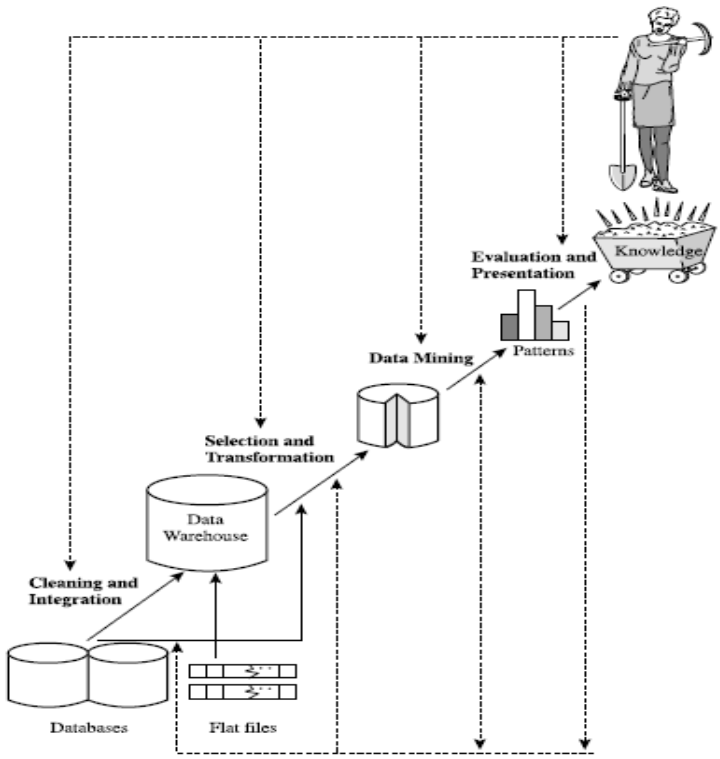
BAB 2

LANDASAN TEORI

2.1. Data *Mining*

Data *mining* adalah bagian dari *knowledge discovery* di *database* yang menganalisa *database* berukuran besar untuk menemukan pola yang berguna pada data (Silberschatz, *et* al. 2006). Data *mining* merupakan proses yang mempekerjakan teknik pembelajaran komputer (*machine learning*) untuk mengekstraksi pengetahuan (*knowledge*) secara otomatis (Hermawati, 2013).



Gambar 2.1. Tahap Pada *Knowledge Discovery Database* (Han & Kamber, 2006)

Tahap pada *knowledge discovery database* (Han & Kamber, 2006), yaitu :

1. *Data cleaning* (untuk menghilangkan *noise* dan data yang tidak konsisten).
2. *Data integration* (mengkombinasikan beberapa sumber data).
3. *Data selection* (pengambilan data yang relevan dengan analisis *database*).
4. *Data transformation* (mengubah bentuk data kedalam bentuk yang sesuai).
5. *Data mining* (proses penting penggunaan metode yang diterapkan untuk mengekstrak pola data).
6. *Pattern evaluation* (mengidentifikasi pola-pola yang menarik yang mewakili pengetahuan didasarkan pada beberapa langkah yang menarik).
7. *Knowledge presentation* (visualisasi dan teknik representasi pengetahuan digunakan untuk menyajikan pengetahuan yang berguna untuk pengguna).

Adapun beberapa teknik data *mining* yang dapat digunakan (Hermawati, 2013), adalah sebagai berikut :

1. Klasifikasi

Klasifikasi adalah menentukan sebuah *record* data baru ke salah satu dari beberapa kategori yang telah didefinisikan sebelumnya, klasifikasi itu merupakan *supervised learning*.

2. Regresi

Memprediksi nilai dari suatu variabel yang berkelanjutan berdasarkan nilai dari variable yang lain, dengan mengasumsikan sebuah model ketergantungan linear atau nonlinear. Teknik ini banyak dipelajari dalam statistika, bidang jaringan syaraf tiruan (*neural network*).

3. Klasterisasi (*clustering*)

Mempartisi *data set* menjadi beberapa bagian atau kelompok sedemikian rupa sehingga elemen-elemen dari suatu kelompok tetentu memiliki bagian yang digunakan bersama, dengan tingkat kesamaan yang tinggi dalam satu kelompok dan tingkat kesamaan antar kelompok yang rendah disebut juga *unsupervised learning*.

4. Kaidah Asosiasi (*association rules*)

Mendeteksi beberapa atribut yang muncul bersamaan dengan frekuensi yang sering, dan membentuk sejumlah kaidah dari atribut tersebut. Tujuannya adalah untuk menemukan pola yang menarik dengan cara yang efisien (Prasetyo, 2014).

5. Pencarian pola sekuensial (*sequence mining*)

Mencari sejumlah kejadian yang secara umum terjadi bersama-sama.

2.2. Data *Cleaning*

Data cleaning merupakan salah satu tahap pada data *mining*. Data *cleaning* biasa disebut dengan data *cleansing* atau *scrubbing*. Proses *data cleaning* dilakukan untuk menghilangkan kesalahan informasi pada data (Rahm & Do, 2000). Sehingga proses data *cleaning* digunakan untuk menentukan data yang tidak akurat, tidak lengkap atau tidak benar dan akan memperbaiki kualitas data melalui pendeteksian kesalahan pada data (Tamilselvi & Saravan, 2010). Data *cleaning* juga merupakan langkah yang dilakukan untuk mendeteksi serta mengkoreksi atau menghapus sejumlah *record* yang kurang atau tidak akurat yang disebabkan adanya kesalahan pada data (Riezka, 2010).

Data *cleaning* dapat dilakukan dengan satu sumber atau beberapa sumber data, juga terdapat permasalahan pada level skema ataupun level *instance*. Permasalahan pada level skema dapat diselesaikan dengan perbaikan desain, *translation* dan *integration* skema. Sedangkan pada tingkat *instance* terdapat kesalahan dan tidak konsisten pada data yang merupakan fokus permasalahan yang dapat diselesaikan dengan *data cleaning*. (Rahm & Do, 2000).

Data *cleaning* membutuhkan waktu yang minimum dalam memperoleh data yang berkualitas baik, karena *data set* yang digunakan merupakan data yang berukuran besar (Riezka, 2010). Kemungkinan terjadinya kesalahan pada data pada saat nama yang sama digunakan untuk objek yang berbeda atau nama yang berbeda digunakan untuk objek yang memiliki kemiripan nilai pada data. Kesalahan pada data terjadi disebabkan *human errors* atau data telah rusak pada saat penyimpanan data. Beberapa permasalahan data *cleaning* yaitu, *incomplete*, *inconsistence*, *outliers*, dan *redundancy* (duplikat). Pada penelitian ini akan difokuskan pada permasalahan data duplikat.

Customer (Source 1)

CID	Name	Street	City	Sex
11	Kristen Smith	2 Hurley PI	South Fork, MN 48503	0
24	Chritian Smith	Hurley St 2	S Fork MN	1

Client (Source 2)

Cno	Last Name	First Name	Gender	Street	Phone/Fax
11	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542/ 333-222-6599
493	Smith	Kris L.	F	2 hurley Place, South Fork MN, 48503-5998	444-555-6666

Gambar 2.2. Gambar Duplikasi Data (Riezka, 2010)

Pada gambar 2.2 terdapat format data yang memiliki hubungan *relational*, tetapi terdapat juga beberapa permasalahan data pada kedua sumber tersebut. Pada bagian skema data, ada beberapa permasalahan pada penamaan (persamaan antara "*Customer/Client*", "Cid/Cno", "*Sex/Gender*") dan pada bagian permasalahan struktural (perbedaan pada atribut "*Name*" dan "*Address*"). Permasalahan pada duplikat data dapat dieliminasi dengan Metode *sort-neighborhood* atau metode pegembangannya *multi-pass neighborhood*.

2.3. Data Duplikat

Permasalahan utama yang menyebabkan adanya data duplikat adalah terjadinya *overlapping* data atau data yang tumpang tindih (Riezka, 2010). *Overlapping* pada data umumnya terjadi pada data-data identitas seperti data mahasiswa, data pegawai dan data *costumer*.

2.4. *Pre-Processing*

1. *Lowercase*

   Mengubah keseluruhan teks menjadi huruf kecil (*toLowerCase*).

2. *Regular Expression*

   Regular expression merupakan karakteristik untuk mencocokan token.

2.5. Multi-Pass Neighborhood

Multi-pass neighborhood merupakan metode pengembangan metode *sorted neighbourhood* yang dapat mengunakan *key* yang berbeda dan menggunakan *window* yang lebih kecil (Riezka, 2010). Selanjutnya dapat dilakukan tahap *transitive closure* untuk mengetahui relasi *record* (Tamilselvi & Saravan, 2010). Metode ini memiliki tiga tahapan dasar, yaitu :

1. Pembentukan *key*

Membentuk *key* dilakukan pada setiap *record* dengan mengekstrak *field* atau menggunakan sebagian *field* yang *relevan* dari atribut yang dipilih

First Name	Last Name	Address	ID	*Key*
Sal	Stolfo	123 First Street	45678987	STLSAL123FRSI456
Sal	Stolfo	123 First Street	45678986	STLSAL123FRSI456
Sal	Stolpho	123 First Street	45678987	STLSAL123FRSI456
Sal	Stiles	123 Forest Street	45654321	STLSAL123FRSI456

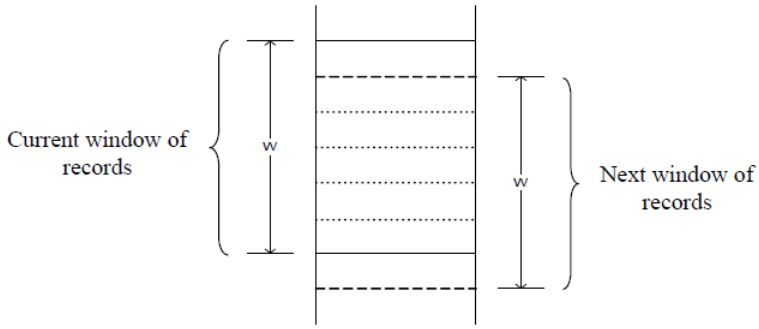
Gambar 2.3. Gambar Tabel Data Dan *Key* (Riezka, 2010)

2. *Sorting* data

Mengurutkan *record* pada *list* data menggunakan *key* yang telah dibentuk. Pada proses ini dilakukan pengurutan data berdasarkan *key* secara *ascending* (Z-A). Pengurutan *record* dilakukan bertujuan untuk memudahkan penghitungan *edit distance* pada *string* hanya pada *key* yang berada pada satu *window*.

3. *Merge* data

Memindahkan *fixed sized window* pada data yang telah berurutan untuk membatasi perbandingan pada tahap berikutnya saat membandingkan nilai *record* yang ada pada *window* dengan ukuran w. *Record* pertama pada *window* akan bergeser keluar dari *window* setelah dibandingkan dengan *record* sebelumnya (w-1) untuk menemukan *record* yang sesuai.



Gambar 2.4. Window Pada Tahap Merge (Riezka, 2010)

2.6. N-gram

N-gram merupakan algoritma *string similarity* (pencocokan *string*), dengan pergeseran *window* dengan panjang n sepanjang karakter *string* (Azma, 2006). *N-gram* memisahkan *string* menjadi potongan *string* dengan panjang n.

2.7. Jaccard

Jaccard similarity merupakan algoritma untuk mengukur kesamaan antara dua *string* (Agarwal et al, 2014). Metode ini membandingkan *string* dengan melihat posisi penulisan yang berbeda (Liliana et al, 2012). Rentang nilai yang dihasilkan berupa 0 untuk menandakan tidak ada kesamaan dan 100 adalah sama persis (Chahal, 2016). Nilai antara 0 dan 100 menunjukkan probabilitas kemiripan antara dua *string*. Perhitungan *jaccard* untuk mencari rentang nilai kemiripan pada data menggunakan persamaan (2.1).

$$J\,(A,B) = \frac{|\,A \cap B\,|}{|\,A \cup B\,|} \qquad (2.1)$$

2.8. Approximate String Matching

Approximate string matching merupakan pencocokan *string* berdasarkan kemiripan jumlah karakter dan susunan karakter antara dua *string* (Primadani, 2014). Beberapa algoritma yang merupakan *approximate string matching* seperti, *hamming, leveinsthein, damerau-leveinsthein, jaro-winkler, smith-waterman* dan lainnya. Pada sistem ini menggunakan metode *leveinsthein distance.*

2.6.1. Leveinsthein Distance

Leveinsthein distance merupakan algoritma yang dapat membandingkan *string* dengan panjang yang berbeda. *Leveinsthein distance* melakukan perhitungan jumlah minimal operasi penambahan (*insert*), penghapusan (*delete*), dan penggantian karakter (*substitute*) yang dibutuhkan untuk menyamakan kedua *string* (Riezka, 2010). Untuk mengetahui *leveinsthein distance* tahap yang dilakukan adalah *edit distance,* untuk mengetahui nilai jumlah modifikasi yang dibutuhkan antara suatu bentuk *string* dengan *string* lain. (Ugon, et al. 2015). Persamaan *Leveinsthin distance* untuk menghitung *edit distance* antara *string* s dan *string* t (Primadani, 2014), dapat dilihat pada persamaan 2.2 dan 2.3 dibawah ini :

$$D(s,\ t) = d(s_1, t_1) + d(s_2, t_2) + \ldots + d(s_l,\ t_l) \quad (2.2)$$

$$D(s, t) = \sum_{i=1}^{l} d(s_i,\ t_i) \quad (2.3)$$

dimana : $s_i, t_i \in V$ untuk $i = 1,2,\ldots,l$

$d(s_i, t_i) = 0$ jika $s_i = t_i$

$d(s_i, t_i) = 1$ jika $s_i \neq t_i$

D(s,t) merupakan banyaknya operasi minimum dari operasi penghapusan, penyisipan, dan penggantian untuk menyamakan *string* s dan t. Beberapa penggunaan operasi tersebut untuk menentukan jarak *leveinsthein* sebagai berikut ini.

1. Operasi penghapusan

   Operasi penghapusan dilakukan dengan menghapus karakter pada indeks tertentu untuk menyamakan *string* sumber (s) dengan *string* target (t).

2. Operasi penyisipan

   Operasi penyisipan dilakukan dengan menyisipkan karakter pada indeks tertentu untuk menyamakan *string* sumber (s) dengan *string* target (t).

3. Operasi pengantian

   Operasi penggantian dilakukan dengan mengganti karakter pada indeks tertentu untuk menyamakan *string* sumber (s) dengan *string* target (t).

2.9. Transitive closure

Transitive closure merupakan tahap untuk dapat mendeteksi apabila *record* a dan b serupa, *record* b dan c serupa sehingga tahap ini dapat menandai *record* a dan c serupa. Hal ini dilakukan apabila hubungan relasi tidak terdeteksi oleh *equational theory*. Penggunaan *transitive closure* pada setiap *single pass* di metode *sorted-neighborhood* dapat mengurangi ukuran *scanning window* (Tamilselvi, J.J. & Saravan, V., 2010).

2.10. Peneliti Terdahulu

Efisiensi pada *data cleaning* dapat dilakukan untuk mendapatkan kualitas data yang baik (Riezka, 2010). Pengimplementasian data *cleaning* pada penelitian ini dilakukan dengan menggunakan data mahasiswa menggunakan metode *multi-pass neighborhood (MPN*). Metode *Multi-pass neighborhood* dapat mendeteksi duplikasi *record* dengan lebih efisien pada data identitas mahasiswa dengan jumlah data 1987 *record*. Proses dilakukan dengan pembuatan *key* dari beberapa kolom pada *record* dan pengurutan *key* dilakukan berdasarkan *last name, first name* dan alamat. Ditetapkan tiga buah parameter untuk melihat kinerja metode tersebut, yaitu ukuran lebar *window*, pengaruh kombinasi *rule* yang digunakan, dan jumlah *passes* yang dipakai saat eksekusi. Penentuan ukuran lebar *window* dan proses pengambilan keputusan duplikat data pada penelitian ini menggunakan *leveinsthein distance*. Sedangkan untuk mengetahui hubungan record yang diidentifikasi duplikat dilakukan menggunakan *transitive closure*.

Penelitian yang dilakukan oleh (He et al, 2011) yaitu penggunaan *multi-pass sorted-neighborhood* (MPN) untuk melakukan efisiensi *cleaning* data duplikat yang dapat dilakukan dengan baik. Pada penelitian ini proses penentuan ukuran *window* dilakukan menggunakan *n-gram* dan proses pengambilan keputusan duplikasi data dilakukan menggunakan metode *smith-waterman*. Sedangkan, hubungan duplikasi pada data duplikat dilakukan menggunakan *transitive closure*.

Proses *cleaning* pada data digunakan untuk mendapatkan kualitas data yang baik. Data cleaning pada penelitian ini dilakukan dengan algoritma *n-gram* pada data yang tidak konsisten dan skema lokasi atau wilayah pada data BKKBN, DEPTAN, dan BPS (Azma, S, 2006).

Metode *Leveinsthein distance* juga pernah digunakan untuk penelitian lain yaitu penelitian yang dilakukan oleh (Primadani, 2014). *Leveinsthein distance* digunakan untuk pencarian judul buku pada katalog perpustakaan. Algoritma ini digunakan untuk menghasilkan layanan *autocomplete* dalam memprediksi judul buku yang diberikan oleh pengguna.

Pada penelitian ini, penulis menggunakan *leveinsthein distance* untuk menyelesaikan pengambilan keputusan duplikasi data pada data identitas *costumer*. Kelebihan dari metode *leveinsthein distance* ini adalah dapat mengetahui kemiripan *string* pada data dengan *edit distance* berupa jarak antara dua *string* (Ugon et al, 2015). Beda penelitian ini dari yang lain adalah metode *n-gram* dan *jaccard* pada penentuan ukuran *window* dan data yang dapat di*input*kan dalam excell. Sedangkan untuk *pre-processing* data pada penelitian ini digunakan *regular expression*, mengubah data menjadi *lowercase* dan memisahkan atribut nama menjadi *last name* dan *first name* (Riezka, 2010). Dan proses penandaan duplikat data pada penelitian ini dilakukan menggunakan metode *transitive closure*.

Tabel 2.1 Penelitian Terdahulu

No.	Peneliti / Tahun	Teknik yang digunakan	Keterangan
1.	Riezka, A (2010)	Metode *multi-pass neighbohood (MPN)*dan *transitive closure*	Metode *multi-pass neighbohood (MPN)* dan *transitive closure* mampu melakukan pengidentifikasian record yang duplikat menggunakan data mahasiswa.
2.	He *et* al (2011)	Metode *n-gram* dan *smith-waterman*	Metode *n-gram* dan *smith-waterman* mampu menghitung kesamaan string pada data *cleaning* untuk duplikat data.

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

No.	Peneliti / Tahun	Teknik yang digunakan	Keterangan
3.	Azma, S (2006)	Metode *n-gram*	Mampu melakukan proses *cleaning* data yang tidak konsisten pada data wilayah pada data BKKBN, DEPTAN, dan BPS.
4.	Primadani,Y (2014)	Metode *leveinsthein distance*	Mampu melakukan simulasi fitur *autocomplete* pada aplikasi katalog perpustakaan menggunakan *Leveinsthein distance*.

Adapun perbedaan penelitian yang dimiliki oleh penulis dengan penelitian terdahulu adalah :

1. Riezka, A (2010)

   Sistem ini tidak dapat menggunakan data selain yang ada pada database dan menerapkan metode *multipass neighborhood, leveinsthein distance* dan *transitive closure* untuk proses *cleaning* data mahasiswa yang duplikat. Sedangkan pada penelitian ini dapat membaca data dari file *excel* menggunakan metode *multipass neighborhood, n-gram, jaccard, leveinsthein distance* dan *transitive closure* untuk proses *cleaning* data identitas pelanggan perusahaan yang duplikat.

2. He *et* al (2011)

   Sistem ini menggunakan metode *multipass neighborhood, smith waterman, n-gram* dan *transitive closure* proses *cleaning* data. Sedangkan pada penelitian ini menggunakan metode *multipass neighborhood, n-gram, jaccard, leveinsthein distance* dan *transitive closure* untuk proses *cleaning* data.

3. Azma, S (2006)

   Sistem ini melakukan proses *cleaning* data yang tidak konsisten pada data wilayah pada data BKKBN, DEPTAN menggunakan *n-gram,* Sedangkan pada penelitian ini menggunakan metode *n-gram* untuk proses *cleaning* data data identitas duplikat.

4. Primadani,Y (2014)

   Sistem ini melakukan simulasi fitur *autocomplete* pada aplikasi katalog perpustakaan menggunakan *leveinsthein distance*. Sedangkan pa*da* penelitian ini menggunakan metode *leveinsthein distance* untuk proses *cleaning* data pada identifikasi data duplikat.