 'A survey of multi-view machine learning', *Neural Computing and Applications*, 23(7–8), pp. 2031–2038. Available at: https://doi.org/10.1007/s00521-013-1362-6.

Sun, Y. (2015) 'A Comparative Evaluation of String Similarity Metrics for Ontology Alignment', *Journal of Information and Computational Science*, 12(3), pp. 957–964. Available at: https://doi.org/10.12733/jics20105420.

Sutinen, E. and Tarhio, J. (1995) 'On using q-gram locations in approximate string matching', *Proc. Third Ann. European Symp. Algorithms (ESA '95)*, pp. 327–340. Available at: https://doi.org/10.1007/3-540-60313-1_153.

Tejada, S., Knoblock, C. a. and Minton, S. (2002) 'Learning Domain-Independent String Transformation Weights for High Accuracy Object Identification', *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 350–359. Available at: https://doi.org/10.1145/775047.775099.

Ukkonen, E. (1992) 'Approximate string-matching with q-grams and maximal matches', *Theoretical Computer Science*, 92(1), pp. 191–211. Available at: https://doi.org/10.1016/0304-3975(92)90143-4.

US Department of State (2008) 'Country Reports on Terrorism 2007', (April), pp. 1–312.

VanderPlas, J. (2017) *Python Data Science Handbook*. O'Reilly Media, Inc.

Wang, F. and Wang, H. (2016) 'Record Linkage Using the Combination of Twice Iterative SVM Training and Controllable Manual Review.', *DASC/PiCom/DataCom/CyberSciTech*, pp. 31–38. Available at: https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2016.21.

Wang, G., Chen, H. and Atabakhsh, H. (2004) 'Automatically detecting deceptive criminal identities', *Communications of the ACM*, 47(3), pp. 70–76. Available at: https://doi.org/10.1145/971617.971618.

Wang, J. and Dong, Y. (2020) 'Measurement of Text Similarity: A Survey', *Information*, 11, pp. 1–17. Available at: https://doi.org/10.3390/info11090421.

Wang, Y.R. and Madnick, S.E. (1989) 'The inter-database instance identification problem in integrating autonomous systems', *Proceedings of the Fifth International Conference on Data Engineering*, pp. 46–55. Available at: https://doi.org/10.1109/ICDE.1989.47199.

Wangikar, V., Deshmukh, S. and Bhirud, S. (2016) 'Study and Implementation of Record De-duplication Algorithms', *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies - ICTCS '16*, (January), pp. 1–6. Available at: https://doi.org/10.1145/2905055.2905063.

Ward, J.H. (1963) 'Hierarchical Grouping to Optimize an Objective Function', *Journal of the American Statistical Association*, pp. 236–244. Available at: https://doi.org/10.1080/01621459.1963.10500845.

Waterman, M.S., Smith, T.F. and Beyer, W.A. (1976) 'Some biological sequence metrics', *Advances in Mathematics*, 20(3), pp. 367–387. Available at: https://doi.org/10.1016/0001-8708(76)90202-4.

Watson, I. and Marir, F. (2007) 'Case-Based Reasoning : A Review', *Published in The Knowledge Engineering Review*, 9(4), pp. 1–34.

What is Soundex and How Does Soundex Work? (page 2). Available at: http://www.genealogyintime.com/GenealogyResources/Articles/what_is_soundex_and_how_does_soundex_work_page2.html (Accessed: 26 November 2022).

Winkler, W. (1994) 'Advanced methods for record linkage', (1991), p. 24. Available at: https://doi.org/10.1.1.39.3724.

Winkler, W.E. (1990) 'String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage', *Proceedings of the Section on Survey Research*, pp. 354–359.

Winkler, W.E. (2006) 'Overview of Record Linkage and Current Research Directions', *Statistical Research Division*, (2), pp. 1–44. Available at: https://doi.org/10.1206/3728.2.

WONG, J.A.H. and M.A. (1979) 'Algorithm AS 136: A k-means clustering algorithm', *ournal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1), pp. 100–108.

Wu, K.L. and Yang, M.S. (2007) 'Mean shift-based clustering', *Pattern Recognition*, 40(11), pp. 3035–3052. Available at: https://doi.org/10.1016/j.patcog.2007.02.006.

Xu, R. and Wunsch, D. (2005) 'Survey of clustering algorithms', *IEEE Transactions on Neural Networks*, 16(3), pp. 645–678. Available at: https://doi.org/10.1109/TNN.2005.845141.

Yadav, S., Sinha, A. and Kumar, P. (2019) 'Multi-attribute identity resolution for online social network', *SN Applied Sciences*, 1(12), pp. 1–15. Available at: https://doi.org/10.1007/s42452-019-1701-z.

Yan, B., Bajaj, L. and Bhasin, A. (2011) 'Entity Resolution Using Social Graphs for Business Applications', in *2011 International Conference on Advances in Social Networks Analysis and Mining*. IEEE, pp. 220–227. Available at: https://doi.org/10.1109/ASONAM.2011.119.

Yan, Y. *et al.* (2020) 'Entity Matching in the Wild: A Consistent and Versatile Framework to Unify Data in Industrial Applications', *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 2287–2301. Available at: https://doi.org/10.1145/3318464.3386143.

Yerva, S.R., Miklos, Z. and Aberer, K. (2010) 'Towards better entity resolution techniques for Web document collections', in *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*. IEEE, pp. 209–214. Available at: https://doi.org/10.1109/ICDEW.2010.5452698.

Yi, F. *et al.* (2020) 'Cybersecurity Named Entity Recognition Using Multi-Modal Ensemble Learning', *IEEE Access*, 8, pp. 63214–63224. Available at: https://doi.org/10.1109/ACCESS.2020.2984582.

Yizong Cheng (1995) 'Mean shift, mode seeking, and clustering', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8), pp. 790–799. Available at: https://doi.org/10.1109/34.400568.

Appendix (A) – Research Publication

A Fuzzy Approach to Identity Resolution

Asif Nawaz(✉) and Hassan Kazemian

Intelligent Systems Research Centre, School of Computing and Digital Media, London Metropolitan University, Holloway Road, London, UK

asn0144@my.londonmet.ac.uk

Abstract. Identity resolution is crucial for law enforcement agencies globally and a difficult task to match the real-world identity in big data due to data inconsistency e.g. typographical errors, naming variation, and abbreviations. The fuzzy approach to identity resolution has been introduced that uses Soundex and Jaro-Winkler distance algorithms in a cascaded manner to calculate an aggregate score for the full name. While the Edit-distance algorithm is used to score the address and ethnicity description attributes. The Soundex code has been modified to numbers only with increased code length to 6-digits for this fuzzy approach. This allowed the matching algorithm to overcome some of the Soundex code limitations of name matching. The approach accommodates three different variations of name for an iterative search process that retrieves matched records based on inputs. In the experiment, searching for a suspect in two different cases, the initial search retrieved 173 and 52 records for each target suspect. These records were grouped using the Mean-Shift clustering technique based on the similarity score of three attributes. For further analysis, the segmentation process of records matched 16 and 22 records for each case respectively, and graph analysis matched the target suspect identity out of other matched identities with links association to different addresses. The overall matching performance of this fuzzy approach is encouraging, and it can benefit law enforcement agencies to speed up the investigation process and most importantly can help to identify the suspect with even minimal information available.

Keywords: Fuzzy string matching · Identity resolution · Graph analysis · Soundex · Jaro-Winkler

1 Introduction

Identity Resolution is not just matching information but to detect, identify and consider past information or associations for the target entity. Fraud and other crimes are a globally major ongoing threat for law enforcement agencies and other institutions to identify the real-world identity from a pool of false or similar identities. Identity is the property and characteristics of an entity that helps to differentiate entities from each other. Each entity [1] has attributes and can be identified by ID number, name, and date of birth as key attributes but there are many reasons which make it difficult to find the correct identity. But matching two records using limited attributes is not sufficient for the true identity

L. Iliadis et al. (Eds.): EANN 2021, INNS 3, pp. 307–318, 2021.
https://doi.org/10.1007/978-3-030-80568-5_26

of any entity. For example, matching records by name will not resolve this issue as there can be many similar names in the database. One of the main issues is a huge amount of unstructured, incomplete, and incorrect data available to extract the required information for a particular individual [2].

The techniques used for record matching and record linkage normally classify records into three categories i.e. “Match”, “No Match” and “Possible Match”. But match or possible match may not be accurate if the information changes or not updated with time because this can lead to incorrectly flagged results as a match or possible match [3]. The record matching in [4] refers to entity resolution as information extraction using names and refers to identity resolution as a technique to determine the extracted information belongs to whom and how it is linked to others in the real world. This statement leads to find uniqueness and commonality between different datasets from different sources to answer who is who and who knows whom [5].

The data quality issues in the database make it difficult to identify one real-world entity due to various similar multiple entries in the database. Removing or resolving duplicate and similar entries can be achieved by merging and de-duplicating records in the database representing the same entity. This is referred to as record matching in [6, 7]. But missing or incomplete information in the database leads to a huge amount of manual work to guess the matching record [8]. The data grows with the passage of time and traditional record matching techniques cannot accurately find a relationship between the records.

2 Literature Review

Fuzzy string matching is a technique to match strings based on approximate pattern similarity for entity resolution. A survey about duplicate record detection [9] explores the string similarity techniques developed for fuzzy string matching. The most common technique is Levenshtein distance also known as Edit-distance to calculate the distance between two strings by applying three edit operations on the strings. The three edit operations are inserting, deleting, or substituting the characters to match any given strings. If the distance of strings is less than the set threshold value after applying edit operations, then the strings are considered a close match with slight variation. But this technique fails in the situation where strings are written in short form or abbreviation, instead of a complete word as it results in the distance incorrectly. Jaro distance is another fuzzy string matching technique, primarily to compare the short strings e.g. first and last names. This technique finds the common characters between strings while tracking the order of the characters [10]. The algorithm was enhanced by Winkler in 1990 by giving name prefix higher weighting. This variation was named as Jaro-Winkler distance. Jaro and Jaro-Winkler distance [9] algorithms cannot perform well if there is a positional difference between two strings and is more than allowed change. For example, strings “Alice bruce Bob” and “Bob bruce Alice” have allowed a change of 6 positions, but the character ‘B’ in the string “Bob” has a difference of 12 positions. In this case, only string “Bruce” will match between two strings, and the algorithms will not find the better match.

For records matching, the record linking data association method [11] was introduced to match the criminal records referring to the same individual record. This method

compares two records and calculates the total similarity measure as a weighted sum of the similarity measures of all corresponding featured values. The approach requires more computing power to calculate the total similarity as the dataset grows with time. Another similar work proposed in [1] to compare four personal attributes such as name, date of birth, social security number, and address for detecting identities by combining the overall similarity score. But the approach is limited and cannot produce accurate results if one or more attributes are missing from the dataset. It does not filter the referring records efficiently and can also ignore needed records on fewer similarity measurements. Another method discussed in [12] to remove duplicates from the dataset by applying dimensional hierarchy over the link relations such as city, state, and country. This approach matches the identity record only if both identities belong to the same area otherwise the record does not match against other similar entries in the dataset for different areas. The foreign keys in [13] were utilized in a relational database using a probabilistic relational model (PRM) for citation matching. The approach is rule-based and relies on the quality of data in the dataset and if the dataset has incomplete or missing information then the true match cannot be generated accurately.

To eliminate the duplicate records, [14] proposed another rule-based model called conditional random field model (CRF) to measure the associations among other different entities. However, [15] suggested that this approach fails and cannot find the links between similar entities. One of the interesting graph-based methods was proposed by [16] in which, between each pair of the reference entity, the relational graph created, matching is based on similarity with the same attribute that matches with similarity measure. Furthermore, the approach was enhanced [17] by adding a collective entity resolution algorithm to match social information based on already matched records to reuse it for matching more records instead of only comparing two records. To match entities from social media websites e.g. Facebook and Twitter, the model in [18] combines user profiles using different attributes into a graph by detecting the social linkages between two user profiles using a CRF-based approach. The recent work in [19] proposed a rule-based approach to score attributes and analyze links between identities using a graph-based method. The majority of the previous researches are rules-based techniques that can be very time-consuming to create a set of rules for big data.

Therefore, in this research, a fuzzy approach to identity resolution has been introduced that uses an iterative search with cascaded string similarity algorithms (Soundex code and Jaro-Winkler distance) to generate an aggregate score for the name variation. The fuzzy approach utilizes minimal attributes to retrieve the matching records from the dataset. The matched records then go through clustering processes to group similar records into clusters. For further analysis, the matching of these records is done by segmentation and graph analysis. This research carries forward previous work of [19] by using string similarity techniques with clustering, segmentation, and graph analysis.

3 Problem Definition for Identity Resolution

In the entity and identity resolution process, record matching and de-duplication is a difficult task to identify duplicates due to different attribute values. The techniques that require a human expert for manual tuning are good but unfeasible for large databases.

Such databases can be referred to as big data. It is difficult to use different techniques to train data samples for generating desired results from big data. One of the other issues is the quality of data and there are not enough techniques available or capable enough to utilize a combination of string similarity metrics to complete the matching process. This leads to unsatisfactory results because every single string similarity metric (fuzzy matching technique) is domain-specific that solves a certain problem. Missing one or more similarity metrics does not help to achieve better results. The previous approaches do not consider the following to achieve better results based on fuzzy matching:

- Use different similarity metrics techniques to calculate a matching score to extract records.
- Use similarity metric on data at different stages to output better entity matching results for record linkage and analysis.
- Use the clustering technique with help of segmentation and graph analysis for identity resolution.

4 The Proposed Fuzzy Approach to Identity Resolution

The fuzzy approach has been applied to identity resolution shown in Fig. 1 by using cascaded string similarity techniques to retrieve an approximate entity match. The fuzzy approach utilizes Soudex, Jaro-Winkler algorithms to calculate the aggerate score for names and Edit-distance to score the other attributes e.g. ethnicity description and address. The aim is to match names simply by using similarity metrics and analyze retrieved records for similarities using clustering, segmentation, and graph analysis. This fuzzy approach is implemented using Python 3.7 using PyCharm (community version) IDE and the anonymized policing dataset is stored in MS SQL Server Express 2017. Pandas (Python data analysis library) is used to clean data and store datasets retrieved during different stages. The NetworkX and matplotlib libraries are used for graph analysis and visualization.

4.1 Policing Dataset

An anonymized criminal dataset from police has been used in this research. The dataset consists of 1,146,212 records containing duplicates, typographical errors, incomplete or missing information. Some records are partial duplicates of other records with only one or more different attribute values. Each reported crime has a unique crime reference number and every individual in the dataset has a unique nominal reference number. But there are individuals with multiple nominal reference numbers. For example, there are 6,032 individuals with 2 or more similar nominal reference numbers assigned. The other attributes such as forename, surname, date of birth, gender, address, and ethnicity are representing an individual. In the dataset, there are 309,518 duplicate records based on similar surname, forename, and date of birth.

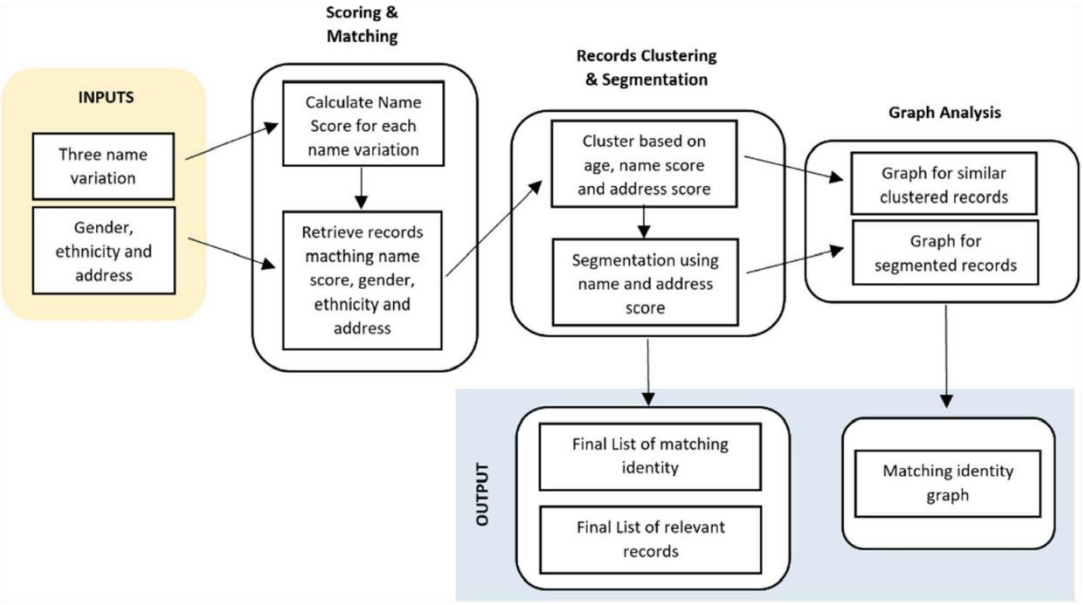

Fig. 1. The block diagram of a fuzzy approach to identity resolution

4.2 Data Pre-processing

The data pre-processing is used to eliminate missing values from the dataset for each record attribute. In the dataset, street_name, town_name, and district_name and postcode are four different fields representing the address. But in this research, the three fields are combined as one address field while keeping the postcode field separate. Similarly, the surname and forename are combined as the name field. There are records defined as 'Unknown' gender rather than 'M' for male' and 'F' for female, so in this fuzzy approach, these records are considered as missing values. Overall, there is a total of 430,293 missing values that are removed from the dataset. Therefore, the dataset after data cleaning still has 715,919 records. Considering the upper case and lower case attribute values in the dataset, all attribute values are converted to lower case. The fuzzy approach utilizes name, gender, ethnicity description, and address attribute to start the initial search of records by generating the aggregate name score.

4.3 String Matching and Aggregated Score

This fuzzy approach uses Soundex, Jaro-Winkler similarity techniques to calculate the aggregate score for name matching. The Soundex code algorithm has been modified by removing the first character of a name as a constant letter from the code and changing the length to 6-digits. This generates a numerical Soundex code of 6-digits to help eliminate the Soundex first character mismatch issue. By increasing the length of the code helps to reduce many false-positive retrievals as compared to the 4-digits code. Later, the Jaro-Winkler technique is applied to this Soundex code to get a Soundex fuzzy score (Sscore) and also calculates the Jaro-Winkler score (JWscore). All these scores are then used to calculate the aggregate score (AggScore). The aggregate score is normalized between

the fuzzy score of 0 and 1. The aggregate score is calculated with the following equation.

$$\mathrm{Agg}_{\mathrm{Score}} = (\mathrm{S}_{\mathrm{score}} + \mathrm{JW}_{\mathrm{score}}) * 0.5 \quad (1)$$

4.4 Searching and Matching Criteria

It is very important to establish search and match criteria for retrieving entities by comparing the calculated score of selected attributes. The fuzzy approach focuses on eliminating most no-match entities during the initial search, based on aggregate score and Soundex code match. To find the results as accurately as possible is important because each record contains different attributes that help to differentiate one entity from other entities. But if there are issues in the value of the attributes e.g. incomplete information or typographical error then matching of records becomes difficult. This fuzzy approach proposes an iterative process by taking three different name variations as an input for the target entity. The approach also uses gender, ethnicity description, and address attributes to retrieve records. These selected attributes help to reduce the number of records retrieved by reducing processing time.

The matching of gender is done by an exact match while matching of ethnicity description is done by a partial matching of the string. For matching addresses, the edit distance technique is used to score each address. Each iteration process is a combination of these selected attributes to generate search results as three data subsets. Once the iteration process is completed, all three data subsets are merged and compared for duplicate records to form one resultant dataset. At this stage, all duplicate records are removed from this retrieved dataset. At this stage, records are retrieved even with the low aggregate score (e.g. score of 0.50 or 0.60) but have matching Soundex code. This to make sure any possible or close-matched records are not ignored or dropped during the initial search.

4.5 Clustering, Segmentation, and Graph Analysis

In this fuzzy approach, the labeling of data from a human expert is not required to group similar records. For this purpose, the Mean-Shift clustering algorithm has been used to group similar records based on age, name aggregate score, and address score. Each record is automatically labeled with a cluster number. These clustered records are then matched and compared based on the highest name score and address score to create segments of records. This is to make sure that similar records are linked together even in different clusters. To do this, records with maximum address scores are extracted from clustered datasets to form a segment of records. In this segmentation process, the records are matched for similar addresses from the initial retrieved dataset and the clustered dataset. The similar segmented records are merged into one dataset and any other relevant records are kept separate.

For further graph analysis, these segmented records are compared with the clustered dataset to match the final identity out of all other identities. The graph creation is layer-based by using different attributes from the dataset. The first graph is created using the entity name and the clusters label for the entity. This visually list all entities linked to

each cluster. The second graph is created using the entity name and address from the segmented dataset. This graph data is then compared with the first graph to find the matched identity out of other identities and show the matched identity with associated addresses.

5 Experimental Evaluation

5.1 Target Entity

For this fuzzy approach to identity resolution, it is assumed the police investigating officer is working on two different criminal cases and during an investigation, he obtained some basic information about the suspects. The information obtained per case is listed in Table 1 and investigating officer use this information to search a suspect for each case using this fuzzy approach. The investigating officer does not know the correct name spelling of the suspect or believes the dataset has typographical errors, so he inputs the full name with three different variations. The gender of the suspect is known while ethnicity description and address details are partially known.

Table 1. The information available to investigating office about the suspect for each case

	First case	Second case
Target search (suspect)	BECH Jaunette	FAROS Abbidah
Available suspect information for input		
Name variation 1	back janette	abidah firos
Name variation 2	bach janet	abiddha feroz
Name variation 3	beck janete	abidha firoz
Gender	f	f
Ethnicity (description)	white	white
Address	town and close	brandearth hey

5.2 Searching Results

First Case. Based on the inputs for the first case in Table 1, the initial search and matching criteria produced three datasets. The name first variation retrieved 95 records; the second variation retrieved 61 records and the third variation retrieved 95 records. It is worth noting, the first and third name variation generated similar Soundex code (125300) and the aggregate score (0.70) that retrieved the same number of records. But the second variation generated a different Soundex code (122530) and retrieved a different number of records compared to the other name variations. A resultant dataset of 173 records is generated after merging all three datasets and removing duplicate records.

Second Case. Based on the inputs for the second case in Table 1, the initial search and matching criteria produced three datasets. All three name variations retrieved 41 records and interestingly the resultant dataset of 52 records is generated after merging all three datasets and removing duplicate records. All these records retrieved have some similarity or are completely different from one another, but this will be differentiated later in the next stage.

5.3 Clustering, Segmentation and Graph Analysis Results

The dataset (produced from the searching results) is fed into the clustering algorithm. This created clusters of records based on age, name aggregate score, and address score. For the first case, records are grouped into 4 clusters, and records for the second case are grouped into 5 clusters based on the similarities score of the attributes. The clustering of records is shown in Figs. 2 and 3 for both cases respectively.

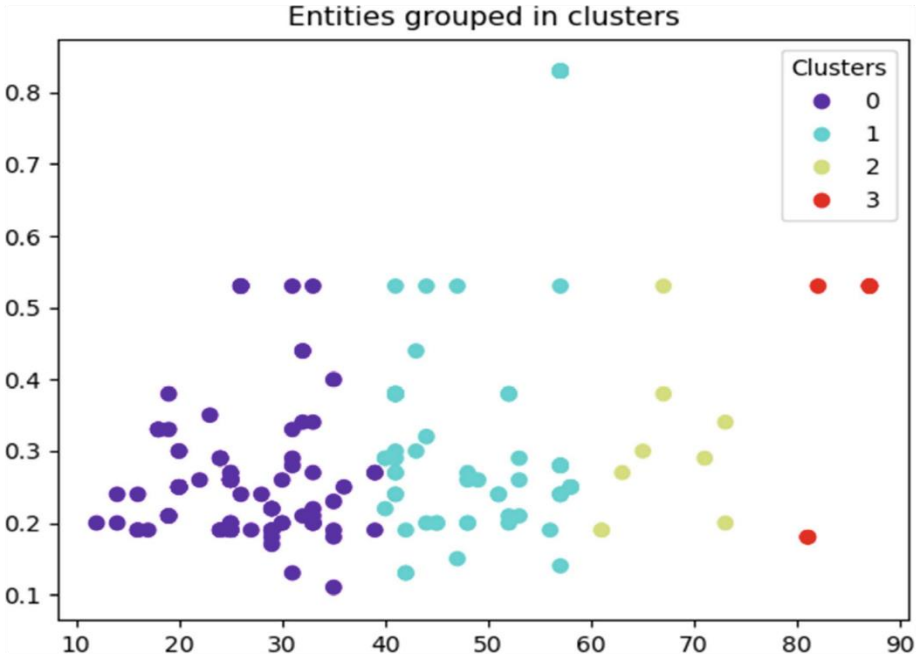

Fig. 2. First case - clustering based on age, name aggregate score, and address score similarity

The clustering of records does not identify the entity but provides a way to label and group records based on the similarity score of attributes. Some of the records in clusters have low scores at the y-axis (address score) but these records are still required for the next stage to make sure not to ignore any matches, related or close matches. After clustering, the segmentation process picked the address with the highest score for the first case and second case as 0.83 and 1.0 respectively.

For the first case, 7 records are retrieved with the highest address score and name aggregate score while 9 other records are retrieved with a similar name, low address score,

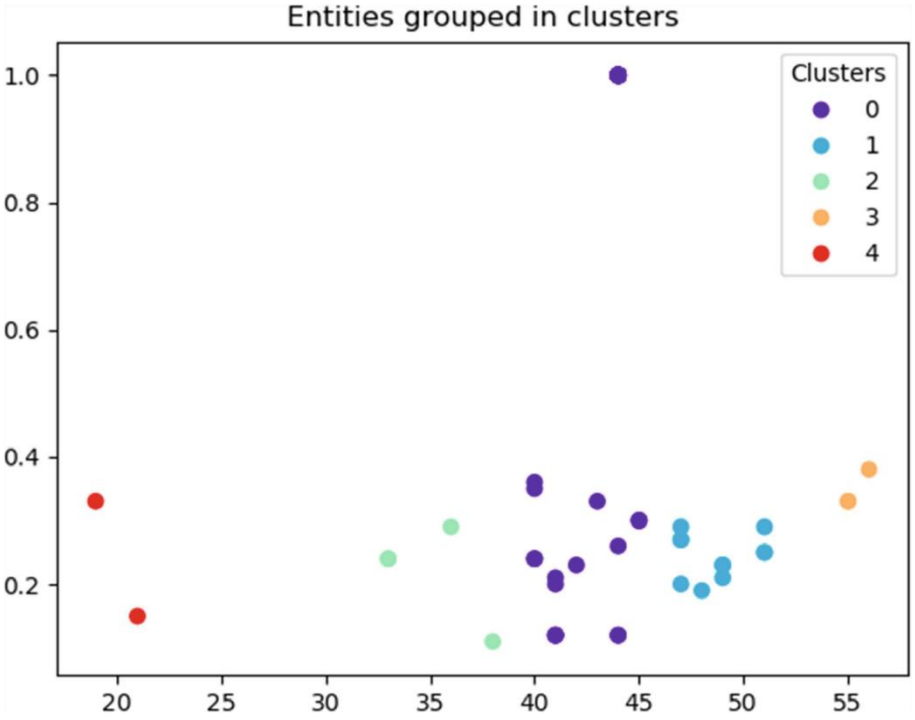

Fig. 3. Second case - clustering based on age, name aggregate score, and address score similarity

and two different dates of birth. So, the final dataset retrieved for the first case contained 16 records with a combination of the same name, two different dates of birth, and different addresses. These records have some similarities but have unique crime reference numbers and duplicate nominal reference numbers for the same name associated with different addresses. For the second case, 16 records are retrieved with the highest address score and name aggregate score while 6 other records are retrieved with a similar name, low address score, and three different dates of birth. So, the final dataset retrieved for the second case contained 22 records with a combination of the same name, three different dates of birth, and different addresses. These records are with some similarities and have a unique crime reference number, duplicate nominal reference numbers associated with different addresses.

For graph analysis, the clustered dataset is used to create the graph with cluster number as main data points associating several different entities linked to each cluster. Another graph is created from a segmented dataset based on the name and address. Both graphs are merged and compared as shown in Fig. 4 for the first case and Fig. 5 for the second case. The suspect is identified out of other entities and is highlighted with Red color in the graph that is associated with different addresses.

5.4 Results Summary

The results discussed show that the fuzzy approach to identity resolution successfully identifies target identity (suspect) out of false identities from a huge dataset. During the evaluation, the results show the reduced number of records retrievals during the initial search and passing through different processes a final dataset was reduced significantly

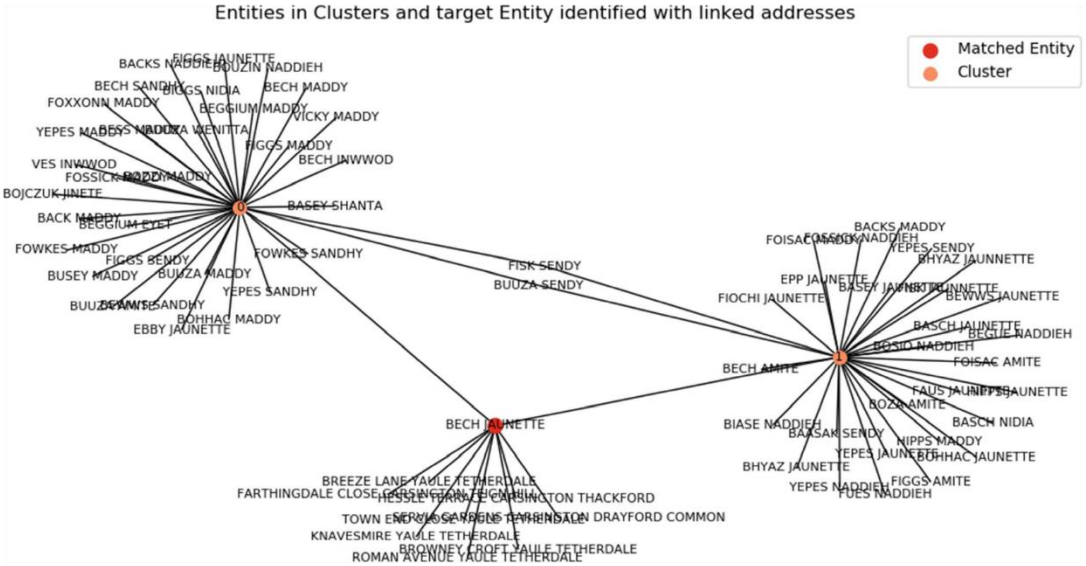

Fig. 4. First case – graph analysis, the suspect identified as "Bech Jaunette" highlighted red and associated to different addresses

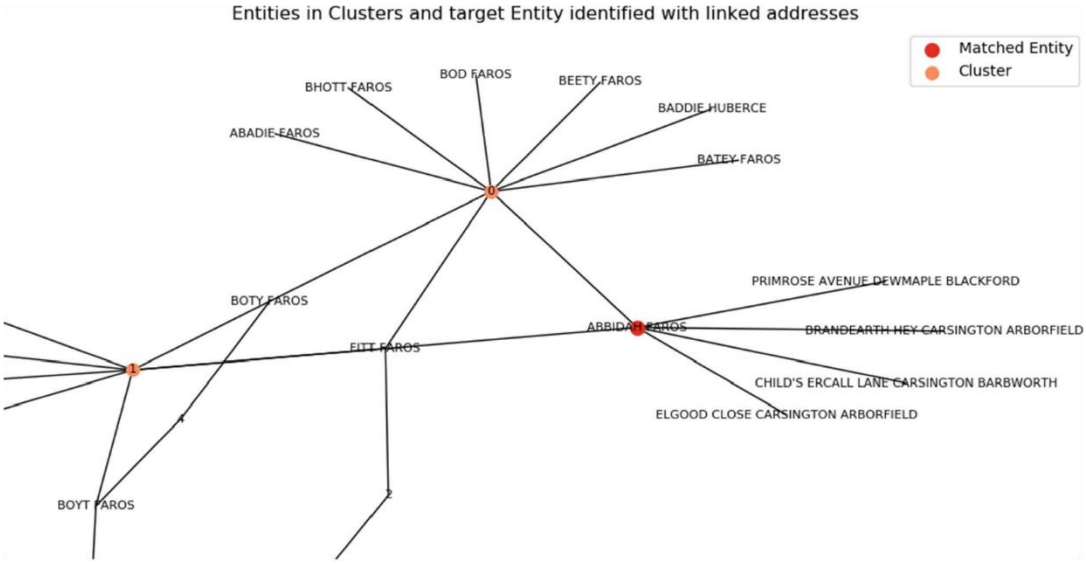

Fig. 5. Second case - graph analysis, the suspect identified as "Abbidah Faros" highlighted red and associated to different addresses

to make the matching process easier. This ensures law enforcement agencies can easily identify a suspect out of false or linked identities and can speed up the investigation process with minimal information in hand.

6 Conclusion

Identity resolution is a very important and crucial task for the law enforcement agency to identify the real identity of a suspect. This research introduced a fuzzy approach

to identity resolution using string similarity techniques with a combination of clustering, segmentation of records, and graph analysis. This research is conducted on an anonymized policing dataset of 1,146,212 records and after data cleaning, a dataset of 715,919 complete records was obtained. The other main feature of this approach is minimal information available for selected attributes e.g. full name, gender, ethnicity description, and part of the address. The similarity algorithms are used in a cascaded manner to calculate the full name aggregate score. The iterative search retrieved records based on the name variation and matching of selected attributes. These records are merged into one dataset and duplicates are removed making the dataset for clustering of records using the Mean-Shift algorithm. Based on the experiment, the search for two suspects created 4 and 5 clusters for records respectively. Later, the segmentation process picked the records with the highest address score to search for similar addresses from the clustered dataset and initial search generated dataset. This process ensured to pick any relevant records that may have missed during the initial search for the suspect's identity. The graph analysis linked all identities on basis of selected attributes and after comparing graphs the suspect identity was identified associated with different addresses. Considering overall results, the fuzzy approach to identity resolution can be very handy for law enforcement agencies to find real identity using minimal information available about any suspect.

In future research, this fuzzy approach can be improved by introducing a weighting system for attribute scores and complete the incomplete records with available information in the dataset or complete records with predicted information for better identity resolution. It will be worth using machine learning techniques to generate a knowledge base that grows with each identity search and simplifies the future search process.

References

1. Wang, G., Chen, H., Atabakhsh, H.: Automatically detecting deceptive criminal identities. Commun. ACM Homel. Secur. **47**(3), 70–76 (2004)
2. Barkay, D., Dror-Rein, E.: Achieving cyber identity resolution via electronic warfare techniques. In: RSA Conference, Singapore (2015)
3. Duncan, J., et al.: Building an ontology for identity resolution in healthcare and public health. Online J. Publ. Health Inform. **7**(2), 1–17 (2015). https://doi.org/10.5210/ojphi.v7i2.6010
4. Roth, D., Ratinov, L.: Who's who in your digital collection: developing a tool for name disambiguation and identity resolution. J. Chicago Colloquium Digital Humanit. Comput. Sci. (DHCS), 1–17 (2009). http://hdl.handle.net/2142/15393
5. Clendenen, C.: A new approach to workers compensation fraud. IAIABC J. Introd. Identity Resolut. **46**(1), 103–114 (2009)
6. Raksha, N., Alankar, R.: Detection of fuzzy duplicates in high dimensional datasets, pp. 1423–1428 (2016)
7. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 39–48 (2003). https://doi.org/10.1145/956755.956759
8. Mon, A.C., Mie, M., Thwin, S.: Effective blocking for combining multiple entity resolution systems. Int. J. Comput. Sci, Eng. **2**(4), 126–136 (2013)
9. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: a survey. IEEE Trans. Knowl. Data Eng. **19**(1), 1–16 (2007). https://doi.org/10.1109/TKDE.2007.250581

10. Gomaa, W., Fahmy, A.: A survey of text similarity approaches. Int. J. Comput. Appl. Found. Comput. Sci. (FCS) **68**(13), 13–18 (2013). https://doi.org/10.5120/11638-7118
11. Brown, D.E., Hagen, S.: Data association methods with applications to law enforcement. Decis. Support Syst. **34**(4), 369–378 (2003)
12. Ananthakrishna, R., Chaudhuri, S., Ganti, V.: Eliminating fuzzy duplicates in data warehouses, Hong Kong, China, pp. 586–597. VLDB Endowment (2002)
13. Pasula, H., Marthi, B., Milch, B., Russell, S., Shpitser, I.: Identity uncertainty and citation matching. Adv. Neural Inform. Proc. Syst., 1425–1432 (2003). https://doi.org/10.1.1.15.8644
14. Culotta, A., McCallum, A.: Joint deduplication of multiple record types in relational data. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 257–258 (2005)
15. Li, J., Wang, A.G.: A framework of identity resolution: evaluating identity attributes and matching algorithms. Secur. Inform. **4**(1), 1–12 (2015). https://doi.org/10.1186/s13388-015-0021-0
16. Bhattacharya, I., Getoor, L.: Entity resolution in graphs. In: Mining Graph Data, p. 311. Wiley-Blackwell, Hoboken (2006)
17. Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. ACM Trans. Knowl. Disc. Data (TKDD) **1**(5), 5–es (2007). https://doi.org/10.1145/1217299.1217304
18. Bartunov, S., Korshunov, A., Park, S., Ryu, W., Lee, H.: Joint link-attribute user identity resolution in online social networks categories and subject descriptors. In: The Sixth SNA-KDD Workshop Proceedings (2012)
19. Phillips, M., Amirhosseini, M.H., Kazemian, H.B.: A rule and graph-based approach for targeted identity resolution on policing data. In: 2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020, pp. 2077–2083. Institute of Electrical and Electronics Engineers Inc. (2020). https://doi.org/10.1109/SSCI47803.2020.9308182

Appendix (B)

I. Python Programming Language

Python is a programming language that is very simple to use and offers excellent features compared to other high-level programming languages. The Python language syntax is straightforward, allowing a programmer to focus on the task rather than the programming syntax complication and making the program development easy. Most importantly, the Python program is much shorter to code than other computer programming languages for many reasons, as a single statement is required to express complex operations. No starting and ending brackets are required for the statements, but they only need indentation to group them. In contrast, the variable declaration is not required (Docs.python.org, 2017).

II. PyCharm IDE

There are many programming editors that a programmer can use to code in Python. However, PyCharm is simple to use and full of features. PyCharm IDE provides many features for developers to use the essential tools for an easy and smooth program development process. It was developed by JetBrains, a software development company known for creating powerful development tools. According to (Jetbrains.com., 2017), there are two versions available to download:

- The **Community Edition** is free and provides many features for Python program development. However, it has limited features available to the programmer because this edition is primarily for academic staff and students.
- The **Professional Edition** is not free and requires a licence to be purchased. It is for a professional programmer and provides a full-featured IDE with robust program development, especially web development, by supporting other frameworks and toolkits on top of those supported in the Community edition.

PyCharm provides a wide range of features to enhance productivity, code quality, and collaboration for Python programmers, making it one of the most popular choices among developers. As an application, it has hardware system requirements for both editions that require at least the following:

- A minimum of 1 GB RAM, but 2 GB RAM is recommended,
- 1024x768 minimum screen resolution,
- Microsoft Windows 10/8/7 (incl.64-bit),
- At least Python 2.4 or higher installed
- The JRE1.8 is required, but it is integrated into the package, so there is no need to install a separate version of Java (Jetbrains.com., 2017)

➲ *Key Features of PyCharm*

- Code Editor - PyCharm offers a feature-rich code editor with syntax highlighting, code completion, formatting, and navigation tools. It supports intelligent code suggestions and auto-completion, which saves time and reduces coding errors.

- Code Inspection and Refactoring - PyCharm performs code inspections and provides suggestions for improving code quality and adhering to Python best practices. It offers various automated refactoring options to improve code maintainability, such as renaming variables, extracting methods, and optimizing imports.

- Debugger - PyCharm includes a powerful debugger that allows developers to step through code, inspect variables and set breakpoints to troubleshoot and fix issues efficiently.

- Testing and Profiling - PyCharm integrates seamlessly with popular testing frameworks like unittest, pytest, and doctest. It provides built-in tools for running tests, viewing test results, and profiling code to identify performance bottlenecks.

- Version Control Integration - PyCharm supports version control systems like Git, Mercurial, and Subversion, allowing developers to manage code repositories directly from the IDE. It provides visual diff and merge tools to simplify code collaboration and merging changes.

- Database Tools - PyCharm offers tools to connect and interact with databases, allowing developers to query and visualize data within the IDE.

- Web Development Support - PyCharm includes support for web development with frameworks like Django, Flask, and Pyramid. It provides code completion, templates, and other features tailored for web development.

- Scientific Tools Integration - PyCharm integrates with tools and libraries like NumPy, SciPy, and Matplotlib, making it suitable for data analysis and scientific computing projects.

- Remote Development - PyCharm allows developers to work on remote projects by connecting to a remote server or a virtual machine.

III. Pandas Python Library

Pandas is an open-source Python data analysis tool that provides fast, flexible and robust data analysis and manipulation features. It is one of Python's most popular data manipulation and analysis libraries. Currently, it provides some of the main features of Python language. It displays data in a tabular format like an SQL table or Excel spreadsheet. It easily handles missing data that is typically displayed as NaN. It allows new columns to be added or deleted from the table called “DataFrame” in Pandas, which provides a powerful “group by” functionality (Pandas.pydata.org., 2017). It is built on top of the NumPy library and also offers additional data structures and functionalities tailored for handling structured data efficiently. Pandas library is widely used in various domains, including data science, machine learning, finance, economics, and social sciences, due to its ease of use, flexibility, and performance.

➲ *Key Features of Pandas*

There are several different features that Pandas offer (McKinney, 2013; VanderPlas, 2017):

- Data Structures - Pandas provides the following two primary data structures.
    - Series: A one-dimensional labelled array containing various data types, including integers, floats, strings, and Python objects. It has an index (labels) and a corresponding array of values, allowing for data alignment.
    - DataFrame: A two-dimensional labelled data structure resembling a table, where data is organised in rows and columns. Each column in a "DataFrame" is a Series. DataFrame provides a flexible and powerful way to work with tabular data.
- Data Manipulation - Pandas offers a rich set of functions and methods for data manipulation tasks, such as filtering, sorting, merging, grouping, reshaping, and aggregating data. Data can be easily sliced, diced, and transformed to meet the requirements of specific data analysis tasks.
- Missing Data Handling - Pandas provides various methods to handle missing data, including dropping missing values or filling them with appropriate values using interpolation or imputation techniques.
- Data Alignment - One of the strengths of Pandas is its ability to automatically align data based on the labels of the data structures. This alignment simplifies performing operations on different datasets with different indices.
- Time Series Functionality - Pandas offers robust support for time series data, making it suitable for working with time-based data, such as financial data, stock prices, and sensor readings. It provides date/time indexing, resampling, time shifting, and frequency conversion functionalities.

- Data I/O - Pandas supports reading and writing data in various formats, such as CSV, Excel, SQL databases, JSON, and more. It allows for seamless data integration and sharing between different data sources.

- Integration with NumPy and Matplotlib - Pandas is built on top of NumPy, enabling seamless integration with NumPy arrays and functions. It also works well with Matplotlib, a popular data visualization library, making insightful plots and charts from Pandas data easy.

Alongside Pandas, the other approximate string-matching packages are used in this framework.

IV. FuzzyWuzzy Python Library

FuzzyWuzzy is a string-matching similarity metric Python library. It offers various fuzzy string-matching algorithms to compare strings and calculate the similarity by using the Levenshtein distance (edit distance) to convert one string into another by calculating the distance. It is open-source and was developed by SeatGeek in 2011. It has been designed to solve the labelling of different events for sports and concert tickets from the internet. Nothing extra software is required as FuzzyWuzzy uses “difflib” from the Python standard library (Bonzanini, 2017). FuzzyWuzzy is particularly useful when dealing with strings with slight variations, typos, or misspellings, as it allows for approximate string matching.

➲ *Key Features of FuzzyWuzzy*

- String Similarity Measurement - FuzzyWuzzy provides several algorithms to calculate the similarity between two strings. The most commonly used algorithm is the Levenshtein distance, which counts the number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another.

- Ratios and Partial Ratios - FuzzyWuzzy calculates similarity ratios, represented as percentages, indicating the similarity between two strings. It offers different ratio types, such as ratio, partial ratio, and token sort ratio, each considering different aspects of string matching.

- Tokenisation and Sorting - FuzzyWuzzy tokenises strings into individual words or tokens before performing comparisons. Tokenisation helps to compare words regardless of their order in the string, and sorting tokens alphabetically improves matching results in some cases.

- Process and Choices - FuzzyWuzzy provides the “process” and “choices” functions, allowing easy comparison of a target string with a list of strings. The “process” function ranks the list of choices based on similarity to the target string (Datacamp.com).

➲ *Usage of FuzzyWuzzy*

- String Matching and Deduplication - FuzzyWuzzy is often used to identify duplicate records in a dataset, especially when dealing with data from different sources with inconsistent or slightly different representations of the same entities.

- Search and Suggestion Systems - FuzzyWuzzy is helpful in search and suggestion systems, where it can be employed to provide more accurate and robust search results, especially when users make typos or misspellings.

V. NetworkX Python Library

NetworkX is an open-source Python library for exploring and analysing networks and network algorithms. The core provides data structures for representing many types of networks, directed graphs, and graphs with self-loops. The nodes in NetworkX graphs can be any (hashable) Python, and edges can contain arbitrary data. Simply put, the nodes represent entities, and the edges represent relationships or connections between

the entities. This kind of flexibility makes NetworkX ideal for networks in many scientific fields. In addition to the primary data structures, many algorithms are implemented to calculate network properties and structure measures, such as shortest paths, betweenness and distribution, and many more. (Schult, 1943). NetworkX is widely used in various domains, including social network analysis, biology, physics, transportation, and computer science, to model and analyze complex systems with interconnected components (Hagberg, Schult and Swart, 2008).

➲ *Key Features of NetworkX*

- Graph Data Structures - NetworkX offers various graph data structures, such as directed graphs (DiGraphs), undirected graphs (Graphs), and multi-graphs (MultiGraphs). The library allows for creating, adding, and removing nodes and edges and attaching attributes to nodes and edges to store additional information (M.E.J.Newman, 2010).

- Graph Algorithms - NetworkX provides a rich set of graph algorithms, such as shortest path finding, centrality measures (e.g., degree centrality, betweenness centrality), clustering coefficient, and community detection (e.g., Louvain method). These algorithms allow for in-depth analysis and insights into the structural properties of networks.

- Graph Visualization - NetworkX integrates with Matplotlib for graph visualization, allowing users to create insightful plots and visualizations of the networks. It provides options to customize node and edge appearance to represent various attributes visually (Hanneman and Riddle, 2005).

- Import and Export - NetworkX supports importing and exporting graphs in various formats, such as GraphML, GML, JSON, and Pajek. It facilitates interoperability with other network analysis tools.

- Extensibility - NetworkX is designed with a modular architecture, allowing users to extend its functionalities by defining custom graph algorithms or graph classes (Hagberg, Schult and Swart, 2008).

➲ *Usage of NetworkX*

- Social Network Analysis (SNA) - NetworkX is commonly used in social network analysis to study relationships between individuals in a social network, measure influence, identify key players, and detect communities.

- Transportation and Infrastructure Networks - NetworkX is employed in modelling transportation systems, such as road, public, and communication networks, to analyze flow patterns and optimize routes.

- Biological and Molecular Networks - In biology, NetworkX is used to study molecular interactions, protein-protein interaction networks, gene regulatory networks, and metabolic pathways.

- Computer Networks - NetworkX is utilized in computer science to analyze computer networks, communication networks, and data networks.