ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IS184853

INTERGRASI DATA BANTUAN PEMERINTAH KOTA TERHADAP PENDUDUK SURABAYA MENGGUNAKAN IDENTITY CORRELATION APPROACH

INTEGRATION OF CITY GOVERNMENT AID DATA FOR SURABAYA RESIDENTS USING IDENTITY CORRELATION APPROACH

ABDUL AZIZUN NAFI
05211540007003

Dosen Pembimbing
1. Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D.
2. Irmasari Hafidz, S.Kom., M.Sc.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR - IS184853

INTEGRASI DATA BANTUAN PEMERINTAH KOTA TERHADAP PENDUDUK SURABAYA MENGGUNAKAN IDENTITY CORRELATION APPROACH

ABDUL AZIZUN NAFI
05211540007003

Dosen Pembimbing
1. Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D.
2. Irmasari Hafidz, S.Kom., M.Sc.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

UNDERGRADUATE THESIS - IS184853

INTEGRATION OF CITY GOVERNMENT AID DATA FOR SURABAYA RESIDENTS USING IDENTITY CORRELATION APPROACH

ABDUL AZIZUN NAFI
05211540007003

Supervisor
1. Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D.
2. Irmasari Hafidz, S.Kom., M.Sc.

INFORMATION SYSTEM DEPARTMENT
Information Technology and Communication Faculty
Sepuluh Nopember Institute of Technology
Surabaya 2019

LEMBAR PENGESAHAN

INTEGRASI DATA BANTUAN PEMERINTAH KOTA TERHADAP PENDUDUK SURABAYA MENGGUNAKAN IDENTITY CORRELATION APPROACH

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu
Syarat Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

ABDUL AZIZUN NAFI
NRP. 0521 15 4000 7003

Surabaya, 15 Juli 2019

KEPALA
DEPARTEMEN SISTEM INFORMASI

Mahendrawathi Er., ST., M.Sc., Ph.D.
NIP 19761011 200604 2 001

LEMBAR PERSETUJUAN

INTEGRASI DATA BANTUAN PEMERINTAH KOTA TERHADAP PENDUDUK SURABAYA MENGGUNAKAN IDENTITY CORRELATION APPROACH

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu
Syarat Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

ABDUL AZIZUN NAFI
NRP. 0521 15 4000 7003

Disetujui Tim Penguji: Tanggal Ujian: 11 Juli 2019
Periode Wisuda: September 2019

Nur Aini Rakhmawati, S.Kom, M.Sc.Eng, Ph.D.	**(Pembimbing I)**
Irmasari Hafidz, S.Kom, M.Sc.	**(Pembimbing II)**
Faizal Johan Atletiko, S.Kom., M.T.	**(Penguji I)**
Radityo Prasetianto Wibowo, S.Kom, M.Kom.	**(Penguji II)**

INTERGRASI DATA BANTUAN PEMERINTAH KOTA TERHADAP PENDUDUK SURABAYA MENGGUNAKAN IDENTITY CORRELATION APPROACH

Nama Mahasiswa : Abdul Azizun Nafi
NRP : 05211540007003
Departemen : Sistem Informasi FTIK-ITS
Pembimbing I : Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D.
Pembimbing II : Irmasari Hafidz, S.Kom., M.Sc.

ABSTRAK

Data dan informasi saat ini memiliki peran yang semakin besar seiring dengan semakin berkembangnya teknologi informasi. Data dan informasi yang memiliki kualitas yang baik dapat menjadi keunggulan kompetitif bagi sebuah organisasi dalam menentukan strategi yang tepat serta dapat menunjang sistem e-government apabila data yang dimiliki dikelola dan dimanfaatkan dengan baik. Pemerintah Kota Surabaya yang merupakan pelopor e-government di Indonesia telah mulai menerapkan sistem e-government sejak tahun 2002 untuk memberikan layanan yang prima kepada masyarakat Kota Surabaya hingga saat ini. Namun, meskipun Pemerintah Kota Surabaya telah membangun sistem berbasis elektronik yang cukup canggih, database SKPD yang dimiliki pemerintah Kota Surabaya memiliki ukuran yang besar dan masih memiliki struktur yang tidak beraturan. Selain itu, database SKPD memiliki isi dataset dari berbagai macam hal yang tidak dikelompokkan dengan baik sehingga sulit untuk mengetahui siapa saja yang menerima bantuan dan bantuan apa saja yang diterima.

Oleh karena itu, dilakukan integrasi database penduduk Kota Surabaya dengan database SKPD terkait bantuan-bantuan yang

diberikan oleh Pemerintah Kota Surabaya menggunakan string matching yang mengadaptasi metode ICA (Identity Correlation Approach). Dataset yang diperlukan dalam penelitian didapatkan dari pihak BAPPEKO Surabaya. Terlebih dahulu dilakukan pembersihan pada data yang telah didapatkan sebelum dilakukan integrasi. Kemudian hasil integrasi data akan divisualisasikan dalam bentuk tabel yang berisi data bantuan penduduk beserta penerimanya serta sebuah histogram yang menunjukkan persebaran data bantuan penduduk kota Surabaya.

Kata Kunci: Integrasi Data, Database Penduduk, Database SKPD, ICA, Bantuan Penduduk

INTEGRATION OF CITY GOVERNMENT AID DATA FOR SURABAYA RESIDENTS USING IDENTITY CORRELATION APPROACH

Name	**: Abdul Azizun Nafi**
NRP	**: 05211540007003**
Department	**: Information System FTIK-ITS**
Supervisor I	**: Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D.**
Supervisor II	**: Irmasari Hafidz, S.Kom., M.Sc.**

ABSTRACT

Data and information now have an increasingly important role along with the development of information technology. Data and information that has good quality can be a competitive advantage for an organization in determining the right strategy and can support the e-government system if the data owned is managed and utilized properly. The Surabaya City Government which is the pioneer of e-government in Indonesia has begun to implement an e-government system since 2002 to provide excellent service to the people of Surabaya City to date. However, even though the Surabaya City Government has developed an electronic-based system that is quite sophisticated, the SKPD database owned by the Surabaya City Government has a large size and still has an irregular structure. In addition, the SKPD database has the contents of datasets of various kinds of things that are not well grouped so that it is difficult to know who received assistance and what assistance was received.

Therefore, the integration of the Surabaya City population database with the SKPD database related to the aid provided by the Surabaya City Government uses string matching that adapts the ICA (Identity Relationship Approach) method. The dataset needed in the study was obtained from BAPPEKO Surabaya.

First cleaning is done on the data that has been obtained before integration. Then the results of data integration will be visualized in the form of a table that contains data on the aid of residents and their recipients as well as a histogram that shows the distribution of aid data for Surabaya city residents.

Keywords: Data Integration, Residents Database, SKPD Database, ICA, Residents Aids

KATA PENGANTAR

Segala puji dan syukur pada Allah SWT. yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan buku yang sederhana ini dengan judul Intergrasi Data Bantuan Pemerintah Kota Terhadap Penduduk Surabaya Menggunakan *Identity Correlation Approach*. Dalam penyelesaian Tugas Akhir ini, penulis diiringi oleh pihak-pihak yang selalu memberi dukungan, saran, dan doa sehingga penelitian berlangsung dengan lancar. Secara khusus penulis mengucapkan terima kasih dari lubuk hati terdalam kepada:

1. Allah SWT. yang selalu menemani dan membimbing penulis dalam segala aspek kehidupan.
2. Seluruh keluarga besar khususnya kedua orang tua penulis yang tiada hentinya mendoakan dan memberikan dukungan kepada penulis.
3. Bapak Galuh selaku kepala BAPPEKO Surabaya yang telah memfasilitasi penelitian ini.
4. Ibu Mahendrawathi Er., ST., M.Sc., Ph.D. selaku Ketua Departemen Sistem Informasi ITS Surabaya.
5. Bapak Bekti Cahyo Hidayanto, S.Si., M.Kom. selaku dosen wali penulis yang selalu membimbing dan memberikan arahan kepada penulis.
6. Ibu Nur Aini Rakhmawati, S.Kom., M.Sc. Eng., Ph.D. dan Ibu Irmasari Hafidz, S.Kom., M.Sc. selaku dosen pembimbing yang telah mencurahkan segenap tenaga, waktu dan pikiran dalam penelitian ini, serta memberikan motivasi yang membangun.
7. Bapak Faizal Johan Atletiko, S.Kom., M.T. dan Bapak Radityo Prasetianto Wibowo, S.Kom, M.Kom. selaku dosen penguji yang telah memberikan kritik dan saran yang membuat kualitas penelitian ini lebih baik lagi.

8. Segenap dosen dan karyawan Departemen Sistem Informasi.
9. Keluarga besar CSSMoRA ITS sebagai keluarga pertama penulis di ITS.
10. Teman-teman Sistem Informasi angkatan 2015 (LANNISTER) yang menemani dan memberikan motivasi bagi penulis selama perkuliahan hingga dapat menyelesaikan tugas akhir.
11. Kakak dan adik angkatan 2013, 2014, 2016 dan 2017 yang membantu dan memberikan semangat bagi penulis.
12. Pihak lainnya yang berkontribusi dalam tugas akhir yang belum dapat penulis sebutkan satu per satu.

Penyusunan tugas akhir ini masih jauh dari kata sempurna, untuk itu penulis menerima segala kritik dan saran yang membangun sebagai upaya menjadi lebih baik lagi ke depannya. Semoga buku tugas akhir ini dapat memberikan manfaat untuk pembaca.

Surabaya, Juli 2019

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR KODE

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan mengenai gambaran umum dari tugas akhir yang dikerjakan meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan relevansi terhadap pengerjaan tugas akhir. Pada bab ini juga ditunjukkan tujuan dan manfaat dari pengerjaan tugas akhir sehingga diharapkan dapat memberikan gambaran awal mengenai tugas akhir yang dikerjakan.

1.1 Latar Belakang

Data dan informasi saat ini memiliki peran yang semakin besar seiring dengan semakin berkembangnya teknologi informasi. Data dan informasi yang memiliki kualitas yang baik dapat menjadi keunggulan kompetitif bagi sebuah organisasi dalam menentukan strategi yang tepat agar dapat memberikan produk atau layanan yang tidak kalah saing dengan pesaing-pesaing lainnya [1]. Meningkatnya peran data dan informasi dipengaruhi oleh beberapa hal seperti munculnya teknologi *big data*, *semantic web*, *crowd technology* dan sebagainya. Oleh sebab itu, perlu dilakukan manajemen data dan informasi yang baik agar dapat memanfaatkan data dan informasi yang dimiki dengan efektif dan efisien.

Pemerintah Kota Surabaya yang merupakan pelopor *e-government* di Indonesia telah mulai menerapkan sistem *e-government* sejak tahun 2002 [2]. Pemerintah Kota Surabaya memanfaatkan teknologi informasi untuk mengolah data dan informasi yang mereka miliki untuk memberikan layanan yang prima kepada masyarakat Kota Surabaya. Salah satu organisasi perangkat yang ikut membantu dalam berjalannya sistem *e-*

government di Kota Surabaya adalah Badan Perencanaan Pembangunan kota (BAPPEKO) Surabaya. BAPPEKO Surabaya bertanggung jawab dalam melaksanakan tugas dan mengkoordinasikan penyusunan, pengendalian, dan evaluasi pelaksanaan rencana pembangunan daerah di kota Surabaya. Sebagai koordinator penyusunan perencanaan pembangunan, BAPPEKO melakukan penyusunan tahapan-tahapan kegiatan yang bertujuan untuk meningkatkan kesejahteraan sosial di kota Surabaya dimana salah satunya dilakukan dalam bentuk menyediakan produk perencanaan pembangunan serta informasi spasial kepada publik melalui peta [3].

Namun, meskipun Pemerintah Kota Surabaya telah membangun sistem berbasis elektronik yang cukup canggih, sistem tersebut belum terintegrasi sepenuhnya sehingga terdapat kesenjangan dalam pemanfaatan teknologi informasi yang dimiliki. Salah satu contohnya adalah belum terintegrasinya *database* kependudukan Kota Surabaya dengan *database* SKPD terkait bantuan-bantuan yang diberikan kepada penduduk Kota Surabaya. *Database* SKPD yang dimiliki pemerintah Kota Surabaya memiliki ukuran yang besar namun memiliki struktur yang tidak beraturan. Selain itu, *database* SKPD memiliki isi *dataset* dari berbagai macam hal yang tidak dikelompokkan dengan baik sehingga sulit untuk mengetahui siapa saja yang menerima bantuan dan bantuan apa saja yang diterima. Oleh karena itu, dilakukan sebuah penelitian yang bertujuan untuk mengintegrasikan *database* kependudukan Kota Surabaya dengan *database* SKPD terkait bantuan-bantuan yang diberikan oleh Pemerintah Kota Surabaya dimana integrasi dilakukan dengan menggunakan *string matching* yang mengadaptasi metode ICA (*Identity Correlation Approach*).

1.2 Rumusan Masalah

Berdasarkan permasalahan yang diuriakan pada bagian latar belakang, maka perumusan masalah yang menjadi fokus dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara mengintegrasikan *database* kependudukan Kota Surabaya dengan *database* SKPD terkait bantuan-bantuan yang diterima penduduk Kota Surabaya dengan menggunakan metode ICA?
2. Bagaimana memvisualisasi hasil integrasi *database* kependudukan Kota Surabaya dengan *database* SKPD terkait bantuan-bantuan yang diterima penduduk Kota Surabaya?

1.3 Batasan Permasalahan

Berdasarkan permasalahan-permasalahan yang dihadapi, terdapat beberapa batasan dalam pengerjaan tugas akhir. Batasan-batasan yang ditetapkan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Data yang digunakan dari *database* SKPD Kota Surabaya hanya data yang berkaitan dengan *database* kependudukan dan *database* BDT Kota Surabaya tahun 2018.
2. Hasil akhir dari pengerjaan tugas akhir ini adalah visualisasi hasil integrasi *database* kependudukan dan *database* BDT Kota Surabaya dengan *database* SKPD terkait bantuan-bantuan yang diterima penduduk Kota Surabaya yang telah terintegrasi.

1.4 Tujuan

Berdasarkan uraian pada bagian perumusan masalah dan batasan masalah yang telah dijelaskan sebelumnya, tujuan yang ingin dicapai dalam pengerjaan tugas akhir ini adalah mengintegrasikan *database* kependudukan Kota Surabaya dengan *database* SKPD terkait bantuan-bantuan yang diterima penduduk Kota Surabaya. Sehingga didapatkan data yang valid mengenai siapa saja yang menerima bantuan dari pemerintah Kota Surabaya dan bantuan apa saja yang diterima oleh setiap penduduk Kota Surabaya.

1.5 Manfaat

Manfaat yang diharapkan dapat diperoleh dari pengerjaan tugas akhir ini adalah dapat memfasilitasi pemerintah Kota Surabaya untuk mengetahui siapa saja yang menerima bantuan dari pemerintah Kota Surabaya dan bantuan apa saja yang diterima oleh setiap penduduk Kota Surabaya. Selain itu juga akan diperlihatkan persebaran bantuan-bantuan yang diberikan oleh pemerintah Kota Surabaya kepada penduduk Kota Surabaya berdasarkan kecamatan melalui grafik *bar chart*.

1.6 Relevansi

Tugas akhir ini berkaitan dengan integrasi *database* kependudukan dan *database* BDT Kota Surabaya dengan *database* SKPD terkait bantuan-bantuan yang diterima penduduk Kota Surabaya dengan menggunakan *string matching* dimana hasil akhir dari tugas akhir ini adalah tabel dan visualisasi mengenai siapa saja yang menerima bantuan dari pemerintah Kota Surabaya dan bantuan apa saja yang diterima oleh setiap penduduk Kota Surabaya, sehingga tugas akhir ini sesuai dengan bidang keilmuan laboratorium Akuisisi Data dan Diseminasi Informasi. Tugas akhir ini juga layak dijadikan sebagai tugas akhir pada tingkat S1 dimana tugas ini merupakan

salah satu pemecahan masalah yang dihadapi oleh pemerintah Kota Surabaya dalam mengintegrasikan *database* kependudukan dan *database* BDT Kota Surabaya dengan *database* SKPD sehingga dapat mengetahui data-data mengenai penduduk yang menerima bantuan dari pemerintah Kota Surabaya dan bantuan apa saja yang diterima.

Selain itu, tugas akhir ini juga memiliki relevansi dengan mata kuliah Sistem Informasi seperti Pemrograman Integratif, Pengantar Basis Data, Pengolahan Bahasa Alami, dan Visualisasi Informasi sehingga layak untuk dijadikan sebagai tugas akhir departemen Sistem Informasi.

Halaman ini sengaja dikosongkan.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dijelaskan mengenai penelitian-penelitian sebelumnya yang berkaitan dengan tugas akhir dan membahas dasar teori yang perlu dipahami sebagai landasan dalam pengerjaan tugas akhir ini.

2.1 Penelitian Sebelumnya

2.1.1 A Mathematical Solution to String Matching for Big Data Linking

Penelitian ini berjudul "*A Mathematical Solution to String Matching for Big Data Linking*" yang ditulis oleh Kevin McCormack dan Mary Smith [4]. Penelitian ini merupakan lanjutan dan penyempurnaan dari penelitian sebelumnya dari penulis yang sama yang berjudul "*Big Data Matching Using the Identity Correlation Approach*" [5] dimana peneliti melakukan integrasi *dataset* dari sensus 2011 dengan *Public Sector Administrative Datasets* (ADS) di Irlandia dengan menggunakan metode *Identity Correlation Approach* (ICA). ICA merupakan bagian dari dikembangkannya proyek *big data matching* yaitu SESADP (*Structure of Earnings Survey Administrative Data Project*) oleh Central Statistics Office, Ireland. ICA bekerja dengan mengkombinasikan beberapa *variable* dari satu individu (untuk setiap baris data) yang menghasilkan *identifier* baru yang unik, misalkan pada satu baris data terdapat *variable date of birth*, *gender*, *marital status* dan sebagainya yang kemudian beberapa *variable* tersebut digabungkan sehingga menjadi satu *identifier* baru yang unik. Tugas akhir ini akan mengadaptasi metode *Identity Correlation Approach* (ICA) untuk melakukan integrasi data kependudukan dengan data intervensi kota Surabaya.

2.1.2 Single and Multiple Pattern String Matching Algorithm

Penelitian ini berjudul ”*Single and Multiple Pattern String Matching Algorithm*“ yang diajukan oleh Chinta Someswara Rao dan K. Butchi Raju [6]. Pada penelitian ini dijelaskan mengenai perbandingan beberapa metode *string matching* seperti algoritma Commentz-Walter (CW), algoritma Wu-Manber (WM), algoritma Aho-Corasick (AC), dan algoritma Rabin-Karp yang dibandingkan dengan metode algoritma *single* dan *multiple pattern string matching* yang diajukan oleh penulis dimana *dataset* yang digunakan dalam perbandingan ini adalah data mengenai DNA *Sequence* dari beberapa jenis monyet. Metodologi yang digunakan terbagi menjadi 4 fase yaitu fase *input*, *divide*, *process*, dan *output* dimana dari hasil penilitian yang telah dilakukan didapati bahwa metode algoritma *single* dan *multiple pattern string matching* yang diajukan oleh penulis menghasilkan *search time* yang lebih cepat dibandingkan dengan algoritma-algoritma *string matching* yang sudah ada.

2.1.3 Integrasi Data dan Visualisasi Graf pada Aplikasi Halal Nutrition Food

Penelitian ini ditulis oleh Berlian Fajar Yusuf [7], dimana pada penelitian ini dibahas mengenai bagaimana cara mengintegrasikan data *World Open Food* Fact ke *Halal Nutrition Food*, cara mengelompokkan produk makanan pada *Halal Nutrition Food* menggunakan *Graph Partitioning*, dan cara memvisualisasikannya. Data diintegrasikan dengan melakukan indeks menggunakan *Apache Lucene* dimana pengindeksan dilakukan dengan menjalankan kode program yang dihasilkan oleh Ahmad Choirun Najib pada penelitian yang telah dilakukan olehnya [8]. Sebelum dilakukan integrasi data, dilakukan *data cleansing* pada data yang telah diakuisisi sebelumnya dan telah diamati skema RDF pada data tersebut.

Data cleansing dilakukan untuk membersihkan data kotor dan mengubah bahasa data bahan makanan yang masih berbahasa perancis ke bahasa inggris.

2.2 Dasar Teori

2.2.1 E-Government

Electronic Government merupakan pengaplikasian teknologi informasi yang berbasis internet dan perangkat lainnya yang dikelola oleh pemerintah dalam penyampaian informasi dari pemerintah kepada masyarakat, mitra, dan lembaga-lembaga lain secara *online* [9]. Di Indonesia, sistem *e-government* sudah diinisiasi dan didukung oleh Instruksi Presiden Republik Indonesia Nomor 3 Tahun 2003 tentang Kebijakan dan Strategi Nasional Pengembangan *e-government* dan didukung pula oleh Undang-Undang No. 14 tahun 2008 tentang Keterbukaan Informasi Publik, serta Peraturan Pemerintah No. 61 tahun 2010 tentang Implementasi Undang-Undang Keterbukaan Informasi Publik [10]. Pemerintah Kota Surabaya yang merupakan pelopor *e-government* di Indonesia telah mulai menerapkan sistem *e-government* sejak tahun 2002 [2]. Pemerintah Kota Surabaya memanfaatkan teknologi informasi untuk mengolah data dan informasi yang mereka miliki untuk memberikan layaanan yang prima kepada masyarakat Kota Surabaya.

2.2.2 Data Matching

Data matching adalah suatu metode yang mengidentifikasi, mencocokkan, dan menggabungkan *records* yang sesuai dengan entitas yang sama dari beberapa *database* atau dalam satu *database*. Pada dasarnya, terdapat 5 langkah utama dalam melakukan *data matching* yaitu [11]:

1. Data Pre-processing
   Langkah ini bertujuan untuk memastikan bahwa data yang akan diperiksa memiliki struktur dan format yang sama. Pada dasarnya pada tahap ini dilakukan *data cleaning* dan standarisasi pada data-data yang akan dibandingkan. Pada data *pre-processing* terdapat beberapa hal yang harus dilakukan diantaranya:
   a. Menghapus karakter dan kata yang tidak diperlukan
   b. Memperluas singkatan dan memperbaiki ejaan yang salah
   c. Melakukan segmentasi atribut menjadi atribut *output* yang *well-defined* dan konsisten
   d. Melakukan verifikasi kebenaran dari nilai atribut
2. Record Pair Comparison
   Setelah dilakukan *indexing*, dilakukan perbandingan lebih mendalam untuk mengetahui tingkat kesamaan dari data-data yang dibandingkan. Pada umumnya, kesamaan antara data-data yang dibandingkan dilihat dari pebandingan antara atribut-atribut dari data tersebut. Namun akan lebih baik bila yang diperhatikan bukan hanya atribut-atribut yang digunakan dalam langkah *indexing* sebelumnya namun juga atribut lain yang ada pada *database* tersebut yang sama.
3. Classification
   Pada tahap ini dilakukan klasifikasi data-data yang telah dibandingkan pada tahap sebelumnya berdasarkan *comparison vector* atau *summed similarities* dari data-data tersebut. Klasifikasi bisa dilakukan dengan menggunakan *two-class* / *binary* (*matches* atau *non-matches*) atau *three-class* (*matches*, *non-matches* dan *potential matches*).

4. Evaluation of Matching Quality and Complexity
   Evaluasi dilakukan setelah data-data yang dibandingkan diklasifikasi sebagai *matches*, *non-matches*, atau *potential matches* jika menggunakan *three-class*. *Matching quality* dilakukan untuk melihat seberapa banyak hasil klasifikasi *matches* sesuai dengan apa yang diharapkan (*real-world entities*). Sedangkan *matching completeness* dilakukan untuk melihat seberapa banyak *real-world entities* yang muncul pada kedua *database* yang dibandingkan yang hasil klasifikasinya benar-benar sesuai.

2.2.3 String Matching

String matching merupakan pencarian sebuah *pattern* pada sebuah teks [T. H. Cormen, 1994]. *String matching* digunakan untuk menemukan suatu *string* yang berperan sebagai *pattern* dalam *string* yang terdapat dalam sebuah teks atau file.

Prinsip kerja algoritma *string matching* [D. Effendi, 2013] adalah sebagai berikut:

1. Memindai teks dengan bantuan sebuah *window* yang ukurannya sama dengan panjang *pattern*.
2. Menempatkan *window* pada awal teks.
3. Membandingkan karakter pada *window* dengan karakter dari *pattern*. Setelah pencocokan (baik hasilnya cocok atau tidak cocok) dilakukan pergeseran ke kanan pada *window*. Prosedur ini dilakukan berulang-ulang sampai *window* berada pada akhir teks. Mekanisme ini disebut mekanisme *sliding window*.

Menurut Effendi, algoritma *string matching* mempunyai tiga komponen utama, yaitu:

1. *Pattern*, yaitu deretan karakter yang akan dicocokkan dengan teks.
2. Teks, yaitu tempat pencocokan *pattern* dilakukan.
3. Alfabet, berisi semua simbol yang digunakan oleh bahasa pada teks dan *pattern*.

Cara yang jelas untuk mencari *pattern* yang cocok dengan teks adalah dengan mencoba mencari di setiap posisi awal dari teks dan mengabaikan pencarian secepat mungkin jika karakter yang salah ditemukan [D. E. Knuth, 1977]. Proses pertama adalah dengan menyelaraskan bagian paling kiri dari *pattern* dengan teks. Kemudian dibandingkan karakter yang sesuai dari teks dan *pattern*. Setelah seluruhnya cocok ataupun tidak cocok dari *pattern*, *window* digeser ke kanan.

[R. Singh dan H. N. Verma, 2011] mengungkapkan bahwa efisiensi dari algoritma terletak pada dua tahap, yaitu:

1. Tahap praproses, tahap ini mengumpulkan informasi penuh tentang *pattern* dan menggunakan informasi ini pada tahap pencarian.
2. Tahap pencarian, *pattern* dibandingkan dengan *window* dari kanan ke kiri atau kiri ke kanan sampai kecocokan atau ketidakcocokan terjadi.

Dalam algoritma pencocokan *string*, teks diasumsikan berada di dalam memori, sehingga bila *string* dicari dalam sebuah *database*, maka semua isi *database* perlu dibaca terlebih dahulu kemudian disimpan di dalam memori. Oleh karena itu, perlu dilakukan pengaturan yang benar mengenai penggunaan memori jika *database* yang perlu dibaca berukuran besar.

2.2.4 Identity Correlation Approach

Identity Correlation Approach (ICA) merupakan bagian dari dikembangkannya proyek *big data matching* yaitu SESADP

(*Structure of Earnings Survey Administrative Data Project*) oleh Central Statistics Office, Ireland [5]. ICA bekerja dengan mengkombinasikan beberapa *variable* dari satu individu (untuk setiap baris data) yang menghasilkan *identifier* baru yang unik, misalkan pada satu baris data terdapat *variable date of birth*, *gender*, *marital status* dan sebagainya yang kemudian beberapa *variable* tersebut digabungkan sehingga menjadi satu *identifier* baru yang unik. Contoh pembuatan *identifier* dengan menggunakan metode ICA ditunjukkan pada tabel 2.1 berikut.

Date of Birth	Sex	County	Marital Status	Matchvar
15031949	M	CORK	M	15031949MCORKM
11021945	F	LIMERICK	S	11021945FLIMERICKS
21111954	M	DUBLIN	D	21111954MDUBLIND
19051964	M	CARLOW	O	19051964MCARLOWO
22091966	M	GALWAY	M	22091966MGALWAYM

Tabel 2.1 Contoh Pembuatan Matchvar

Dalam melakukan *matching* pada *dataset* yang diinginkan, digunakan 1 – n *matchvar identifier* dimana akan dihapus satu *variable* tertentu disetiap *matchvar* selanjutnya sehingga pada setiap *record* akan memiliki sebanyak n *identifier* yang unik. Kemudian *matchvar* 1 sampai *matchvar* n tersebut digunakan untuk melakukan *matching* pada kedua dataset yang ingin dibandingkan.

Unique Identifier (UI)	Variables constituting UI
Matchvar 1	(CBRno|DoB|sex|NACE2|PP|County|MS)
Matchvar 2	(CBRno|DoB|sex|NACE2|PP|County)
Matchvar 3	(CBRno|DoB|sex|NACE2|PP|MS)
Matchvar 4	(CBRno|DoB|sex|NACE2|PP)
Matchvar n	(DoB|sex|NACE2|PP|County|MS)

Tabel 2.2 Contoh Pembuatan Matchvar Identifier

Setelah *matchvar 1* sampai *matchvar n* terbentuk, kemudian dilakukan *matching* pada kedua *dataset* yang ingin dibandingkan menggunakan *matchvar 1*. Kemudian, jika masih terdapat *dataset* yang masih berstatus *unmatched*, akan diperiksa menggunakan *matchvar 2* begitu seterusnya sampai seluruh *dataset* telah berstatus *matched* atau telah mencapai *matchvar n*.

Probabilitas dari *matching records* dari dua *datasets* dapat dihitung menggunakan formula MRUI (*Matching Rate of Unique Identifier*) apabila *records* tersebar merata di seluruh *class* yang ada. Formula MRUI ditunjukkan pada tabel 2.3 berikut [4].

$$MRUI = N * \frac{1}{V1_{ui}} * \frac{1}{V2_{ui}} * \frac{1}{V3_{ui}} * \frac{1}{V4_{ui}} \ \ldots * \frac{1}{Vx_{ui}}$$

Where:

N = Population

V = Variable

x = No. of variables ui = Uniqueness Factor, no. of classes in the variable (if records are distributed evenly across all classes)

Tabel 2.3 Formula MRUI

Apabila MRUI = 1, maka hal ini menunjukkan bahwa terdapat *unique identifier variable* untuk setiap *record* sehingga bisa dilakukan *matching* secara langsung ke *records* yang ada. MRUI < 1 menunjukkan bahwa terdapat *unique identifier variable* untuk setiap *record* dan terdapat *variable* lain yang masih bisa dibuat untuk meningkatkan hasil *matching* pada *records* yang ada. Sedangkan MRUI > 1 menunjukkan bahwa tidak ada *unique identifier variable* sehingga akan ada duplikat yang muncul saat proses *matching* dilakukan.

Misalkan terdapat *dataset* penduduk Irlandia dimana terdapat 65.000 penduduk yang lahir pada tahun yang sama, dengan rincian ditunjukkan pada tabel 2.5 berikut.

Variable Name	Symbol	No. Classes	Description of Classes
DoB	$V1_{ui}$	365	ui = 365 (days of the year)
Gender	$V2_{ui}$	2	2 genders
NACE	$V3_{ui}$	10	10 different NACE
Marital Status	$V4_{ui}$	2	2 marital status codes
County	$V5_{ui}$	5	5 counties

Tabel 2.4 Rincian Contoh Dataset

Dengan menggunakan formula MRUI, permasalahan tersebut dapat diselesaikan seperti pada tabel 2.6 berikut.

$$MRUI = 65.000 * \frac{1}{365} * \frac{1}{2} * \frac{1}{10} * \frac{1}{2} * \frac{1}{5} = \frac{65.000}{73.000} = 0.89$$

Tabel 2.5 Penyelesaian Contoh Menggunakan MRUI

Dari hasil penyelesaian diatas menunjukkan bahwa MRUI dihasilkan adalah kurang dari 1, sehingga nilai tersebut menunjukkan bahwa terdapat *unique identifier variable* untuk setiap *record* dan juga terdapat *unique identifier variable* lain yang masih bisa dibuat untuk meningkatkan hasil *matching* pada *records* yang ada.

2.2.5 Visualisasi Informasi

Visualisasi informasi mengacu pada representasi grafis informasi yang dihasilkan oleh komputer. Visualisasi informasi lebih berfokus terhadap bagaimana desain, pengembangan, dan aplikasi dari representasi informasi grafik interaktif yang dihasilkan oleh komputer. Tujuan dari visualisasi informasi sendiri adalah untuk menyampaikan ide yang kompleks yang dapat menginspirasi penggunanya untuk menemukan koneksi baru dari informasi tersebut. Visualisasi informasi memiliki karakteristik yang unik seperti *visual thinking*, fokus terhadap gambaran besar, dan untuk menggali wawasan yang lebih mendalam dari data yang tersedia untuk dijadikan sebagai informasi yang lebih bermanfaat [12]. Proses visualisasi informasi dapat dibagi menjadi 6 langkah yaitu [13]:

1. *Mapping*, yaitu bagaimana cara memproses informasi menjadi bentuk visual dengan cara mengubah data atau informasi yang dimiliki menjadi bentuk grafik dengan fitur visual yang sesuai.
2. *Selection*, yaitu proses memilih data yang sesuai dari banyaknya data yang tersedia untuk diubah menjadi grafik visual. Langkah ini merupakan langkah yang

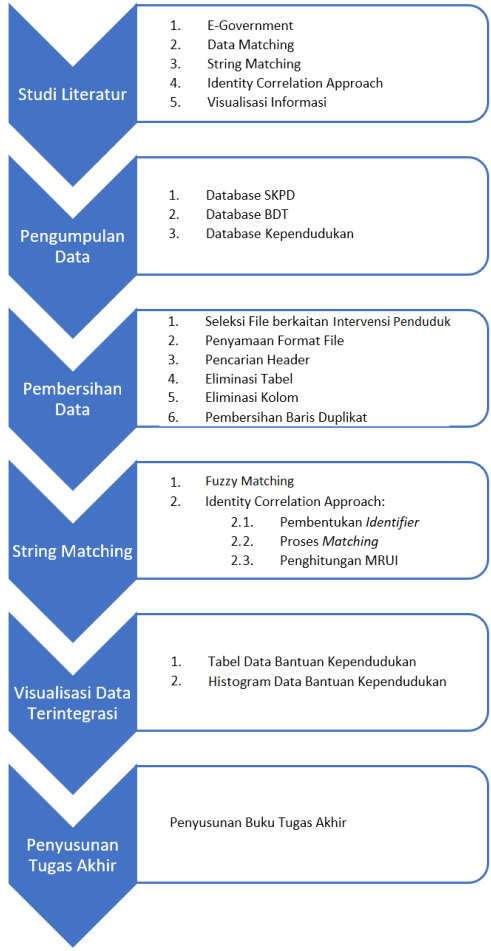
paling penting dikarenakan kesalahan memilih data yang digunakan dapat menyebabkan kesalahan pemahaman oleh pengguna sehingga pengguna bisa mengalami kesalahan dalam pengambilan keputusan penting.

3. *Presentation*, dimana dalam hal ini yang dimaksud adalah bagaimana cara mengelola dan mengorganisir informasi yang ingin disampaikan seefektif mungkin.
4. *Interactivity*, merupakan fasilitas-fasilitas yang digunakan untuk mengatur dan eksplorasi visualisasi yang dilakukan.
5. *Human Factor*, mencakup dua faktor yaitu kegunaan dan aksesibilitas, dimana visualisasi informasi harus mudah digunakan dan mudah diakses meskipun oleh orang yang berkebutuhan khusus.
6. *Evaluation*, yang dilakukan untuk mengetahui seberapa efektif metode visualisasi yang digunakan dan apakah tujuan dari visualisasi informasi sudah tercapai.

Halaman ini sengaja dikosongkan.

BAB III
METODOLOGI

Pada bab metodologi akan dijelaskan mengenai langkah-langkah yang dilakukan dalam pengerjaan tugas akhir dengan penjelasan mengenai langkah-langkah tersebut. Langkah-langkah tersebut digambarkan dengan gambar 3.1 berikut.



Gambar 3.1 Tahapan Pengerjaan Tugas Akhir

3.1 Studi Literatur

Pada tahapan ini dilakukan penggalian dan pemahaman mengenai informasi yang berhubungan dengan pengerjaan tugas akhir. Informasi yang dibutuhkan dicari melalui jurnal ilmiah, buku, dan media lainnya. Informasi-informasi yang dicari meliputi:

1. E-Government
2. Data Matching
3. String Matching
4. Identity Correlation Approach
5. Visualisasi Informasi

3.2 Pengumpulan Data

Data yang digunakan dalam pengerjaan tugas akhir ini merupakan data sekunder yang didapat dari BAPPEKO Surabaya. Data yang dibutuhkan yaitu:

1. Contoh *database* SKPD yang ditangani oleh BAPPEKO (selanjutnya disebut "*database* SKPD")
2. Contoh *database* BDT yang ditangani oleh BAPPEKO (selanjutnya disebut "*database* BDT")
3. Contoh *database* penduduk Kota Surabaya yang digunakan untuk integrasi data (selanjutnya disebut "*database* penduduk")

Data-data tersebut diperoleh dari sebuah tim yang sebelumnya ingin mengajukan proposal program pengabdian pada masyarakat yang dipimpin oleh Ibu Nur Aini Rakhmawati, P.hd dimana tim tersebut telah mendapatkan data tersebut dari BAPPEKO Surabaya secara formal.

3.3 Pembersihan Data

Pada tahap ini dilakukan pembersihan data pada *database* SKPD yang telah didapatkan sebelumnya. Tahapan ini meliputi:

3.3.1 Seleksi File Berkaitan dengan Intervensi Penduduk

Langkah pertama yang dilakukan pada tahap pembersihan data adalah melakukan seleksi *file* dari *database* SKPD. Langkah ini bertujuan untuk mengambil *file* SKPD yang berkaitan dengan data intervensi penduduk Surabaya dimana *file* ini nantinya akan digunakan untuk proses selanjutnya. Sedangkan *file* SKPD yang tidak berkaitan dengan intervensi penduduk tidak akan digunakan.

3.3.2 Penyamaan Format File

Pada langkah ini dilakukan penyamaan format *file* dari *database* SKPD yang telah diseleksi, *database* BDT dan *database* utama agar *file* menjadi lebih ringan sehingga menjadi lebih mudah diproses. Hasil dari langkah ini adalah tabel-tabel SKPD memiliki format *file* yang sama. Proses penyamaan format *file* dilakukan dengan menggunakan shell. Contoh dapat dilihat pada tabel 3.1.

Contoh File	
Sebelum	Sesudah
DataSby.csv	DataSby.csv
DataBDT.csv	DataBDT.csv
SKPD_2017_1.xls	SKPD_2017_1.csv
SKPD_2017_2.xlsx	SKPD_2017_2.csv

Tabel 3.1 Contoh Penyamaan Format File

3.3.3 Pencarian Header

Pencarian *header* dilakukan karena seringkali ditemukan peletakan *header* yang tidak berada pada baris pertama pada *database* yang dapat mengakibatkan terhambatnya proses identifikasi data. Proses pencarian *header* dilakukan dengan menggunakan Jupyter.

3.3.4 Eliminasi Tabel

Pada langkah ini dilakukan eliminasi tabel pada *database* SKPD yang tidak diperlukan. Eliminasi dilakukan dengan cara mencari nama kolom pada *header* yang ada pada *database* SKPD. Bila pada tabel tersebut tidak ditemukan *value* yang diperlukan maka tabel tersebut akan dibuang. Pada langkah ini, tabel-tabel pada *database* SKPD yang tidak memiliki kolom NIK tidak akan digunakan dan akan dihapus. Langkah ini dilakukan dengan menggunakan Jupyter. Contoh langkah ini ditunjukkan pada tabel 3.2.

Contoh Tabel	Kolom NIK	Eliminasi Tabel
SKPD_2017_1.csv	Ada	Tidak
SKPD_2017_2.csv	Ada	Tidak
SKPD_2017_3.csv	Tidak Ada	Ya
SKPD_2017_4.csv	Ada	Tidak

Tabel 3.2 Contoh Proses Eliminasi Tabel

3.3.5 Eliminasi Kolom

Eliminasi kolom dilakukan dengan cara membuang kolom-kolom yang tidak diperlukan dan hanya menyisakan kolom-kolom yang sesuai dengan *value* yang diperlukan. Selain itu, kolom yang tidak memilki *value* sama sekali juga akan dihapus. Eliminasi ini dilakukan dengan menggunakan Jupyter. Contoh eliminasi kolom ditunjukkan pada tabel 3.3.

Tabel	Kolom	
	Sebelum	Sesudah
SKPD_2017_1.csv	Nama	Nama
	NIK	NIK
	No. KK	Usia
	No, Kartu Kuning	Alamat
	Usia	
	Alamat	

Tabel 3.3 Contoh Proses Eliminasi Kolom

3.3.6 Pembersihan Baris Duplikat

Pembersihan baris duplikat dilakukan dengan memeriksa dan memastikan bahwa tidak ada baris yang memiliki nilai yang sama persis (nilai duplikat). Langkah ini dilakukan dengan menggunakan Jupyter. Contoh pembersihan baris duplikat dapat dilihat pada tabel 3.4.

Kolom	*Value*	
	Sebelum	Sesudah
NIK	3578162502970004	3578162502970004
	3578171504950004	3578171504950004
	3578171504950004	3578031308940002
	3578031308940002	3578022004940003
	3578022004940003	

Tabel 3.4 Contoh Pembersihan Baris Duplikat

3.4 String Matching

Pada tahap ini dilakukan pembandingan antara *value* dari *database* SKPD, *database* BDT, dan *database* penduduk Surabaya. Namun, sebelum dilakukan intergrasi antara *database* SKPD dan *database* BDT, dilakukan pembersihan terhadap *database* BDT terlebih dahulu dikarenakan terdapat *records* dari *database* BDT yang masih kotor seperti terdapat kesalahan penulisan nama, NIK yang salah, dan sebagainya. Proses *string matching* ini nantinya akan dilakukan dengan menggunakan *shell script* AWK.

3.4.1 Fuzzy Matching

Pembersihan *database* BDT dilakukan dengan menggunakan metode *fuzzy matching* antara *database* BDT dengan *database* penduduk Surabaya agar *records* antar kedua *database* terserbut terintegrasi. Seperti yang telah disebutkan sebelumnya, proses ini diperlukan untuk membersihkan *database* BDT dikarenakan terdapat *records* dari *database* BDT yang masih kotor seperti terdapat kesalahan penulisan nama, NIK yang salah, *value* yang masih kosong dan sebagainya. Kemudian *database* BDT bersih akan digunakan untuk membentuk *unique identifier* pada proses selanjutnya.

3.4.2 Identity Correlation Approach

Tahap ini dilakukan dengan mengadaptasi algoritma *Identity Correlation Approach* (ICA). ICA bekerja dengan mengkombinasikan beberapa *variable* dari satu individu (untuk setiap baris data) yang menghasilkan *identifier* baru yang unik [5] yang kemudian akan digunakan untuk proses *matching*. Tujuan dari tahap ini adalah untuk mengintegrasikan *database* SKPD dengan *database* BDT dan *database* penduduk.

1. Pembentukan *Identifier*
   Identifier unik yang disebut *matchvar* akan dibentuk berdasarkan pada *dataset* BDT yang telah dibersihkan. *Identifier* akan digunakan untuk melakukan *matching* antara *database* BDT dengan *database* SKPD dimana nantinya akan dibentuk 10 kombinasi *matchvar* (*matchvar* 1 - 10) agar memiliki peluang keberhasilan lebih tinggi dalam melakukan proses *matching*. Dikarenakan ukuran *database* SKPD yang besar dan terdapat banyak *dataset* yang tidak memiliki hubungan dengan *database* kependudukan, maka *dataset* yang akan diambil dari *database* SKPD hanya *dataset* yang memiliki kolom NIK saja, yang kemudian dipilih *dataset* yang berkaitan dengan intervensi penduduk Surabaya.
2. Proses *Matching*
   Matching dilakukan pada *database* SKPD dan *database* BDT menggunakan *matchvar* 1. Kemudian apabila masih terdapat *records* yang masih berstatus *unmatched*, akan diperiksa menggunakan *matchvar* 2 begitu seterusnya sampai seluruh dataset telah berstatus *matched* atau telah mencapai *matchvar* 10.
3. Penghitungan MRUI
   Proses penghitungan MRUI (*Matching Rate for Unique Identifier*) dilakukan pada *database* hasil *matching* yang telah dilakukan sebelumnya. Apabila hasil penghitungan MRUI telah mendekati angka 1, maka dapat dikatakan efektivitas *matchvar* yang dibentuk adalah tinggi dan begitu pula dengan tingkat probabilitas kecocokan *dataset* hasil *matching.*

3.5 Visualisasi Data Terintegrasi

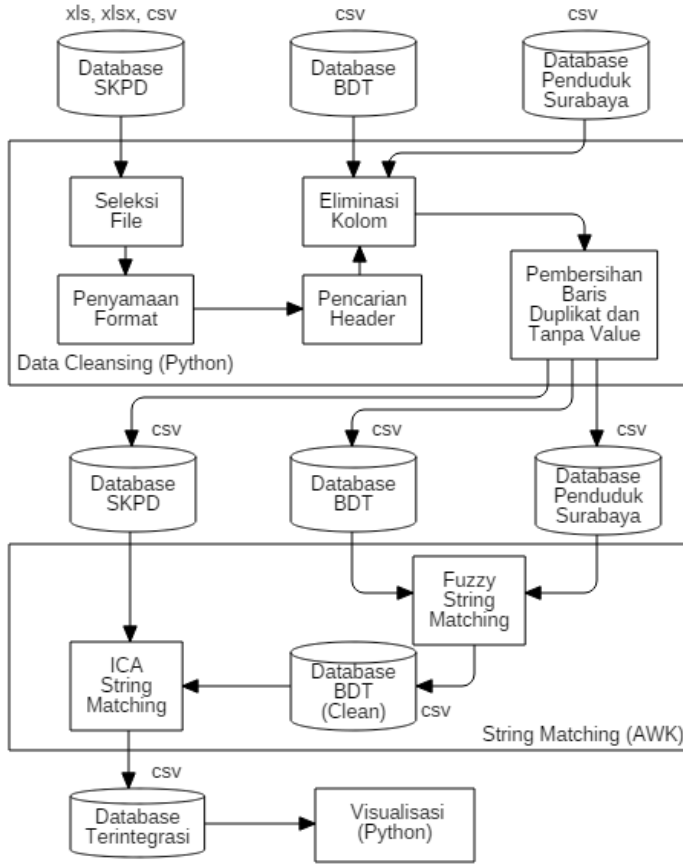
Setelah *database* SKPD, *database* BDT dan *database* utama berhasil terintegrasi melalui tahap sebelumnya, selanjutnya dilakukan pembuatan visualisasi yang dapat menunjukkan hasil data terintegrasi. Visualisasi akan ditunjukkan dalam bentuk tabel sehingga nantinya dapat ditunjukkan *value* yang dihasilkan dari data terintegrasi seperti data penduduk Surabaya dan bantuan-bantuan apa saja yang diterima dari Pemerintah Kota Surabaya. Selain itu, visualisasi juga dilakukan dalam bentuk *histogram* yang menunjukkan banyaknya penduduk yang menerima bantuan dari pemerintah Kota Surabaya setiap kecamatan sehingga dapat terlihat kecamatan mana yang mendapatkan bantuan paling sedikit dan sebaliknya.

3.6 Penyusunan Buku Tugas Akhir

Penyusunan buku tugas akhir dilakukan dengan melakukan dokumentasi pada seluruh aktivitas-aktivitas yang dijalankan selama proses pengerjaan tugas akhir. Dokumentasi dilakukan mulai dari permasalahan yang diangkat, metodologi pengerjaan tugas akhir yang digunakan, eksekusi metodologi, serta kesimpulan dan saran dari pengerjaan tugas akhir ini hingga rujukan studi pustaka yang digunakan. Hasil akhir dari tahap ini adalah buku tugas akhir.

3.7 Arsitektur Sistem

Arsitektur sistem yang digunakan dalam pengerjaan Tugas Akhir ini dapat dilihat pada gambar 3.2 berikut.



Gambar 3.2 Arsitektur Sistem

Database SKPD akan melalui proses *data cleansing* mulai dari seleksi *file*, penyamaan format *file*, pencarian *header*, eliminasi kolom, dan pembersihan baris duplikat. Sedangkan untuk *database* BDT dan *database* penduduk Surabaya hanya akan melalui proses eliminasi kolom dan pembersihan baris duplikat saja.

Setelah melalui proses *data cleansing*, *database* BDT dan *database* penduduk Surabaya akan melalui proses *fuzzy*

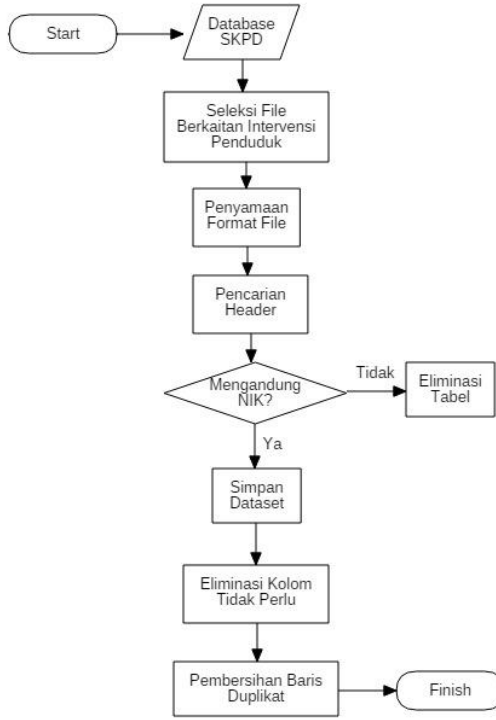
matching untuk membersihkan data yang ada pada *database* BDT. Kemudian dilakukan proses *string matching (ICA)* antara *database* BDT yang telah bersih dengan *database* SKPD untuk menghasilkan *database* yang terintegrasi. *Database* terintegrasi kemudian digunakan dalam proses visualisasi.

BAB IV
PERANCANGAN

Pada bab ini akan dijelaskan mengenai rancangan penelitian tugas akhir yang dimulai dari rancangan proses pembersihan data, *string matching* hingga visualisasi data terintegrasi.

4.1 Pembersihan Data

Pembersihan data dilakukan pada *database* SKPD dan *database* BDT dengan menggunakan python, sedangkan untuk *database* MBR tidak perlu dilakukan pembersihan data karena data tersebut sudah bersih. Pada gambar 4.1 berikut ditunjukkan *flow chart* proses pembersihan pada *database* SKPD.

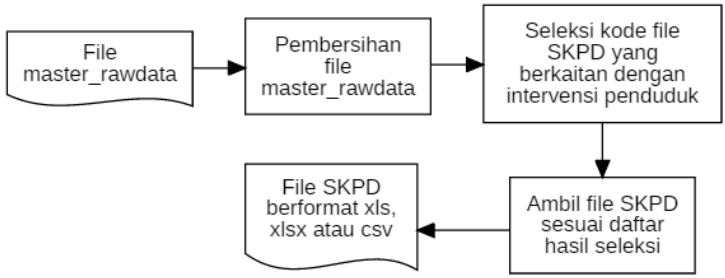


Gambar 4.1 Tahapan Pembersihan Database SKPD

Untuk *database* BDT hanya akan dilakukan eliminasi kolom yang tidak diperlukan serta pemeriksaan dan pembersihan baris duplikat.

4.1.1 Seleksi File Berkaitan dengan Intervensi Penduduk

Seleksi *file* dilakukan pada *database* SKPD untuk memisahkan *file* SKPD yang berkaitan dengan data intervensi penduduk Surabaya dan *file* yang tidak berkaitan. Proses seleksi dilakukan berdasarkan *file* master_rawdata.csv yang didapatkan dari BAPPEKO Surabaya dimana *file* tersebut berisi tentang daftar nama kode dan nama *file* SKPD. Dari *file* tersebut diambil kolom yang diperlukan yaitu kolom id_rawdata yang berisi kode *file* SKPD dan kolom narasi_rawdata yang berisi tentang penjelasan nama *file* SKPD dari setiap kode *file*. Kemudian dilakukan pembersihan pada *record*s dari kolom narasi_rawdata dan dilakukan proses seleksi pada *records* tersebut untuk menghapus nama *file* SKPD yang tidak berkaitan dengan data intervensi penduduk. Kemudian daftar nama hasil seleksi akan digunakan untuk mengambil *file* yang berkaitan dengan data intervensi penduduk dari *database* SKPD.

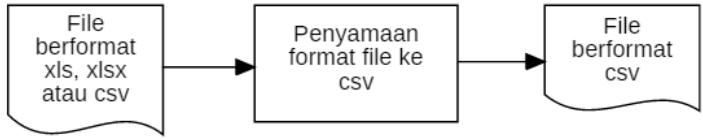


Gambar 4.2 Alur Seleksi File SKPD

4.1.2 Penyamaan Format File

Penyamaan format *file* dilakukan pada *database* SKPD karena terdapat banyak *file* yang memiliki ekstensi berbeda seperti xls,

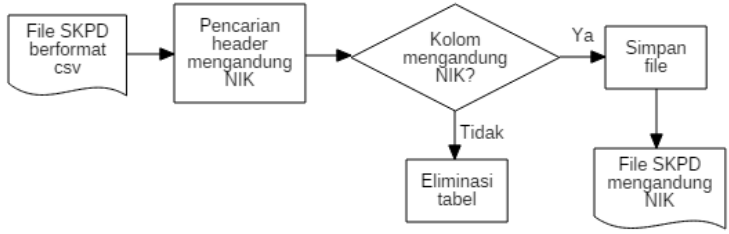
xlsx dan csv sehingga nantinya seluruh *file database* SKPD memiliki ekstensi yang sama yaitu csv. Contoh dapat dilihat pada tabel 3.1.



Gambar 4.3 Alur Penyamaan Format File

4.1.3 Pencarian Header dan Eliminasi Tabel

Pencarian *header* dilakukan pada *database* SKPD untuk memastikan bahwa *header* dari setiap *file* SKPD berada pada baris pertama. Kemudian dilakukan pencarian pada setiap *header* apakah terdapat kolom NIK atau tidak, dimana jika pada tabel tersebut tidak terdapat kolom NIK maka tabel tersebut akan dieliminasi. Contoh dapat dilihat pada tabel 3.2.

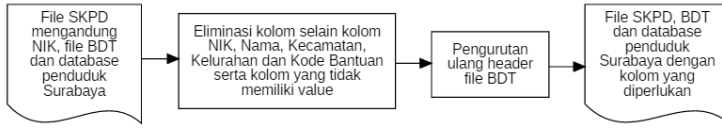


Gambar 4.4 Alur Pencarian Header dan Eliminasi Tabel

4.1.4 Eliminasi Kolom

Eliminasi kolom dilakukan pada *database* SKPD, *database* BDT dan *database* penduduk Kota Surabaya dimana kolom-kolom yang dirasa tidak diperlukan dan kolom yang tidak memiliki *value* sama sekali akan dieliminasi. Pada *database* SKPD, *database* BDT dan *database* penduduk Kota Surabaya nanti, kolom selain kolom NIK, Nama, Kecamatan dan Kelurahan akan dihapus. Kemudian dilakukan pengurutan

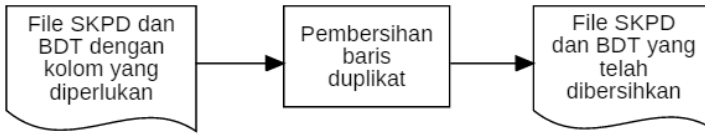
ulang pada *header database* BDT dengan urutan NIK, Nama, Kecamatan dan Kelurahan agar nantinya memudahkan dalam proses *matching*. Contoh dari proses ini dapat dilihat pada tabel 3.3.



Gambar 4.5 Alur Eliminasi Kolom Tidak Diperlukan

4.1.5 Pembersihan Baris Duplikat

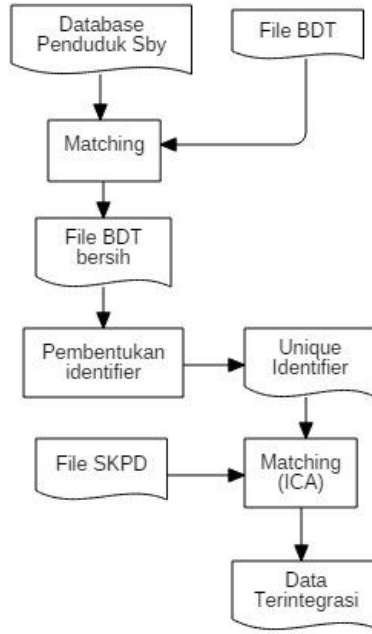
Pembersihan baris duplikat dilakukan pada *database* SKPD dan *database* BDT dengan memeriksa apakah terdapat *record* yang memiliki *value* yang sama. Jika terdapat *record* dengan *value* yang sama (duplikat) maka salah satu *record* akan dihapus. Contoh dapat dilihat pada tabel 3.4.



Gambar 4.6 Alur Pembersihan Baris Duplikat

4.2 String Matching

Proses *string matching* akan dilakukan sesuai dengan alur pada gambar 4.7 berikut.



Gambar 4.7 Alur Proses Matching File

Pertama akan dilakukan *fuzzy matcing* antara *database* BDT dengan *database* penduduk Surabaya sehingga menghasilkan *database* BDT bersih. Kemudian *database* BDT bersih akan digunakan untuk membentuk *identifier* dimana *identifier* nantinya akan digunakan dalam proses *matching* dengan *database* SKPD menggunakan metode ICA. Hasil dari *matching* ICA ini adalah *database* terintegrasi yang berisi NIK, nama, kecamatan, kelurahan, dan bantuan yang diterima.

4.2.1 Fuzzy Matching

Pembersihan *database* BDT dilakukan dengan menggunakan metode *fuzzy matching* antara *database* BDT dengan *database* penduduk Surabaya agar *records* antar kedua *database* terserbut terintegrasi. Proses ini bertujuan untuk membersihkan *database* BDT dikarenakan terdapat *records* dari *database* BDT yang masih kotor seperti terdapat kesalahan penulisan nama, NIK yang salah, *value* yang masih kosong dan sebagainya. Pada proses ini, pertama akan dicari *records* dengan NIK dan kelurahan yang sama persis, jika ada akan diambil. Kemudian akan dicari *records* dengan nama dan kelurahan yang sama persis, jika ada akan diambil kembali. Lalu sisanya akan dilakukan perbandingan dengan *fuzzy matching* antara nama dan kelurahan pada setiap *records* di kedua *database*, dimana apabila kedua *variable* tersebut memiliki kesamaan lebih dari 80% maka akan dianggap sebagai *instance* yang sama. NIK tidak dilibatkan dalam proses *fuzzy matching* karena pada *database* BDT terdapat *records* dengan NIK kosong atau tidak valid.

4.2.2 Identity Correlation Approach

Pengerjaan tugas akhir ini mengadaptasi metode *Identity Correlation Approach* dengan membentuk *identifier* dari *database* BDT yang telah dibersihkan. *Identifier* nantinya akan digunakan untuk proses *matching* dengan *database* SKPD yang berisi banyak *file* dengan struktur kolom yang berbeda-beda. Dibentuk *unique identifier* seperti pada tabel 4.1.

Unique Identifier (UI)	Variables constituting UI
Matchvar 1	Nama | NIK | Kec | Kel
Matchvar 2	Nama | NIK | Kec

Unique Identifier (UI)	Variables constituting UI
Matchvar 3	Nama | NIK
Matchvar 4	Nama | NIK | Kel | Kec
Matchvar 5	Nama | NIK | Kel
Matchvar 6	NIK | Kec | Kel
Matchvar 7	Kec | Kel | NIK | Nama
Matchvar 8	Kel | NIK | Nama
Matchvar 9	Kec | NIK | Nama
Matchvar 10	NIK | Nama

Tabel 4.1 Unique Identifier

Unique identifier yang telah dibentuk digunakan untuk proses *matching* pada *file* SKPD untuk mengintegrasikan data penduduk penerima bantuan yang terdaftar pada *file* SKPD dan *file* BDT. Pada tahap ini juga dibentuk kolom baru yaitu kolom id_skpd. Kolom ini berfungsi untuk mengetahui bantuan apa yang diterima oleh seseorang berdasarkan *id* nama *file* SKPD dimana orang tersebut terdapat didalamnya.

4.3 Visualisasi Data Terintegrasi

Visualisasi akan dilakukan menggunakan data yang dihasilkan dari proses *matching* yang telah dilakukan pada tahap sebelumnya.

4.3.1 Tabel Data Bantuan Penduduk

Tabel data bantuan pemerintah Kota untuk penduduk Surabaya akan divisualisasikan dari *database* berisi data penduduk Surabaya dan bantuan yang didapatkan dari pemerintah kota yang telah diintegrasikan di tahap sebelumnya. Dilakukan *inner join* pada *database* terintegrasi dengan tabel master_rawdata

yang berisi daftar *id* dan nama *file* SKPD agar dapat diketahui bantuan apa yang diterima oleh penduduk didalam *database* tersebut. Tabel ini nantinya akan berisi nama orang yang mendapatkan bantuan beserta NIK, kecamatan dan kelurahan orang tersebut serta bantuan apa yang diterima olehnya.

4.3.2 Histogram Bantuan Penduduk

Visualisasi ini berbentuk *bar chart* yang dibentuk berdasarkan *dataset* berisi data penduduk Surabaya dan bantuan yang didapatkan dari pemerintah kota yang dihasilkan dari proses sebelumnya. Tujuan dari visualisasi ini adalah untuk mengetahui persebaran penduduk penerima bantuan berdasarkan kecamatan sehingga hasil dari visualisasi ini adalah *bar chart* jumlah bantuan untuk penduduk Surabaya berdasarkan kecamatan.

BAB V
IMPLEMENTASI

Bab ini menjelaskan tentang proses implementasi penelitian tugas akhir berdasarkan perancangan yang telah dibahas pada bab sebelumnya

5.1 Lingkungan Penelitian

Penelitian tugas akhir ini dilakukan dengan perangkat pendukung yang dapat dilihat pada tabel 5.1 dan tabel 5.2.

Perangkat Keras	
Perangkat	Spesifikasi
Jenis	ASUS X454Y
CPU	AMD Quad Core A8-7410
RAM	8 GB
DISK	500 GB HDD + 250 GB SSD

Tabel 5.1 Spesifikasi Perangkat Keras

Perangkat Lunak	
Perangkat	Kegunaan
Windows 10 Pro 64-bit	Sistem Operasi
Jupyter	Python IDE, visualisasi
Notepad ++	Text editor
Windows Subsystem for Linux	Menjalankan Ubuntu di Windows OS
Ubuntu 18.04 Terminal	AWK IDE

Tabel 5.2 Spesifikasi Perangkat Lunak

5.2 Pembersihan Data

Pembersihan data dilakukan pada *database* SKPD dan *database* BDT dengan menggunakan python. Beberapa *library* tambahan dari python yang digunakan dalam pengerjaan tugas akhir ini dapat dilihat pada tabel 5.3 berikut. Sedangkan *library* lain yang yang tidak disebutkan pada tabel 5.3 merupakan *library* bawaan milik python.

No.	Library	Kegunaan
1.	Pandas 0.24.2	Mengolah *dataframe*
2.	NLTK 3.4.3	Mengolahan dan tokenisasi kata
3.	Numpy 1.16.4	Mengolah *dataframe*
4.	Plotly 3.10.0	Visualisasi

Tabel 5.3 Library Tambahan untuk Python

5.2.1 Seleksi File Berkaitan dengan Intervensi Penduduk

Seleksi *file* dilakukan pada *database* SKPD untuk memisahkan *file* SKPD yang berkaitan dengan data intervensi penduduk Surabaya dan *file* yang tidak berkaitan. Proses seleksi dilakukan berdasarkan *file* master_rawdata.csv yang didapatkan dari BAPPEKO Surabaya dimana *file* tersebut berisi tentang daftar nama kode dan nama *file* SKPD.

id_rawdata	id_data_master	narasi_rawdata	data_angka	tahun_data
3814	232	Jumlah bahan pangan segar yang disampling dan aman tahun 2017	n	2017
3815	232	Jumlah bahan pangan segar yang disampling tahun 2017	n	2017
3816	233	Skor Pola Pangan Harapan tahun 2017	n	2017
3817	232	Angka Kecukupan Gizi tahun 2017	n	2017
3818	232	Jumlah dan data rincian sampel ketersediaan, kualitas konsumsi, dan keamanan pangan yan	n	2017
3819	57	jumlah intervensi ketersediaan komoditas tahun 2017	n	2017
3820	57	jumlah kejadian gejolak harga tahun 2017	n	2017
3821	231	jumlah pasar yang dipantau harganya tahun 2017	n	2017
3822	56	jumlah pasar tradisional yang dikelola tahun 2017	n	2017
3823	231	Tingkat stabilitas harga pangan tahun 2017	n	2017
3824	231	Jumlah jenis komoditas tahun 2017	n	2017
3825	220	jumlah kesempatan kerja yang dapat diinformasikan tahun 2017	n	2017
3826	220	jumlah kesempatan kerja yang dapat diinformasikan tahun 2016	y	2016
3827	219	Jumlah peserta sertifikasi keseluruhan tahun 2017	n	2017
3828	219	Jumlah peserta sertifikasi yang lulus tahun 2017	n	2017
3829	219	Jumlah peserta pelatihan yang lulus pelatihan tahun 2017	n	2017
3830	219	jumlah peserta yang mengikuti pelatihan tahun 2017	n	2017
3831	219	Jumlah pencari kerja yang terdaftar tahun 2017	n	2017

Gambar 5.1 Cuplikan File Master_rawdata.csv

Dari *file* tersebut diambil kolom yang diperlukan yaitu kolom id_rawdata yang berisi kode *file* SKPD dan kolom narasi_rawdata yang berisi tentang penjelasan nama *file* SKPD dari setiap kode *file*. Kemudian dilakukan pembersihan pada *record*s dari kolom narasi_rawdata dan dilakukan proses seleksi pada *records* tersebut untuk menghapus nama *file* SKPD yang tidak berkaitan dengan data intervensi penduduk.

Proses seleksi dilakukan dengan menggunakan daftar *stop words* dimana apabila dalam *records* terdapat baris yang mengandung *stop words* maka *record* tersebut akan dihapus. *Stop words* dibentuk dari berbagai kata yang tidak berkaitan dengan intervensi / bantuan penduduk Surabaya.

No.	Stop Words
1.	psks
2.	instrumen
3.	pejabat
4.	pns
5.	disiplin
6.	angkutan
7.	lembaga
8.	persimpangan

Tabel 5.4 Cuplikan Stop Words

```
import pandas as pd
import re
import string
import nltk
from nltk.tokenize import word_tokenize

mdata = pd.read_csv('master_rawdata.csv', index_col=None)
to_drop =
['id_data_master','data_angka','tahun_data','kolom_ke','jenis_realisasi','ketera
ngan_lanjutan']
mdata.drop(columns=to_drop, inplace=True)
new_df = mdata
new_df.rename(columns={'id_rawdata':'id_skpdfile','narasi_rawdata':'narasi'},
inplace=True)
user_defined_stop_words =
'psks|instrumen|tanah|pns|pejabat|disiplin|kecelakaan|angkutan|penumpang|
daya
tampung|lembaga|prestasi|kurikulum|kondisi|dibangun|sampling|tpm|rumah
sakit|buku|event|kota|genangan|persimpangan|jalan|ketertiban|keamanan|ut
ilitas|pdam|pelanggaran|kejadian|kb|usia|lahir|kematian|puskesmas|bayi|bers
alin|balita|hamil|nifas|odtw|bumd|indikator|kecamatan|kelurahan|pos|sp|kajia
n|hukum|layanan|sampah|komunikasi|rencana|lahan|berita|rupabumi|produk
|capaian|mou|survey|survei|sarana|volume|sentra|sistem|luas|bencana|pju|ad
ministrasi|direncanakan|arsip|akreditasi|unit|bidang|koperasi|perpustakaan|h
ewan|nilai|energi|obat|pajak|dokumen|wisata|realisasi|total|del|temuan|kelo
mpok|izin|laboratorium|bahan|kendaraan|pemerintahan|skor|angka|sampel|
hotel|kader|kegiatan|jenis|gudang|aset|pasar|harga|komoditas|diinformasika
n|perusahaan|bangunan'

i = nltk.corpus.stopwords.words('indonesian')
j = list(string.punctuation)
stopwords = set(i).union(j)
def preprocess(x):
    x = re.sub('[^0-9a-z\s]', '', x.lower())
    x = [w for w in x.split() if w not in set(stopwords)]
    return ' '.join(x)
new_df['narasi'] = new_df['narasi'].apply(preprocess)
cdata = new_df[['id_skpdfile','narasi']]
cdata1 =
cdata[cdata["narasi"].str.contains(user_defined_stop_words)==False]
cdata1.to_csv(r'daftarfileSKPDintervensi.csv', encoding='utf-8', index=False)
```

Kode 5.1 Pembersihan dan Seleksi Records File master_rawdata.csv

Kemudian daftar nama hasil seleksi akan digunakan untuk mengambil *file* yang berkaitan dengan data intervensi penduduk dari *database* SKPD.

```
import os
import re
import functools
import shutil

Hasil_1 = r'Hasil_1'
def validate_file(validators, file_path):
    return any(re.search(validator, file_path) for validator in validators)
def get_matching_files_in_dir(directory, validator, append_dir=True):
    for file_path in os.listdir(directory):
        if validator(file_path):
            yield os.path.join(directory, file_path) if append_dir else file_path
matching_patterns = cdata2['id_skpdfile'].astype(str).values.tolist()
validator = functools.partial(validate_file, matching_patterns)
skpdbantuan = list(get_matching_files_in_dir(r'SKPD', validator))
co = 0
for file in skpdbantuan:
    shutil.copy(file, Hasil_1)
    co += 1
    print(file)
print(co)
```

Kode 5.2 Ambil File BDT Berkaitan dengan Intervensi Penduduk

5.2.2 Penyamaan Format File

Setelah proses seleksi *file* SKPD yang berkaitan dengan intervensi penduduk berhasil dilakukan, proses selanjutnya adalah penyamaan format *file* pada *file* SKPD yang terpilih. Penyamaan format *file* dilakukan pada *database* SKPD karena terdapat banyak *file* yang memiliki ekstensi berbeda seperti xls, xlsx dan csv sehingga nantinya seluruh *file database* SKPD memiliki ekstensi yang sama yaitu csv.

```
import glob
import pandas as pd

for excel in glob.glob(r'Hasil_1\*.xlsx'):
    out = excel.split('.')[0]+'.csv'
    data_xls = pd.read_excel(excel, index=False)
    data_xls.to_csv(out, encoding='utf-8', index=False)
for excel in glob.glob(r'Hasil_1\*.xls'):
    out = excel.split('.')[0]+'.csv'
    data_xls = pd.read_excel(excel, index=False)
    data_xls.to_csv(out, encoding='utf-8', index=False)
```

Kode 5.3 Penyamaan Format File ke CSV

5.2.3 Pencarian Header dan Eliminasi Tabel

Pencarian *header* dilakukan pada *database* SKPD untuk memastikan bahwa *header* dari setiap *file* SKPD berada pada baris pertama. Kemudian dilakukan pencarian pada setiap *header* apakah terdapat kolom NIK atau tidak, dimana jika pada tabel tersebut tidak terdapat kolom NIK maka tabel tersebut akan dieliminasi.

```
import glob
import pandas as pd
import shutil

Hasil_2 = r'Hasil_2'
count = 0
for excel in glob.glob(r'Hasil_1\*.csv'):
    df0 = pd.read_csv(excel.split('.')[0]+'.csv', engine='python',
encoding='utf-8')
    df0.columns = map(lambda x: str(x).upper(), df0.columns)
    df1 = df0.apply(lambda x: x.str.upper() if x.dtype == "object" else x)
    if 'NIK' in df1:
        print (excel.split('.')[0]+'.csv'+' ada')
        df1.to_csv(excel.split('.')[0]+'_nik.csv', encoding='utf-8', index=False)
        shutil.copy(excel.split('.')[0]+'_nik.csv', Hasil_2)
        count += 1
    else:
        with open(excel.split('.')[0]+'.csv', encoding="utf8") as fd:
            index = 0
```

```
        for line in fd:
            index+=1
            if 'NIK,' in line:
                print (excel.split('.')[0]+'.csv'+' Kelewatan')
                print (index)
                count +=1
                df_0 = pd.read_csv(excel.split('.')[0]+'.csv', engine='python',
header = index-1)
                df_0.columns = map(lambda x: str(x).upper(), df_0.columns)
                df_miss = df_0.apply(lambda x: x.str.upper() if x.dtype ==
"object" else x)
                df_miss.to_csv(excel.split('.')[0]+'_nik.csv', encoding='utf-8',
index=False)
                shutil.copy(excel.split('.')[0]+'_nik.csv', Hasil_2)
            if index > 5:
                break
print ('\r\n All Checked, ada ', count, 'file mengandung NIK')
```

Kode 5.4 Pencarian Header dan Eliminasi Tabel

5.2.4 Eliminasi Kolom dan Pembersihan Baris Duplikat

Eliminasi kolom dilakukan pada *database* SKPD, *database* BDT dan *database* penduduk Kota Surabaya dimana kolom-kolom yang dirasa tidak diperlukan dan kolom yang tidak memiliki *value* sama sekali dieliminasi. Sebelum dilakukan eliminasi kolom pada *database* SKPD, dilakukan terlebih dahulu pembersihan baris duplikat, baris tanpa *value* sama sekali dan mengisi *value* yang kosong pada *database* SKPD.

```
import glob
import pandas as pd
import shutil

hasil_3 = r'Hasil_3'
for excel in glob.glob(r'Hasil_2\*.csv'):
    df_skpd = pd.read_csv(excel.split('.')[0]+'.csv', engine='python',
encoding='utf-8')
    dd_skpd = df_skpd.drop_duplicates().dropna(how='all')
    print(excel.split('.')[0]+'.csv')
    dd_skpd.fillna("0", inplace=True)
```

```
    print('row sebelum proses eliminasi duplikat:', len(df_skpd.index))
    dd_skpd.to_csv(excel.split('.')[0]+'_nd.csv', encoding='utf-8', index=False)
    shutil.move(excel.split('.')[0]+'_nd.csv', hasil_3)
    print('row setelah proses eliminasi duplikat:',len(dd_skpd.index))
```

Kode 5.5 Pembersihan Baris Database SKPD

Pada *database* SKPD, *database* BDT dan *database* penduduk Kota Surabaya nanti, kolom selain kolom NIK, Nama, Kecamatan dan Kelurahan akan dihapus.

```
import glob
import pandas as pd
hasil_4 = r'Hasil_SKPD'
for excel in glob.glob(r'Hasil_3\*.csv'):
    dc = pd.read_csv(excel.split('.')[0]+'.csv', engine='python', encoding='utf-
8')
    dx = dc.loc[:,dc.columns.isin(['NAMA', 'NIK', 'KECAMATAN',
'KELURAHAN'])|
                (dc.columns.str.startswith('NAMA_KECAMATAN'))|
                (dc.columns.str.startswith('NAMA_KELURAHAN'))|
                (dc.columns.str.startswith('KECAMATAN '))|
                (dc.columns.str.startswith('NAMA PENCARI'))|
                (dc.columns.str.startswith('NAMA PELAKU'))|
                (dc.columns.str.startswith('NAMA ATLET'))|
                (dc.columns.str.startswith('NAMA WARGA'))|
                (dc.columns.str.startswith('NAMA PENDIDIK'))|
                (dc.columns.str.startswith('NAMA PEMBUDIDAYA'))|
                (dc.columns.str.startswith('NAMA PEMILIK'))|
                (dc.columns.str.startswith('NAMA PMKS'))|
                (dc.columns.str.startswith('NAMA RESPONDEN'))|
                (dc.columns.str.startswith('NAMA PEMUDA'))]
    print(excel.split('.')[0]+'.csv')
    print('Header sebelum proses eliminasi:', list(dc),'\n')
    dx.to_csv(excel.split('.')[0]+'_cn.csv', sep=';', encoding='utf-8',
index=False)
    shutil.move(excel.split('.')[0]+'_cn.csv', hasil_4)
    print('Header setelah proses eliminasi:', list(dx),'\n')
```

Kode 5.6 Eliminasi Kolom SKPD

Kemudian dilakukan pengurutan ulang pada *header database* BDT dengan urutan NIK, Nama, Kecamatan dan Kelurahan agar nantinya memudahkan dalam proses *matching*. Selanjutnya, pembersihan baris duplikat dilakukan pada *database* SKPD dan *database* BDT dengan memeriksa apakah terdapat *record* yang memiliki *value* yang sama. Jika terdapat *record* dengan *value* yang sama (duplikat) maka salah satu *record* akan dihapus. Selain itu, jika terdapat baris *record* yang tidak memiliki *value* sama sekali juga akan dihapus.

```
import pandas as pd
df_bdt = pd.read_csv(r'BDT.csv', usecols = [4,5,7,8], dtype=str)
df_bdt = pd.DataFrame(np.row_stack([df_bdt.columns, df_bdt.values]),
columns=['KECAMATAN','KELURAHAN','NAMA', 'NIK'])
df_bdt.fillna("0000000000000000", inplace=True)
df_bdt = df_bdt[['NIK', 'NAMA', 'KECAMATAN', 'KELURAHAN']]
dd_bdt = df_bdt.drop_duplicates()
dd_bdt.to_csv(r'Data_siap_matching\BDT_cn.csv', sep=';', encoding='utf-8',
index=False)
```

Kode 5.7 Seleksi Kolom, Pengurutan Ulang Header dan Pembersihan Baris Database BDT

Terakhir dilakukan pengambilan kolom NIK, Nama, Kecamatan dan Kelurahan pada *database* penduduk Surabaya dan mengisi *value* yang kosong jika ada. Proses ini dilakukan dengan menggunakan kode yang ditunjukkan pada kode 5.8 berikut.

```
import pandas as pd

dbsby = pd.read_csv(r'dbsby.csv', sep=';', engine='python',
usecols=[0,1,5,6], dtype=str)
dbsby.rename(columns={'nik':'NIK','nama_lgkp':'NAMA','nama_kec':'KECAM
ATAN','nama_kel':'KELURAHAN'}, inplace=True)
dbsby.fillna("0000000000000000", inplace=True)
dbsby.to_csv(r'Data_siap_matching\DBSBY_cn.csv', sep=';', encoding='utf-8',
index=False)
```

Kode 5.8 Seleksi Kolom dan Pembersihan Value Kosong pada Database Penduduk Surabaya

5.3 String Matching

String matching dilakukan terhadap *file* SKPD, *database* BDT dan *database* penduduk Surabaya untuk melakukan standarisasi data yang ada. Proses ini dilakukan dengan memanfaatkan *shell script* AWK dari Linux.

5.3.1 Proses Matching Database BDT dengan Database Penduduk Surabaya

String matching pertama kali dilakukan pada *database* BDT dengan *database* penduduk Surabaya untuk membersihkan data kotor pada *database* BDT seperti kesalahan penulisan nama, NIK yang masih kosong, dan sebagainya. Proses ini dilakukan dengan menggunakan metode *fuzzy matching* yang memanfaatkan metode *levenshtein distance* yang dinormalisasi.

```
awk -F ";" 'function levdist(str1, str2,    l1, l2, tog, arr, i, j, a, b, c) {
    if (str1 == str2) {return 0}
    else if (str1 == "" || str2 == "") {return length(str1 str2)}
    else if (substr(str1, 1, 1) == substr(str2, 1, 1)) {
        a = 2
        while (substr(str1, a, 1) == substr(str2, a, 1)) a++
        return levdist(substr(str1, a), substr(str2, a))}
    else if (substr(str1, l1=length(str1), 1) == substr(str2, l2=length(str2), 1)) {
        b = 1
        while (substr(str1, l1-b, 1) == substr(str2, l2-b, 1)) b++
        return levdist(substr(str1, 1, l1-b), substr(str2, 1, l2-b))}
    for (i = 0; i <= l2; i++) arr[0, i] = i
    for (i = 1; i <= l1; i++) {
        arr[tog = ! tog, 0] = i
        for (j = 1; j <= l2; j++) {
            a = arr[! tog, j  ] + 1
            b = arr[  tog, j-1] + 1
            c = arr[! tog, j-1] + (substr(str1, i, 1) != substr(str2, j, 1))
            arr[tog, j] = (((a<=b)&&(a<=c)) ? a : ((b<=a)&&(b<=c)) ? b : c)
        }
    }
    return arr[tog, j-1]
}
```

```
function max(xx, yy) {
   return xx > yy ? xx : yy
}
function similarity(strr1,strr2, distp){
   distp = 100-(100*levdist(strr1, strr2)/max(length(strr1), length(strr2)))
   return distp
}
FNR == NR {
   seq[++n]=$1
   nama[NR]=$2
   kel[NR]=$4
   nikel[$1,$4]
   nmkel[$2,$4]
   next
}
BEGIN{getline; print}
{
   if(($1,$2) in nikel){print;next}
   else if(($2,$4) in nmkel){print;next}
   else{for (rA = 2; rA <= n; ++rA){
            if (substr($2, 2, 5) == substr(nama[rA], 2, 5)){
               if ( (similarity($2, nama[rA]) >= 80) &&
                  (similarity($4, kel[rA]) >= 80)){print;break}
            }
        }
   }
}' BDT_cn.csv DBSBY_cn.csv > BDT_clean.csv
```

Kode 5.9 Matching Database BDT dengan Database Penduduk Surabaya

Untuk setiap *records* yang ada di *database* BDT, pertama dilakukan pengecekan kesamaan antara NIK dan Kelurahan antar *records* pada *database* BDT dan *database* penduduk Surabaya, apabila sama akan diambil. Jika tidak sama akan dilakukan pengecekan lagi kesamaan Nama dan Kelurahan, apabila sama akan diambil. Jika ternyata masih tidak sama akan dicek kesamaan karakter ke 2 sampai karakter ke 6 pada kolom Nama, apabila sama kedua *records* tersebut akan masuk ke proses pengecekan *similarity* melalui fungsi *levenshtein distance*.

Fungsi *levenshtein distance* yang digunakan dalam proses ini diambil dari *script* AWK yang ditulis oleh *gnomon* dan dipublikasikan oleh Pierre Gaston [14], kemudian dibuat fungsi normalisasi untuk menghasilkan *string similarity* dalam bentuk persentase. Apabila *value* yang dihasilkan adalah 0% berarti kedua *string* yang dibandingkan sama sekali berbeda, sedangkan apabila *value* yang dihasilkan adalah 100% berarti kedua *string* yang dibandingkan merupakan kata yang benar-benar sama.

Dalam proses *matching* BDT dengan database penduduk Surabaya ini, diambil batas *value* 80% untuk masing-masing kolom Nama dan Kelurahan pada setiap *record* sebagai syarat bahwa kedua *record* tersebut merupakan *instance* yang sama (*pair*). Kolom NIK tidak diikutkan dikarenakan terdapat beberapa *records* pada *database* BDT yang tidak memiliki NIK, sedangkan kolom Kecamatan tidak diikutkan karena kolom Kelurahan dirasa sudah cukup untuk digunakan dalam proses *matching*. Selain itu juga untuk meningkatkan performa AWK agar tidak melakukan perhitungan terlalu banyak mengingat *database* penduduk Surabaya yang ukurannya sangat besar.

5.3.2 Proses Matching Database BDT dengan Database SKPD

Setelah didapatkan *database* BDT yang bersih dari proses *matching* sebelumnya, akan dilakukan *matching* antara *database* BDT dengan *database* SKPD dengan tujuan untuk mengetahui siapa saja yang terdata di *database* BDT yang juga terdata di *database* SKPD dan mengetahui bantuan-bantuan apa saja yang mereka terima. Pertama dibentuk sebanyak 10 *matchvar* dari *database* BDT bersih sesuai dengan yang dijelaskan pada bab sebelumnya, kemudian dari kesepuluh *matchvar* tersebut dilakukan *matching* dengan *database* SKPD. Jika ditemukan *records* yang *match* akan diambil dan disimpan

pada *file* BDT_SKPD.csv dimana hasil akhir dari *script* ini adalah tabel *database* Bantuan berisi nama *file* SKPD, NIK, Nama, Kecamatan dan Kelurahan.

```
awk -F ";" 'FNR == NR {
    rec[$1]=$0
    gsub (" ", "", $0)
    bdt1[$2$1$3$4] = $2$1$3$4  #Matchvar 1. nama nik kec kel
    bdt2[$2$1$3] = $2$1$3      #Matchvar 2. nama nik kec
    bdt3[$2$1] = $2$1          #Matchvar 3. nama nik
    bdt4[$2$1$4$3] = $2$1$4$3  #Matchvar 4. nama nik kel kec
    bdt5[$2$1$4] = $2$1$4      #Matchvar 5. nama nik kel
    bdt6[$1$3$4] = $1$3$4      #Matchvar 6. nik kec kel
    bdt7[$3$4$1$2] = $3$4$1$2  #Matchvar 7. kec kel nik nama
    bdt8[$4$1$2] = $4$1$2      #Matchvar 8. kel nik nama
    bdt9[$3$1$2] = $3$1$2      #Matchvar 9. kec nik nama
    bdt10[$1$2] = $1$2         #Matchvar 10. nik nama
    next
}
BEGIN{getline}
{
    gsub (" ", "", $0)
    a4[$1$2$3$4] = $1$2$3$4    #File SKPD yang memiliki 4 kolom
    a3[$1$2$3] = $1$2$3        #File SKPD yang memiliki 3 kolom
    a2[$1$2] = $1$2            #File SKPD yang memiliki 2 kolom

    if (a4[$1$2$3$4] in bdt1){print FILENAME";"rec[$2]}
    else if (a3[$1$2$3] in bdt2){print FILENAME";"rec[$2]}
    else if (a2[$1$2] in bdt3){print FILENAME";"rec[$2]}
    else if (a4[$1$2$3$4] in bdt4){print FILENAME";"rec[$2]}
    else if (a3[$1$2$3] in bdt5){print FILENAME";"rec[$2]}
    else if (a3[$1$2$3] in bdt6){print FILENAME";"rec[$1]}
    else if (a4[$1$2$3$4] in bdt7){print FILENAME";"rec[$3]}
    else if (a3[$1$2$3] in bdt8){print FILENAME";"rec[$2]}
    else if (a3[$1$2$3] in bdt9){print FILENAME";"rec[$2]}
    else if (a2[$1$2] in bdt10){print FILENAME";"rec[$1]}
}' BDT_clean.csv *cn.csv > BDT_SKPD.csv
```

Kode 5.10 Pembentukan Identifier dan ICA Matching

5.4 Visualisasi Data Terintegrasi

Setelah proses *matching database* BDT dengan *database* SKPD selesai, dilakukan innerjoin antara *database* Bantuan dengan *file* master_rawdata pada kolom id_skpd dengan kolm nama *file* SKPD dimana terdapat kode id_skpd disetiap nama *file* SKPD. *Innerjoin* dilakukan agar dapat mengetahui bantuan apa yang diterima oleh masing-masing orang yang terdata.

```
import pandas as pd
import numpy as np

bdt_bant = pd.read_csv(r'Data_siap_matching\Hasil_SKPD\BDT_SKPD.csv',
sep=';', engine='python', dtype=str)
bdt_bant = pd.DataFrame(np.row_stack([bdt_bant.columns,
bdt_bant.values]), columns=['BANTUAN','NIK','NAMA', 'KECAMATAN',
'KELURAHAN'])
daftar_file = pd.read_csv(r'daftarfileSKPDintervensi.csv', dtype=str)

daftar_file.rename(columns={'id_skpdfile':'BANTUAN','narasi':'BANTUAN
DITERIMA'}, inplace=True)
daftar_file = daftar_file.apply(lambda x: x.str.upper() if x.dtype == "object"
else x)

bdt_bant["BANTUAN"]= bdt_bant["BANTUAN"].str.slice(5, 9, 1)
bdt_skpd = pd.merge(bdt_bant, daftar_file, on=['BANTUAN'], how='left')
bdt_skpd = bdt_skpd.drop_duplicates()
bantuan_bdt = bdt_skpd.drop(['BANTUAN'], axis=1)
bantuan_bdt.to_csv(r'BANTUAN_BDT.csv', sep=';', encoding='utf-8',
index=False)
```

Kode 5.11 Innerjoin Database Terintegrasi

5.4.1 Tabel Data Bantuan Penduduk

Tabel data bantuan pemerintah Kota untuk penduduk Surabaya divisualisasikan dari *dataset* yang telah diintegrasikan yang berisi data penduduk Surabaya dan bantuan yang didapatkan dari pemerintah kota yang dihasilkan dari tahap sebelumnya. Tabel ini berisi nama orang yang mendapatkan bantuan beserta NIK, kecamatan dan kelurahan orang tersebut serta bantuan apa

yang diterima olehnya. Karena data ini bersifat rahasia, tabel hanya akan disimpan di komputer lokal dengan menggunakan format csv.

5.4.2 Histogram Bantuan Penduduk

Visualisasi ini berbentuk *bar chart* yang dibentuk berdasarkan *dataset* yang telah diintegrasikan yang berisi data penduduk Surabaya dan bantuan yang didapatkan dari pemerintah kota yang dihasilkan dari proses sebelumnya. Visualisasi ini dilakukan dengan menggunakan python dengan memanfaatkan *library* plotly yang menghasilkan *bar chart* jumlah penerima bantuan per kecamatan dengan mengambil jumlah *records* per kecamatan dari *database* yang telah terintegrasi.

```
import pandas as pd
import plotly.plotly as py
import plotly.graph_objs as go

df_bantuan = pd.read_csv(r'BANTUAN_BDT.csv', sep=';')
df_kec = df_bantuan['KECAMATAN'].value_counts()
trace1 = go.Bar(
    x=df_kec.index,
    y=df_kec,
    text=df_kec,
    textposition = 'outside',
    marker=dict(
        color='rgb(158,202,225)',
        line=dict(
            color='rgb(8,48,107)',
            width=1.5)
    )
)
data = [trace1]
layout = go.Layout(
    title = 'Jumlah Penerima Bantuan Setiap Kecamatan'
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='Penerima-Bantuan-Setiap-Kecamatan')
```

Kode 5.12 Bar Chart Penerima Bantuan Setiap Kecamatan

Kemudian, dilakukan juga visualisasi untuk memperlihatkan seberapa banyak orang yang menerima bantuan lebih dari sekali di setiap kecamatan menggunakan *bar chart*. Kode untuk visualisasi ini dapat dilihat pada kode 5.13.

```
import pandas as pd
import plotly.plotly as py
import plotly.graph_objs as go

df_bantuan = pd.read_csv(r'BANTUAN_BDT.csv', sep=';')
bantuan_dup = df_bantuan[df_bantuan.duplicated(['NIK'], keep=False)]
bantuan_dup2 = bantuan_dup.drop_duplicates(['NIK'], keep='last')]

df_kec_dup = bantuan_dup2['KECAMATAN'].value_counts()
trace1 = go.Bar(
    x=df_kec_dup.index,
    y=df_kec_dup,
    text=df_kec_dup,
    textposition = 'outside',
    marker=dict(
        color='rgb(158,202,225)',
        line=dict(
            color='rgb(8,48,107)',
            width=1.5)
    )
)
data = [trace1]
layout = go.Layout(
    title = 'Jumlah Penerima Bantuan Lebih Dari Sekali Setiap Kecamatan'
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='PenerimaBantuanLebihDariSekaliSetiapKecamatan')
```

Kode 5.13 Bar Chart Penerima Bantuan Lebih Dari Sekali Setiap Kecamatan

BAB VI
HASIL DAN PEMBAHASAN

Bab ini menjelaskan mengenai hasil serta analisis terhadap hasil yang diperoleh pada bab sebelumnya.

6.1 Analisis Seleksi File Berkaitan dengan Intervensi Penduduk

Dari sebanyak 2.013 jumlah *file* SKPD yang ada, hanya ditemukan sebanyak 214 *file* SKPD yang berkaitan dengan intervensi penduduk Surabaya. Hal ini menunjukkan bahwa terdapat banyak sekali *file* pada *database* SKPD yang tidak berkaitan dengan intervensi penduduk Surabaya. Untuk tabel daftar kode dan nama *file* SKPD yang berkaitan dengan intervensi penduduk dapat dilihat pada lampiran A.

6.2 Analisis Penyamaan Format File

Dilakukan penyamaan format *file* terhadap 214 *file* SKPD yang telah didapatkan dari proses seleksi sebelumnya. Proses ini dilakukan karena di dalam *database* SKPD berisi *file* dengan format yang berbeda-beda yaitu xls, xlsx dan csv. Hasil akhir dari proses ini adalah seluruh isi *file* SKPD memiliki format yang sama yaitu csv.

6.3 Analisis Pencarian Header dan Eliminasi Tabel

Dari 214 *file* SKPD yang telah ditemukan sebelumnya, hanya terdapat 1 *file* dengan *header* yang tidak terletak pada baris pertama, dimana *header file* tersebut terletak pada baris ke 3. Kemudian hasil seleksi dari 214 *file* SKPD yang ditemukan sebelumnya, didapatkan 137 *file* SKPD yang mengandung kolom NIK, sedangkan kolom NIK bersifat penting karena digunakan dalam proses *matching* antara *database* BDT yang telah dibersihkan dengan *database* SKPD.

6.4 Analisis Eliminasi Kolom

Eliminasi kolom dilakukan pada *database* SKPD, *database* BDT dan *database* penduduk Surabaya. Proses ini dilakukan untuk mengambil kolom-kolom yang diperlukan saja dari ketigas *database* tersebut. Hasil akhir dari proses ini adalah *database* BDT dan *database* penduduk Surabaya dengan kolom NIK, nama, kecamatan dan kelurahan. Sedangkan untuk *database* SKPD diambil kolom NIK, nama, kecamatan dan kelurahan jika ada.

6.5 Analisis Pembersihan Baris Duplikat

Pembersihan baris duplikat dilakukan terhadap *database* BDT dan *database* SKPD. Dari 514.425 baris *records* yang ada di *database* BDT, ditemukan sebanyak 2.238 baris duplikat dan baris tanpa *value* sama sekali sehingga hasil akhir jumlah baris BDT sebanyak 512.187 *records*. Sementara itu, tidak ditemukan baris duplikat pada database penduduk Surabaya. Untuk daftar hasil pembersihan baris duplikat pada *file* SKPD dapat dilihat pada lampiran A.

6.6 Analisis String Matching

Dilakukan 2 kali proses *string matching* yaitu *fuzzy matching* yang dilakukan antara *database* BDT dan *database* penduduk Surabaya agar menghasilkan *database* BDT yang bersih. Kemudian dilakukan proses *string matching* dengan mengadaptasi metode ICA yang dilakukan terhadap *database* BDT bersih dengan *database* SKPD.

Proses *matching* antara *database* BDT yang berjumlah 512.187 *records* dan *database* Penduduk Surabaya yang berjumlah 3.640.226 *records* menghasilkan 1.219.065 *records* yang disimpan sebagai *database* BDT bersih. Besarnya jumlah hasil yang melebihi jumlah *records database* BDT sendiri ini didapatkan karena tidak disertakannya kolom NIK dalam proses

matching membuat banyak *records* yang memiliki nama, kecamatan dan kelurahan yang sama persis ikut terambil.

Pertama dilakukan pencarian untuk *record* dengan NIK dan nama yang sama, jika ada akan diambil. Kemudian dicari *record* dengan NIK dan kelurahan yang sama, jika ditemukan akan diambil. Setelah itu apabila masih ada *record* dari *database* BDT yang belum *match*, akan dicek kesamaan karakter ke 2 sampai karakter ke 6 pada kolom Nama, apabila sama kedua *records* tersebut akan masuk ke proses pengecekan *similarity*. *Fuzzy matching* ini dilakukan untuk melakukan pembersihan pada *database* BDT seperti kesalahan penulisan nama, kesalahan penulisan NIK, dan NIK kosong. Beberapa penulisan nama salah pada *database* BDT yang berhasil diperbaiki ditunjukkan pada tabel 6.1 berikut.

No	Nama Sebelum Matching	Nama Sesudah Matching
1.	, BADAR RUDDIN	M. BADAR RUDDIN
2.	A AFIF AMIRUDIN	A AFIF AMRUDDIN
3.	,URTAFIAH	MURTAFIAH

Tabel 6.1 Contoh Perbaikan Penulisan Nama

Kemudian terdapat beberapa kesalahan penulisan NIK yang berhasil diperbaiki dimana contohnya ditunjukkan pada tabel 6.4 berikut.

Nama	NIK	
	Sebelum	Sesudah
MARGIAT	3578153006560006	3578153006550006
ABDUL AZIZ	3578301804800001	3578301204690002
ABADI	3578270301081326	3578271211710001

Tabel 6.2 Contoh Perbaikan Penulisan NIK

Beberapa contoh NIK kosong yang berhasil ditemukan ditunjukkan pada tabel 6.5 berikut.

Nama	NIK	
	Sebelum	Sesudah
SOEKIRAN	0000000000000000	3578100412480005
KUNTIYANAH	0000000000000000	3578267006650005
MARFEL	0000000000000000	3578060704080001

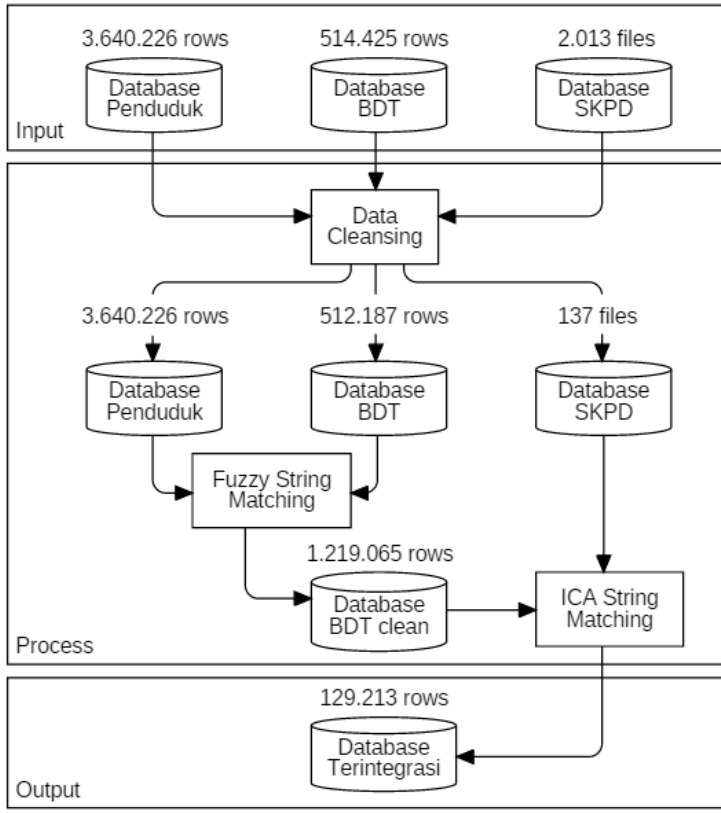
Tabel 6.3 Contoh Perbaikan NIK Kosong

Matching antara BDT bersih dengan SKPD dilakukan dengan menggunakan metode ICA yang bertujuan untuk menemukan *match* dari *database* BDT bersih ke *database* SKPD dimana isi dari *database* SKPD merupakan kumpulan *file* yang memiliki struktur kolom yang berbeda-beda. *Database* BDT bersih memiliki kolom NIK, Nama, Kecamatan dan Kelurahan. Sedangkan contoh struktur kolom pada beberapa *file* yang ada pada *database* SKPD adalah sebagai berikut:

- Nama, NIK, Kecamatan, Kelurahan
- Nama, NIK, Kelurahan, Kecamatan
- Kecamatan, Kelurahan, NIK, Nama
- NIK, Kecamatan, Kelurahan
- Kelurahan, NIK, Nama
- Nama, NIK

Hasil akhir dari *matching database* BDT bersih dengan *database* SKPD menghasilkan sebanyak 129.213 *records* bersih yang berisi NIK, Nama, Kecamatan, Kelurahan, dan Bantuan Diterima. Proses *matching* antara *database* BDT bersih dengan *database* SKPD hanya menghasilkan 129.213 *records* dikarenakan pada *database* SKPD masih terdapat beberapa *records* dengan *value* yang tidak valid, misalkan NIK yang

masih kosong, nama yang masih salah penulisannya dan sebagainya sehingga proses *matching* antara *database* BDT dengan *database* SKPD tidak berjalan dengan maksimal. Selain itu juga terdapat nama di *database* BDT yang tidak ada dalam *database* SKPD.



Gambar 6.1 Perubahan Jumlah Data

6.7 Penghitungan MRUI

Dari *database* terintegrasi yang dihasilkan pada tahap sebelumnya, didapatkan sebanyak 129.213 *records* dengan rincian ditunjukkan pada tabel 6.1 berikut.

Nama Variable	Simbol	Jumlah Kelas	Deskripsi Kelas
Kecamatan	$V1_{ui}$	34	Terdapat 34 kecamatan pada *database*
Kelurahan	$V2_{ui}$	154	Terdapat 154 Kelurahan pada *database*
Bantuan Diterima	$V3_{ui}$	37	Terdapat 37 jenis bantaun pada *database*

Tabel 6.4 Rincian Database Terintegrasi

Dengan menggunakan formula MRUI yang dapat dilihat pada tabel 2.3, hasil yang didapatkan berdasarkan keterangan diatas dapat dilihat pada tabel 6.2 berikut.

$$MRUI = 129.213 * \frac{1}{34} * \frac{1}{154} * \frac{1}{37} = \frac{129.213}{193.732} = 0.667$$

Tabel 6.5 Hasil MRUI Database Terintegrasi

Dari hasil penyelesaian diatas menunjukkan bahwa MRUI dihasilkan adalah kurang dari 1, sehingga nilai tersebut menunjukkan bahwa terdapat *unique identifier variable* untuk setiap *record* dan juga terdapat *unique identifier variable* lain yang masih bisa dibuat untuk meningkatkan hasil *matching* pada *records* yang ada.

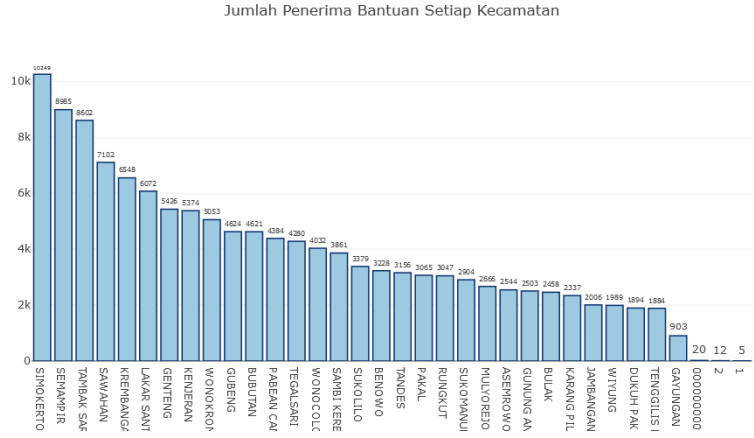
6.8 Analisis Visualisasi Data Terintegrasi

6.8.1 Tabel Data Bantuan Penduduk

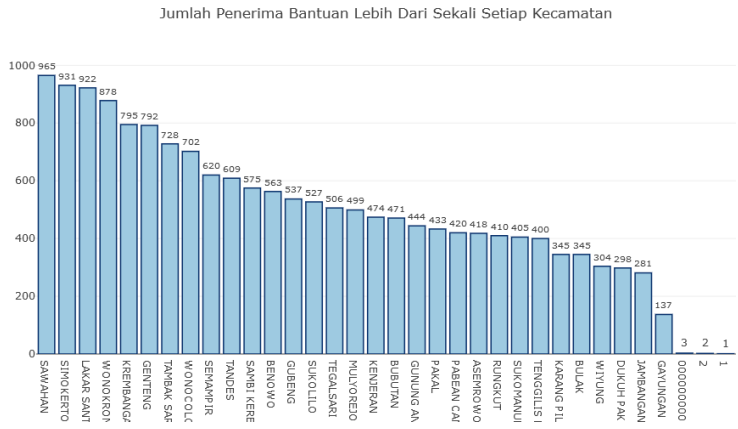
Data yang dihasilkan dari proses *matching* antara *database* BDT bersih dengan *database* SKPD disimpan dalam bentuk tabel yang berisi NIK, nama, kecamatan, kelurahan dan bantuan yang diterima. Tabel ini disimpan dalam bentuk csv. Karena data yang digunakan bersifat rahasia maka tabel hanya disimpan di komputer lokal dan penyimpanan online dengan pengaturan sebagai *file* privat. Hasil tabel data bantuan penduduk dapat dilihat pada lampiran A.

6.8.2 Histogram Bantuan Penduduk

Visualisasi *bar chart* dihasilkan menggunakan python dengan memanfaatkan *library* pandas dan plotly. *Histogram* mengenai jumlah penerima bantuan setiap kecamatan dapat dilihat pada gambar 6.1, sedangkan *histogram* mengenai jumlah penerima bantuan lebih dari sekali untuk setiap kecamatan dapat dilihat pada gambar 6.2 berikut.



Gambar 6.2 Histogram Jumlah Penerima Bantuan Setiap Kecamatan



Gambar 6.3 Histogram Jumlah Penerima Bantuan Lebih Dari Sekali Setiap Kecamatan

Untuk *bar chart* jumlah penerima bantuan setiap kecamatan yang ditunjukkan pada gambar 6.1, terlihat bahwa kecamatan Simokerto memiliki jumlah penerima bantuan yang paling besar, kemudian diikuti oleh kecamatan Semampir dan kecamatan Tambak Sari. Pada *bar chart* tersebut terlihat ada beberapa kecamatan yang tertulis sebagai "000000000", "1", dan "2". Hal ini disebabkan oleh adanya kolom kecamatan yang kosong atau berisi angka yang ada di *database* penduduk Surabaya sehingga ketika proses *matching* antara *database* BDT dan *database* penduduk Surabaya selesai masih terdapat *record* dengan kecamatan yang tidak terisi dengan benar. *Detail* isi dari *bar chart* pada gambar 6.1 dapat dilihat pada tabel 6.6 berikut.

No.	Kecamatan	Jumlah
1.	SIMOKERTO	10249
2.	SEMAMPIR	8985
3.	TAMBAK SARI	8602
4.	SAWAHAN	7102

No.	Kecamatan	Jumlah
5.	KREMBANGAN	6548
6.	LAKAR SANTRI	6072
7.	GENTENG	5426
8.	KENJERAN	5374
9.	WONOKROMO	5053
10.	GUBENG	4624
11.	BUBUTAN	4621
12.	PABEAN CANTIAN	4384
13.	TEGALSARI	4280
14.	WONOCOLO	4032
15.	SAMBI KEREP	3861
16.	SUKOLILO	3379
17.	BENOWO	3228
18.	TANDES	3156
19.	PAKAL	3065
20.	RUNGKUT	3047
21.	SUKOMANUNGGAL	2904
22.	MULYOREJO	2666
23.	ASEMROWO	2544
24.	GUNUNG ANYAR	2503
25.	BULAK	2458
26.	KARANG PILANG	2337
27.	JAMBANGAN	2006
28.	WIYUNG	1989
29.	DUKUH PAKIS	1894
30.	TENGGILIS MEJOYO	1884
31.	GAYUNGAN	903
32.	0000000000000000	20
33.	2	12

No.	Kecamatan	Jumlah
34.	1	5

Tabel 6.6 Jumlah Penerima Bantuan Setiap Kecamatan

Kemudian untuk *bar chart* jumlah penerima bantuan lebih dari sekali untuk setiap kecamatan ditunjukkan pada gambar 6.2. Pada gambar tersebut terlihat bahwa jumlah penerima bantuan lebih dari sekali yang paling tinggi terdapat pada kecamatan Sawahan yaitu sebanyak 965 orang, kemudian diikuti oleh kecamatan Lakar Santri sebanyak 931 orang dan kecamatan Simokerto sebanyak 922 orang. Untuk *detail* isi dari *bar chart* pada gambar 6.7 dapat dilihat pada tabel 6.4 berikut.

No.	Kecamatan	Jumlah
1.	SAWAHAN	965
2.	LAKAR SANTRI	931
3.	SIMOKERTO	922
4.	WONOKROMO	878
5.	GENTENG	795
6.	KREMBANGAN	792
7.	TAMBAK SARI	728
8.	WONOCOLO	702
9.	SEMAMPIR	620
10.	TANDES	609
11.	SAMBI KEREP	575
12.	BENOWO	563
13.	GUBENG	537
14.	SUKOLILO	527
15.	MULYOREJO	506
16.	TEGALSARI	499
17.	KENJERAN	474
18.	BUBUTAN	471

No.	Kecamatan	Jumlah
19.	GUNUNG ANYAR	444
20.	PAKAL	433
21.	PABEAN CANTIAN	420
22.	ASEMROWO	418
23.	SUKOMANUNGGAL	410
24.	RUNGKUT	405
25.	TENGGILIS MEJOYO	400
26.	KARANG PILANG	345
27.	BULAK	345
28.	WIYUNG	304
29.	DUKUH PAKIS	298
30.	JAMBANGAN	281
31.	GAYUNGAN	137
32.	0000000000000000	3
33.	2	2
34.	1	1

Tabel 6.7 Jumlah Penerima Bantuan Lebih Dari Sekali Setiap Kecamatan

Halaman ini sengaja dikosongkan

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan beserta saran yang didapatkan dari penelitian tugas akhir ini.

7.1 Kesimpulan

Berdasarkan dengan pengerjaan tugas akhir dengan judul ”Integrasi Data Bantuan Pemerintah Kota Terhadap Penduduk Surabaya Menggunakan *Identity Correlation Approach*” yang telah dilakukan dapat disimpulkan beberapa hal sebagai berikut:

1. *Database* SKPD dan BDT yang dimiliki pemerintah Kota Surabaya masih belum di tangani dengan baik dibuktikan dengan kotornya isi kedua *database* tersebut seperti kesalahan penulisan, kolom bersifat penting yang masih kosong, dan sebagainya sehingga perlu dilakukan pembersihan bertahap untuk dapat mengolah kedua *data* tersebut.
2. Proses *matching* antara *database* BDT dan *database* penduduk Surabaya tidak dapat dilakukan dengan optimal dikarenakan banyaknya *records* pada *database* BDT dengan NIK yang tidak valid meskipun telah menggunakan metode *fuzzy matching*.
3. Berdasarkan hasil *matching* yang telah dilakukan, metode adaptasi *Identity Correlation Approach* berhasil menemukan pasangan *record* dari *database* BDT ke *database* SKPD meskipun *database* SKPD berisi kumpulan *file* yang memiliki bentuk dan struktur kolom yang berbeda-beda.

7.2 Saran

Dalam pengerjaan tugas akhir, terdapat beberapa saran yang diharapkan dapat bermanfaat bagi pihak BAPPEKO Surabaya maupun untuk pengembangan penelitian ke depan, yaitu:

1. Melakukan standarisasi cara melakukan *input* data pada setiap dinas untuk mengurangi kesalahan penulisan dan data redundan, mengingat banyaknya ditemukan data yang tidak valid, data kosong, dan data redundan yang ditemukan baik pada *database* SKPD dan *database* BDT. Hal ini akan mempermudah dalam proses pengintegrasian antar *database*.
2. Melakukan pengecekan kembali pada *database* penduduk Surabaya, dikarenakan telah ditemukannya beberapa *value* kosong pada *database* tersebut. Mengingat *database* ini merupakan *master database* penduduk kota Surabaya dan dianggap sebagai *database* dengan data yang paling bersih.

DAFTAR PUSTAKA

[1] M. Madhikermi, S. Kubler, J. Robert, A. Buda dan K. Framling, "Data Quality Assessment of Maintenance Reporting Procedures," *Expert Systems with Applications,* vol. 63, pp. 145-164, 2016.

[2] "Pelopori e-Government, Pemkot Surabaya Sudah Ciptakan Ratusan Aplikasi," 5 April 2018. [Online]. Available: https://humas.surabaya.go.id/2018/04/05/pelopori-e-government-pemkot-surabaya-sudah-ciptakan-ratusan-aplikasi/. [Diakses 31 Januari 2019].

[3] "Badan Perencanaan Pembangunan Kota Surabaya," 2019. [Online]. Available: https://bappeko.surabaya.go.id/#. [Diakses 31 Januari 2019].

[4] K. McCormack dan M. Smyth, "A Mathematical Solution to String Matching for Big Data Linking," *Journal of Statistical Science and Application,* vol. 5, pp. 39-55, 2017.

[5] K. McCormack dan M. Smyth, "Big Data Matching Using the Identity Correlation Approach," *First International Conference on Advanced Research Methods and Analytics, CARMA2016,* pp. 46-55, 2016.

[6] C. S. Rao dan K. B. Raju, "Single and Multiple Pattern String Matching Algorithm," *Indian Journal of Science and Technology,* vol. 10, no. 3, pp. 1-6, 2017.

[7] B. F. Yusuf, "Integrasi Data dan Visualisasi Graf pada Aplikasi Halal Nutrition Food," *Institut Teknologi Sepuluh Nopember Surabaya,* 2018.

[8] A. C. Najib, "Rancang Bangun Aplikasi Android Halal Nutrition Food Menggunakan Kombinasi Query Independent dan Query Dependent Ranking," *Institut Teknologi Sepuluh Nopember Surabaya,* 2018.

[9] E. Sosiawan dan Arief, Tantangan dan Hambatan Dalam Implementasi E-government di Indonesia, 2008.

[10] V. Elysia, A. Wihadanto dan Sumartono, "Implementasi e-Government untuk Mendorong Pelayanan Publik Yang Terintegrasi di Indonesia," pp. 353-380, 2017.

[11] P. Christen, Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection, Canberra: Springer, 2012.

[12] C. Chen, "Information Visualization," *WIREs Comp Stat,* vol. 2, 2010.

[13] M. Khan, "Data and Information Visualization Methods, and Interactive Mechanisms: A Survey," *International Journal of Computer Applications,* vol. 34, no. 1, p. 2, 2011.

[14] P. Gaston, "LevenshteinEditDistance," 2010. [Online]. Available: http://awk.freeshell.org/LevenshteinEditDistance. [Diakses 11 Juni 2019].

LAMPIRAN A. DATA

Data yang ada di bawah ini dapat dilihat pada tautan https://intip.in/TAintervensiSBY2019.

A.1 Daftar kode dan nama file SKPD yang berkaitan dengan intervensi penduduk.

A.2 Hasil pembersihan baris duplikat pada file SKPD

A.3 Tabel Data Bantuan Penduduk Terintegrasi

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis lahir di Bojonegoro pada tanggal 10 Mei 1998. Penulis berasal dari Bojonegoro dan telah menempuh pendidikan formal yaitu; MI Tarbiyatul Athfal Dukohlor, SMPN 1 Padangan, dan MA Al Anwar Sarang Rembang. Pada tahun 2015, penulis melanjutkan pendidikan di Departemen Sistem Informasi, FTIK, Institut Teknologi Sepuluh Nopember Surabaya dan terdaftar sebagai mahasiswa dengan NRP 05211540007003. Selama menjadi mahasiswa, penulis juga aktif mengikuti kegiatan kemahasiswaan seperti organisasi CSSMoRA ITS dan panitia event tingkat institut, regional maupun nasional. Pada tahun keempat, penulis mengambil bidang minat Laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI). Penulis dapat dihubungi melalui email abdul.azizun15@gmail.com.