# BAB 1 PENDAHULUAN

Pada bab ini akan dijelaskan mengenai beberapa hal dasar dalam pembuatan proposal penelitian yang meliputi latar belakang, perumusan masalah, tujuan, manfaat, kontribusi penelitian, dan batasan masalah.

1. Latar Belakang

Rekayasa kebutuhan (*software engineering*) adalah suatu proses mewujudkan serangkaian layanan yang dibutuhkan oleh pengguna atas suatu sistem dan batasan-batasan yang harus dipenuhi ketika dibangun maupun dioperasikan [1]. Penelusuran, pengumpulan dan penyajian segala kebutuhan pengguna kedalam sebuah Dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL) adalah tahap awal dalam pengembangan perangkat lunak. Karena Dokumen Spesifikasi Perangkat Lunak (SKPL) memiliki peran yang sangat penting dalam menghasilkan perangkat lunak yang berkualitas tinggi.

Penjaminan kualitas perangkat lunak (PKPL) merupakan sebuah proses yang memantau seluruh proses pengembangan perangkat lunak untuk memastikan bahwa proses-proses tersebut dapat menghasilkan perangkat lunak yang berkualitas. Proses Penjamin Kualitas Perangkat Lunak (PKPL) mencakup seluruh tahap pada Daur Hidup Perangkat Lunak (DHPL). Proses PKPL, khususnya pada tahap spesifikasi kebutuhan, dinilai sangat penting menurut Boehm dan Basili [2], untuk mencegah peningkatan biaya yang diperlukan saat pembangunan perangkat lunak. Perangkat lunak yang telah selesai diperbaiki memerlukan biaya lebih besar daripada pada saat tahap spesifikasi kebutuhan dan desain perangkat lunak.

Dalam melakukan spesifikasi kebutuhan terdapat tiga metode pendekatan yang dapat digunakan yaitu *natural languange*, *semi-formal languange*, dan *formal languange*. Natural languange adalah pendekatan spesifikasi kebutuhan yang menggunakan bahasa sehari - hari, semi-formal languange menggunakan bahasa grafikal yang disertai penjelasan teks, dan formal languange menggunakan konsep matematika seperti *finite-state machines*. Menurut survei yang pernah dilakukan Rossi pada tahun 1999 [2], sebanyak 71,8% dokumen spesifikasi kebutuhan perangkat lunak dibuat dengan menggunakan pendekatan *natural languange*. Hal ini disebabkan karena pendekatan tersebut lebih mudah dipahami oleh banyak pihak.

Meyer menjelaskan bahwa proses spesifikasi kebutuhan perangkat lunak yang menggunakan bahasa natural memiliki kekurangan jika dibandingkan dengan bahasa formal [2]. Hal tersebut disebabkan karena spesifikasi kebutuhan yang menggunakan *natural languange* memiliki tujuh kesalahan umum yang sering dilakukan oleh pengembang perangkat lunak yang dirumuskan oleh Meyer sebagai *The seven sins of specifier*. Salah satu dari *The seven sins of specifier* adalah *noise*. *Noise* dapat muncul pada proses spesifikasi kebutuhan di dalam dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL). Noise terjadi jika dalam spesifikasi kebutuhan, pengembang perangkat lunak menyertakan suatu informasi yang tidak relevan dengan keseluruhan kebutuhan perangkat lunak. Hal tersebut dapat membingungkan pembaca dan dapat berakibat buruk pada tahap - tahap pengembangan perangkat lunak selanjutnya.

1. Perumusan Masalah

Rumusan masalah yang diangkat dalam penelitian ini adalah sebagai berikut.

1. Bagaimana sebuah kalimat dianggap sebagai *noise* pada dokumen SKPL?
2. Bagaimana melakukan klasterisasi pernyataan kebutuhan dari suatu dokumen SKPL?
3. Bagaimana memodelkan noise sebagai pencilan dari proses klasterisasi pernyataan kebutuhan dalam dokumen SKPL?
4. Apa saja parameter yang berpengaruh dalam akurasi pendeteksian pencilanpada proses klasterisasi pernyataan kebutuhan dalam dokumen SKPL?
5. Tujuan

Tujuan yang akan dicapai dalam pembuatan tesis ini adalah mendeteksi *noise* pada dokumen SKPL dengan cara mendeteksi pencilan dari proses klasterisasi pernyataan kebutuhan menggunakan metode klasterisasi spektral.

1. Manfaat

Manfaat dari penelitian ini adalah memberikan kemudahan kepada perekayasa kebutuhan untuk mendeteksi noise pernyataan kebutuhan agar dapat meningkatkan relevansi spesifikasi kebutuhan terhadap permasalahan yang dihadapi di dalam dokumen SKPL.

1. Kontribusi Penelitian

Kontribusi penelitian ini adalah penggunaan metode klasterisasi spektral dalam mendeteksi *noise* pada dokumen SKPL.

1. Batasan Masalah
2. Data dokumen SKPL yang digunakan berbahasa inggris.
3. Pernyataan kebutuhan dinyatakan dalam bahasa alamiah.

**BAB 2  
KAJIAN PUSTAKA**

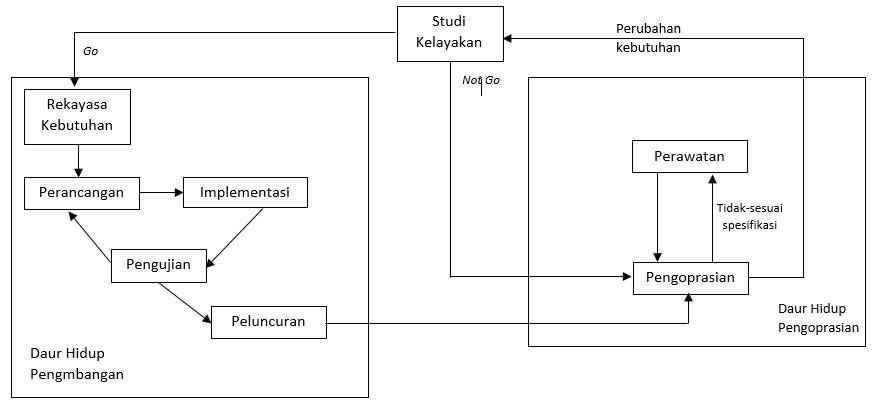
Pada bab ini akan dijelaskan tentang pustaka yang terkait dengan landasan penelitian. Pustaka yang terkait dianalisa dan dirangkum dalam bentuk studi komparasi.

1. Kebutuhan Perangkat Lunak

Kebutuhan dalam perangkat lunak dapat diartikan sebagai sekumpulan deskripsi tentang bagaimana suatu sistem berperilaku sehingga dapat memiliki nilai guna bagi penggunanya [3]. Kebutuhan tersebut dispesifikasikan dalam tahap rekayasa kebutuhan.

Daur Hidup Perangkat Lunak dapat dilihat pada Gambar 2.1 yang menunjukkan bahwa dalam kebutuhan perangkat lunak terdapat dua siklus kehidupan utama yaitu daur hidup pengembangan dan daur hidup pengoperasian perangkat lunak, yang dihubungkan oleh studi kelayakan dan proses peluncuran.

Terdapat dua siklus kehidupan dari suatu perangkat lunak, yaitu daur hidup pengembangan dan daur hidup pengoperasian. Dalam membuat sebuah aplikasi kita harus mengetahui kebutuhan dari aplikasi tersebut. Setelah mengetahui kebutuhannya, kita dapat merancang aplikasi tersebut. Dari perancangan kita dapat mengimplementasikan. Hasil dari implementasi selanjutnya diuji sampai benar-benar bisa digunakan sebelum pada akhirnya dilucurkan. Setelah daur hidup pengembangan dilakukan sampai peluncuran, kita masuk ke daur hidup pengoperasian yaitu pengoperasian dan perawatan [3].



Gambar .1 Daur Hidup Perangkat Lunak

Proses rekayasa kebutuhan dilakukan dalam beberapa tahap seperti yang ditunjukkan pada . Proses dimulai dengan mempelajari tentang ranah lingkup permasalahan pada ranah pemahaman. Dari pemahaman ranah tersebut dikumpulkan kebutuhan perangkat lunak dan masuk pada kalsifikasi dan diakhiri dengan pemeriksaan kembali kebutuhan – kebutuhan yang telah didapatkan terhadap relevansi dengan permasalahan utama. Kebutuhan – kebutuhan perangkat lunak yang telah berhasil dikumpulkan dirumuskan dalam sebuah dokumen SKPL [3].



Gambar .2 Tahapan Rekayasa Kebutuhan

1. Kebutuhan Pengguna

Kebutuhan pengguna adalah pernyataan atau gambaran pelayanan yang disediakan oleh sistem dan batasan – batasan bagi sistem yang akan dibangun agar pengguna dapat memenuhi tujuan penggunaan sistem.

Tabel 2.1. Contoh Kebutuhan Sistem Serta Kebutuhan Penggunaanya

|  |  |
| --- | --- |
| Sistem | Kebutuhan Pengguna |
| SIM Akademik | * Mahasiswa hendak menyusun rencana kuliah semester ini * Orangtua hendak melihat prestasi belajar anak didik |

Secara umum, kebutuhan perangkat lunak dapat dibagi ke dalam dua jenis yaitu : fungsional dan non-fungsional. Kebutuhan fungsional adalah kebutuhan yang dilihat dari sudut pandang pengguna. Kebutuhan non-fungsional adalah kebutuhan yang dilihat dari sudut pandang sistem. Selain dapat dikelompokkan ke dalam kebutuhan fungsional dan non-funsional, kebutuhan juga dapat dikelompokkan lebih rinci lagi berdasarkan tingkat dari stakeholder [3].

1. Dokumen SKPL (Spesifikasi Kebutuhan Perangkat Lunak)

Dokumen SKPL merupakan keluaran dari tahap rekayasa kebutuhan di dalam Daur Hidup Perangkat Lunak seperti pada . Dokumen tersebut berisi tentang kumpulan kebutuhan-kebutuhan perangkat lunak yang akan dibuat. Dokumen SKPL (Spesifikasi Kebutuhan Perangkat Lunak) berfungsi sebagai landasan proses pengembangan perangkat lunak yang telah disetujui oleh pihak pengembang dan klien.

Terdapat empat pendekatan dalam menyusun dokumen SKPL (Spesifikasi Kebutuhan Perangat Lunak) yaitu :

1. Bahasa Alamiah

Dokumen SKPL disusun dengan menggunakan bahasa sehari -hari.

1. Bahasa Alamiah Terstruktur

Dokumen SKPL disusun dengan menggunakan bahasa sehari – hari yang ditata sesuai dengan struktur tertentu.

1. Bahasa Semi-formal

Dokumen SKPL disusun dengan menggunakan bahasa grafikal yang dilengkapi dengan penjelasan teks.

1. Bahasa Formal

Dokumen SKPL disusun sepenuhnya dengan notasi – notasi matematika seperti *finite state machine*.

Menurut survei yang telah dilakukan sebelumnya, pendekatan bahasa alamiah paling banyak digunakan untuk menyusun dokumen SKPL sebesar 71,8% [4]. Hal tersebut disebabkan karena pendekatan bahasa alamiah lebih mudah dipahami dan diimplementasi. Namun karena sifatnya yang subjektif, penggunaan bahasa alamiah dapat menimbulkan kerancuan dan kesalahan-kesalahan lainnya yang telah dibahas oleh Meyer dalam “*The seven sins of specifier”* [2].

Mayer mendeskrisikan secara mendalam tujuh kesalahan umum yang sering dilakukan pengembang perangkat lunak sebagai *“The seven sins of specifier”* salah satunya adalah noise, noise dapat muncul pada proses spesifikasi kebutuhan yaitu menyertakan suatu informasi yang tidak relevan dengan keseluruhan kebutuhan perangkat lunak.

1. *Noise* pada Dokumen SKPL

Spesifikasi dalam rekayasa perangkat lunak adalah suatu aktivitas untuk mengumpulkan kebutuhan dalam pembuatan perangkat lunak. Kumpulan kebutuhan tersebut dirumuskan dalam dokumen SKPL (Spesifikasi Kebutuhan Perangkat Lunak). Namun terkadang terdapat suatu kebutuhan yang tidak relevan dengan kebutuhan lainnya. Kebutuhan tersebut dapat mengurangi kualitas informasi yang diberikan oleh dokumen SKPL dan dapat dianggap sebagai *noise*, seperti yang telah dijelaskan oleh Meyer [2].

Tabel .2 Contoh Kebutuhan Perangkat Lunak

|  |  |  |
| --- | --- | --- |
| **ID** | **Requirement Statement** | **Noise** |
| R-01 | Submit jobs with the associated deadline, cost, and execution time | No |
| R-02 | Query the cluster to establish the current cost per unit time for submitting new jobs | No |
| R-03 | Monitor the status of submitted jobs | No |
| R-04 | cancel jobs submitted by him | No |
| R-05 | Requirements engineer can remove a document | Yes |

Terlihat pada .2 contoh data yang akan digunakan untuk menguji pendeteksi *noise* pada dokumen SKPL yang akan dibuat. Tabel tersebut berisi tentang kebutuhan sebuah perangkat lunak. Kebutuhan R-05 tidak relevan dengan kebutuhan lainnya, karena tidak berhubungan dengan manajemen pekerjaan. Penelitian yang akan dilakukan bertujuan untuk dapat mendeteksi kebutuhan tersebut secara otomatis.

**2.5 Pengolahan Bahasa Alamiah**

Salah satu teknik pemrosesan bahasa alami yang seringkali digunakan pada pra proses sebagai berikut:

1. *Tokenisasi*

Pada proses *tokenisasi*, teks dokumen dipecah menjadi bagian-bagian berupa kalimat, paragraf, atau dokumen

1. *Lowercase*

Merubah semua huruf menjadi huruf kecil

1. *Stopword Removal*

Proses untuk menghilangkan angka, simbol, dan kata ganti

1. *Stemming*

Menghilangkan imbuhan dari kata-kata yang berimbuhan sehingga, hanya ada kata-kata yang penting yang tersisa dalam teks.

* 1. Klasterisasi

Klasterisasi (*clustering*) adalah pengelompokkan data ke dalam sejumlah kelompok berdasarkan kesamaan karakteristik masing-masing data pada kelompok-kelompok yang ada. Sehingga kelompok dengan karakteristik yang sama (memiliki hubungan) akan berada pada satu kelompok yang sama. Sedangkan kelompok yang memiliki karakteristik berbeda (tidak ada hubungan) akan dikelompokkan dalam kelompok lain (*cluster* lain). Klasterisasi juga dikelompokkan sebagai metode pembelajaran tidak terbimbimg (*unsupervised learning)* [5].

Klasterisasi (c*lustering)* bertujuan untuk mengelompokkan data dengan karakteristik yang sama ke suatu kelompok yang sama dan data dengan karakteristik yang berbeda ke kelompok yang lain. Pada umumnya terdapat dua pendekatan *clustering* yaitu pendekatan partisi dan pendekatan hirarki. *Clustering* dengan pendekatan partisi merupakan pengelompokan data dari satu kelompok besar kemudian dibagi menjadi beberapa kelompok yang lebih kecil. Contoh metode *clustering* dengan pendekatan partisi adalah *K-Means Clustering*. *Clustering* dengan pendekatan hirarki atau sering disebut dengan *Hierarchical Clustering* mengelompokkan data dengan menggabungkan masing-masing *record* atau individu pada data menjadi *cluster-cluster*. Contoh metode *clustering* dengan pendekatan hirarki adalah *Agglomerative Hierarchical Clustering*[5].

**2.7** Kakas Bantu Pemrosesan Bahasa Alamiah

1. Cosine Similarity

Cosine similarity merupakan salah satu metode pengukuran kemiripan antara dua dokumen atau teks. Pada cosine similarity dokumen atau teks dianggap sebagai vektor. Rumus cosine similarity [6]:

(1. 1)

Di mana Ri dan Rj adalah vektor term *frequency* dari dokumen, dengan nilai *cosine similarity* berada pada *range* 0-1.

1. Koofisien *Kappa*

Koofisien *Kappa* berfungsi untuk menghitung tingkat persetujuan antara dua penilai dalam memberikan penilaian pada suatu kasus [7]. Koofisien *Kappa* memperhitungkan probabilitas terjadinya persetujuan dan penilaian di antara kedua penilai. Nilai koofisien *Kappa* selalu diantara 0 dan 1, dengan nilai 1 menunjukan kesempurnaan tingkat persetujuan.

Tabel 2.3 Masukan Perhitungan Nilai Koofisien *Kappa*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Penilai 2 | |  |
|  | Nilai | A | B | **Total** |
| Penilai 1 | A | n\_aa | n\_ab | n\_a\* |
| B | n\_ba | n\_bb | n\_b\* |
|  | **Total** | n\_\*a | n\_\*b |  |

Pada Tabel 2.2 dapat dilihat bahwa terdapat 2 penilai yaitu *Penilai 1* dan *Penilai 2*. Setiap penilai dapat memberikan dua kategori penilaian yaitu nilai *a* atau nilai *b* pada suatu kasus. Elemen dari Tabel 2.2 merupakan jumlah kejadian *Penilai 1* dan *Penilai 2* memberikan penilaian suatu kasus. Penjelasan variabel pada elemen Tabel 2.2 adalah sebagai berikut :

* *n\_aa* adalah jumlah kejadian *Penilai 1* dan *Penilai 2* memberikan nilai *a* pada suatu kasus.
* *n \_ab* adalah jumlah kejadian *Penilai 1* memberikan nilai *a* dan *Penilai 2* memberikan nilai *b* pada suatu kasus.
* *n \_ba* adalah jumlah kejadian *Penilai 1* memberikan nilai *b* dan *Penilai 2* memberikan nilai *a* pada suatu kasus.
* *n \_bb* adalah jumlah kejadian *Penilai 1* dan *Penilai 2* memberikan nilai *b* pada suatu kasus.

Nilai koofisien *Kappa k* dapat dihitung jika jumlah kejadian penilaian *Penilai 1* dan *Penilai 2* telah diketahui. Pertama – tama probabilitas persetujuan yang telah diobservasi antara kedua penilai, atau dihitung dengan Persamaan (2.1).

(2.1)

Lalu probabilitas kedua penilai memberikan nilai *a* secara acak, atau , dihitung, begitu juga dengan nilai dengan Persamaan (2.2) dan (2.3). Sehingga nilai probabilitas persetujuan secara acak, atau dapat dihitung dengan Persamaan (2.4).

(2.2)

(2.3)

(2.4)

Nilai koofisien *Kappa k* dapat dihitung dengan Persamaan

(2.5)

# BAB 3 METODOLOGI PENELITIAN

Bab ini akan memaparkan tentang metodologi penelitian yang digunakan pada penelitian ini, yang terdiri dari (1) studi literatur, (2) desain dan implementasi, (3) pengujian, dan (4) dokumentasi dan pembuatan laporan. Ilustrasi alur metodologi penelitian dapat dilihat pada .



Gambar .1 Alur Metodologi Penelitian

1. Studi Literatur

Tahap studi literatur bertujuan untuk mengumpulkan referensi - referensi yang dapat menunjang penelitian. Sumber referensi dapat berupa jurnal ilmiah atau buku teks. *Noise* pada dokumen SKPL dapat dianggap juga sebagai *outlier* pada sebuah dokumen teks. Sehingga referensi yang dikumpulkan berhubungan dengan *noise* di dalam dokumen SKPL dan metode pendeteksi *outlier* pada dokumen teks. Referensi tersebut digunakan untuk merumuskan permasalahan yang menjadi landasan dilakukannya penelitian ini dan solusi yang akan diusulkan. Berdasarkan studi literatur\ yang telah dilakukan, informasi yang berkaitan dengan penilitian yang dilakukan ini, seperti berikut :

1. *Noise* merupakan salah satu kesalahan yang sering dilakukan oleh analis kebutuhan dalam menyusun dokumen SKPL.
2. *Noise* adalah suatu elemen pada dokumen SKPL yang tidak relevan dengan informasi lainnya.
3. *Noise* dapat dianggap sebagai *outlier* pada dokumen teks.
4. Metode klasterisasi dapat digunakan sebagai pendeteksi *noise* pada dokumen SKPL.
5. Desain dan Implementasi

Tahap ini bertujuan untuk merancang dan mengimplementasi solusi yang diusulkan untuk mengatasi permasalahan yang telah dirumuskan sebelumnya. Metode yang akan diusulkan memiliki alur proses seperti pada .



Gambar .2 Alur Proses Metode

**3.2.1. Kebutuhan Perangkat Lunak di Dokumen SKPL**

Masukan dari metode deteksi *noise* yang akan dibuat adalah kebutuhan – kebutuhan perangkat lunak yang tertera pada dokumen SKPL. Kebutuhan tersebut diekstraksi secara manual dari dokumen SKPL. Setiap dokumen SKPL akan menghasilkan sebuah set kebutuhan perangkat lunak.

Setiap kebutuhan yang telah diekstraksi akan diberikan label berupa nilai *boolean* yaitu 1 atau 0. Kebutuhan akan berlabel 1 jika kebutuhan tersebut dianggap sebagai *noise*, sisanya akan berlabel 0. Pemberian label akan dilakukan secara manual oleh penulis. Contoh kebutuhan yang telah diekstraksi dari dokumen SKPL dan labelnya dapat dilihat pada Tabel 2.2.

**3.2.2 Pre-Processing Text**

Pre-Processing Text atau Pra-pemrosesan teks akan diimplementasi pada setiap kebutuhan perangkat lunak. Detil pra-pemrosesan teks telah dijelaskan pada sub-bab 2.4.



Gambar .3 Alur Diagram Pra-pemrosesan Teks

Pada tahap pra-pemrosesan teks terdapat tiga proses utama, berikut adalah Contoh hasil pra-pemrosesan teks pada kebutuhan perangkat lunak :

1. *Tokenisasi*

Tokenisasi mempunyai peranan untuk memecah teks menjadi bagian unit terkecil dalam pemrosesan bahasa atau disebut dengan potongan kata. Dokumen teks

R1 : Submit jobs with the associated deadline cost and execution time

R2 : Query the cluster to establish the current cost per unit time for submitting new job

R3 : Monitor the status of submitted jobs

R4 : Cancel job submitted by him

R5 : Requirements engineer can remove a document

Akan dipecah menjadi :

R1 : Submit// jobs// with// the// associated// deadline// ,// cost// and// execution //time//

R2 : Query// the// cluster// to// establish// the// current// cost// per// unit// time// for// submitting// new// job//

R3 : Monitor// the// status// of// submitted// jobs//

R4 : Cancel// job// submitted// by// him//

R5 : Requirements// engineer// can// remove// a// document//

1. *Lowercase* *(*Konversi menjadi huruf kecil)

Merubah semua huruf pada teks dokumen menjadi huruf kecil. Pada teks dokumen sebelumnya

R1 : Submit// jobs// with// the// associated// deadline// ,// cost// and// execution //time//

R2 : Query// the// cluster// to// establish// the// current// cost// per// unit// time// for// submitting// new// job//

R3 : Monitor// the// status// of// submitted// jobs//

R4 : Cancel// job// submitted// by// him//

R5 : Requirements// engineer// can// remove// a// document//

Akan berubah menjadi

R1 : submit// jobs// with// the// associated// deadline// ,// cost// and// execution //time//

R2 : query// the// cluster// to// establish// the// current// cost// per// unit// time// for// submitting// new// job//

R3 : monitor// the// status// of// submitted// jobs//

R4 : cancel// job// submitted// by// him//

R5 : requirements// engineer// can// remove// a// document//

1. *Stopword Removal* (Eliminasi angka, simbol, dan kata ganti)

Proses ini berfungsi untuk menghilangkan angka simbol dan kata ganti. Dokumen sebelumnya

R1 : submit// jobs// with// the// associated// deadline// ,// cost// and// execution //time//

R2 : query// the// cluster// to// establish// the// current// cost// per// unit// time// for// submitting// new// job//

R3 : monitor// the// status// of// submitted// jobs//

R4 : cancel// job// submitted// by// him//

R5 : requirements// engineer// can// remove// a// document//

akan berubah menjadi

R1 : submit// jobs// associated// deadline// cost// execution //time//

R2 : query// cluster// establish// current// cost// per// unit// time// submitting// new// job//

R3 : monitor// status// submitted// jobs//

R4 : cancel// job// submitted//

R5 : requirements// engineer// remove// document//

1. *Stemming*

*Stemming* memiliki peran untuk menghilangkan imbuhan dari kata-kata yang berimbuhan menjadi kata dasar. Dokumen sebelumnya

R1 : submit// jobs// associated// deadline// cost// execution //time//

R2 : query// cluster// establish// current// cost// per// unit// time// submitting// new// job//

R3 : monitor// status// submitted// jobs//

R4 : cancel// job// submitted//

R5 : requirements// engineer// remove// document//

akan berubah menjadi

R1 : submit// job// associate// deadline// cost// execution //time//

R2 : query// cluster// establish// current// cost// per// unit// time// submit// new// job//

R3 : monitor// status// submit// job//

R4 : cancel// job// submit//

R5 : requirement// engineer// remove// document//

**3.2.3 Spectral Clustering**

Pada metode *spectral clustering*, pengelompokan data didasarkan atas kesamaan antara setiap data. Kesamaan tersebut dilihat dari keterkaitan antara setiap data, dimana *verteks* dari *graph* tersebut merupakan setiap *record* pada data [8].



Gambar .4 Diagram Alir Algoritma *Spectral Clustering*

Langkah-langkah dalam melakukan Algoritma *Spectral Clustering* [9][9][9][9][9][9][9][9] sebagai berikut :

* + - 1. Menyediakan kebutuhan yang telah di pre-processing.

Contoh kebutuhan :

R1 : submit job associate deadline cost execution time

R2 : query cluster establish current cost per unit time submit new job

R3 : monitor status submit job

R4 : cancel job submit

R5 : requirement engineer remove document

* + - 1. Membentuk *graph* tak berarah G = {S,A}. Di mana *S* merupakan himpunan *node*. *Node* adalah kalimat dalam bentuk fitur vektor dengan ukuran m, dimana m adalah jumlah kata unik. Sedangkan n adalah jumlah kalimat dalam data set dan A adalah nilai bobot edge yang menghubungkan kalimat yang satu dengan yang lain [10].

Tabel 3.1. Matriks S

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **submit** | **job** | **associate** | **Deadline** | **cost** | **Execution** | **time** | **query** | **cluster** | **establish** |
| R1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| R2 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| R3 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R4 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **current** | **per** | **unit** | **new** | **Monitor** | **status** | **cancel** | **requirement** | **engineer** | **remove** | **document** |
| R1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R2 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R3 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| R4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

* + - 1. Konstruksi matriks *affinity* A [] merupakan matriks bobot dari semua *edge* yang menghubungkan satu *node* dengan *node* yang lain pada *graph* G. Nilai bobot dari Aij dihitung berdasarkan nilai kemiripan antara kalimat dan . Perhitungan kemiripan ini didasarkan pada rumus *cosine similarity*.

(3.1)

Tabel 3.2 Contoh salah satu nilai kemiripan untuk kebutuhan R1-R2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **R1** | **R2** | R1.R2 | kuadrat(^R1) | kuadrat(^R2) |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
|  | | 4 | 7 | 11 |
| akar jml R1^2 |  | 2,65 |  |  |
| akar jml R2^2 |  | 3,32 |  |  |
| perkalian akar |  | 8,77 |  |  |
| (jml R1^2+jml R2^2) |  |  |  |
| hasil |  | 0,46 |  |  |

Pada nilai kemiripan R3 dan sterusnya dilakukan perhitungan yang sama, hasil hitungan dari *cosine similarity* yang sudah dihitung pada tahap sebelumnya direpresentasikan dalam bentuk matriks seperti yang ditampilkan pada tabel 3.3

Tabel 3.3 Matriks A (*Affinity*)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Aij** | **R1** | **R2** | **R3** | **R4** | **R5** |
| **R1** |  | 0,46 | 0,38 | 0,44 | 0,00 |
| **R2** | 0,46 |  | 0,30 | 0,35 | 0,00 |
| **R3** | 0,38 | 0,30 |  | 0,58 | 0,00 |
| **R4** | 0,44 | 0,35 | 0,58 |  | 0,00 |
| **R5** | 0,00 | 0,00 | 0,00 | 0,00 |  |

* + - 1. Langkah berikutnya adalah membentuk matriks *laplacian.*

Langkah pertama dalam membangun matriks Laplacian adalah :

* Menghitung matriks *diagonal* (*D*). Matriks diagonal (D) dibangun dari matriks *affinity (A)* yang sudah dihitung sebelumnya. Perhitungan matriks D (*diagonal*) dapat dihitung dengan rumus :

(3.2)

Tabel 3.4 Matriks D (*diagonal)*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **D** | **R1** | **R2** | **R3** | **R4** | **R5** |
| **R1** | 1,28 | 0,00 | 0,00 | 0,00 | 0,00 |
| **R2** | 0,00 | 1,11 | 0,00 | 0,00 | 0,00 |
| **R3** | 0,00 | 0,00 | 1,26 | 0,00 | 0,00 |
| **R4** | 0,00 | 0,00 | 0,00 | 1,37 | 0,00 |
| **R5** | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |

Contoh hitungan

Nilai diagonal R1 dan R1 nilainya 1,28 diperoleh dari hasil penjumlahan nilai dari kebutuhan R1-R5 pada matriks *affinity*. Perhitungan dengan cara yang sama dilakukan pada kebutuhan yang lain.

* Menghitung nilai matriks *laplacian.* Nilai matriks *laplacian* diperoleh dari pengurangan nilai pada matriks D ( *diagonal* ) dan matriks A *(affinity*) dengan rumus: (3.3)

Tabel 3.5 Matriks L (*laplacian*)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **L** | **R1** | **R2** | **R3** | **R4** | **R5** |
| **R1** | 1,28 | **0,82** | 0,90 | 0,84 | 1,28 |
| **R2** | 0,65 | 1,11 | 0,81 | 0,76 | 1,11 |
| **R3** | 0,88 | 0,96 | 1,26 | 0,68 | 1,26 |
| **R4** | 0,93 | 1,02 | 0,79 | 1,37 | 1,37 |
| **R5** | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |

Contoh hitungan

Nilai *laplacian* R1 dan R2 = 0,82 diperoleh dari nilai baris matriks D (*diagonal*) R1- nilai kolom matriks A (*affinity*) R2. Perhitungan dengan cara yang sama dilakukan untuk kebutuhan yang lain.

* + - 1. Membentuk normalisasi matriks *laplacian.* Nilai normalisasi matriks *laplacian* dibangun dari matriks bobot *diagonal* (D) yang sudah dihitung sebelumnya dengan rumus berikut [9] :

(3.4)

Jika i = j dan di ≠ 0 maka nilainya 1

Jika i ≠ j dan di bertetangga dengan dj

Jika tidak keduanya atau sebalik.

Tabel 3.6 Matriks Normalisasi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lsym** | **R1** | **R2** | **R3** | **R4** | **R5** |
| **R1** | 1,00 | -0,65 | -0,63 | -0,61 | -0,88 |
| **R2** | -0,65 | 1,00 | -0,65 | -0,64 | -0,95 |
| **R3** | -0,63 | -0,65 | 1,00 | -0,89 | -0,89 |
| **R4** | -0,61 | -0,64 | -0,89 | 1,00 | -0,88 |
| **R5** | -0,88 | -0,95 | -0,89 | -0,88 | 0,00 |

Contoh hitungan:

Nilai normalisasi *laplacian* kebutuhan R1 dan R2 = -0,65 diperoleh dari -1/ nilai kuadrat dari matriks *diaogonal* R1+nilai kuadrat dari matiks *diagonal* R2. Perhitungan dengan cara yang sama dilakukan pada kebutuhan yang lain. Setelah mendapat nilai dari setiap kebutuhan dibentuklah ke dalam matriks normalisasi seperti pada tabel 3.6

* + - 1. Konstruksi matriks U. Matriks U dibangun dengan menghitung *eigen* *vector* dan *eigen value* dari hasil matriks normalisasi *laplacian*. Maka terbentuklah matriks U = {u1,u2,...uk}, seperti pada tabel 3.7.

Tabel 3.7 Matrik U

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **U** | **U1** | **U2** | **U3** | **U4** | **U5** |
| **1** | 0,38 | 0,08 | -0,74 | -0,55 | 0,03 |
| **2** | 0,40 | 0,00 | -0,40 | 0,83 | 0,00 |
| **3** | 0,00 | 0,48 | 0,28 | -0,06 | -0,72 |
| **4** | 0,41 | 0,49 | 0,32 | -0,04 | 0,70 |
| **5** | 0,60 | -0,72 | 0,33 | -0,13 | 0,00 |

* + - 1. Langkah berikut adalah normalisasi data dengan nilai matriks U sehingga membentuk *k* kolom yang merepresentasikan setiap matriks normalisasi u pada setiap kolom. Perhitungan nilai normalisasi u didasarkan pada rumus [9]:

(3.5)

Dari hitungan yang didasrkan pada rumus tersebut akan membentuk matriks *t.*

Tabel 3.8 Matriks T

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **T** | **1** | **2** | **3** | **4** | **5** |
| **y1** | 0,38 | 0,08 | -0,74 | -0,55 | 0,03 |
| **y2** | 0,40 | 0,00 | -0,40 | 0,83 | 0,00 |
| **y3** | 0,00 | 0,53 | 0,31 | -0,07 | -0,79 |
| **y4** | 0,41 | 0,49 | 0,32 | -0,04 | 0,70 |
| **y5** | 0,60 | -0,72 | 0,33 | -0,13 | 0,00 |

Contoh hitungan:

Nilai 0,38 diperoleh dari nilai y1/kuadrat dari nilai normalisasi u1,u2,u3,u4 dan u5, perhitungan dengan cara yang sama dilakukan untuk kebutuhan yang lainnya. Seperti pada tabel 3.9.

* + - 1. Menentukan nilai *k (Cluster)*

Dari perhitungan normalisasi matriks U yang telah dihitung sebelumnya, dapat menentukan nilai *k* yang kemudian akan *dicluster* dengan *k-mens clustering.* Perhitungan nilai k didasarkan pada rumus [10] :

*k =argmax*{ƛk+1(L)-ƛk(L) (3.6)

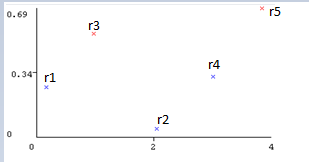
Tabel 3.9 Menentukan nilai *k*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| y1 | y2 | y3 | y4 | y5 |
| -2,39 | 1,31 | 1,54 | 1,66 | 1,89 |
|  |  |  |  |  |
| -3,70 | C1 |  |  |  |
| -0,23 | C2 |  |  |  |
| -0,12 | C3 |  |  |  |
| -0,23 | C4 |  |  |  |
| berdasarkan rumus tersebut dapat terdapat tiga cluster. | | | | |
| nilai cluster yang besar sebagai nilai k = 4,00 | | | | |

Contoh hitungan :

Nilai -3,70 diperoleh dari (-2,39)-1,31= -3,70

* + - 1. Hasil dari data normalisasi matriks U kemudian di-*cluster* dengan K-*means* *clustering* dimana jarak rata-rata nilai terkecil dianggap sebagai noise.



Gambar 3.5 Visualisasi Hasil Identifikasi Noise

Dari hasil visualisasi yang diperoleh dapat disimpulkan bahwa kebutuhan R5 dianggap noise.

**3.2.4. Identifikasi *Pencilan***

Hasil dari klasterisasi spektral adalah beberapa klaster yang berisi kebutuhan perangkat lunak. Klaster dengan jumlah anggota paling kecil akan dianggap sebagai *noise*.

Implementasi metode akan menggunakan bahasa pemrograman Python versi 2.7. Pustaka - pustaka pendukung yang akan digunakan dalam implementasi metode adalah pustaka numpy, scipy, scikit-learn, panda dan NLTK.

1. Pengujian

Data uji diperoleh dari dataset yang ada berupa kebutuhan-kebutuhan di dalam sebuah dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL). Sebelumnya, kebutuhan – kebutuhan tersebut diberi label berupa nilai *boelean* 1 atau 0. Label 1 menandakan bahwa kebutuhan tersebut dianggap *noise* oleh penulis. Pelabelan data uji tersebut merepresentasikan penilaian *noise* oleh penulis.

Pada pengujian, data uji seperti pada .2 digunakan sebagai masukan untuk pendeteksi *noise*. Pendeteksi *noise* akan menghasilkan prediksi kebutuhan yang dianggap sebagai *noise*. Kedua hasil *noise* dari pendeteksi *noise* dan penilaian penulis akan dibandingkan menggunakan kakas bantu yaitu menghitung nilai indeks koofisien *Kappa*, seperti yang telah dijelaskan pada subbab 2.5. Nilai dari koofisien *Kappa* dapat merepresentasikan tingkat konsistensi hasil *noise* yang diberikan oleh pendeteksi *noise* dengan penilaian penulis.

Adapun skenario pengujian uji coba yang dilakukan sebagai berikut :

1. Mengekstraksi pernyataan kebutuhan dari setiap dokumen SKPL yang ada ke dalam format csv, seperti berikut:

KODE; PERNYATAAN\_KEBUTUHAN;LABEL

Setiap SKPL akan disimpan ke dalam satu buah berkas csv. Setiap berkas csv merupakan satu buah data set pengujian. Berikut contoh berkas csv dari sebuah SKPL.

R01; Operator dapat membuka berkas.; 0

R02; Mahasiswa dapat memilih mata kuliah yang hendak diambil.; 0

R03; Pemegang kartu dapat melihat saldo tabungannya saat ini.; 0

NR01; Sistem melakukan backup data minimal setiap 1 minggu sekali.; 0

1. Memberi label pada setiap data set. Proses pelabelan ini melibatkan ahli dalam bidang rekayasa perangkat lunak. Untuk setiap pernyataan dalam sebuah data set, ahli akan diminta untuk memberikan label 0 jika pernyataan kebutuhan tersebut bukan noise, dan label 1 pernyataan kebutuhan tersebut noise. Ada tiga orang ahli yang akan dilibatkan dalam pelabelan pernyataan kebutuhan ini. Dari setiap data set, maka akan dibentuk data set dengan dua kesepakatan dan data set dengan tiga kesepakatan. Data set dengan dua kesepakatan adalah data set dengan label akhir merupakan label mayoritas diantara label yang diberikan oleh ketiga ahli. Data set dengan tiga kesepakatan adalah data set dengan aturan sebagai berikut: (a) jika salah satu ahli memberi label 0, maka label akhir dari pernyataan kebutuhan tersebut adalah 0, dan (b) jika semua ahli memberi label 1, maka label akhir dari pernyataan kebutuhan tersebut adalah 1.
2. Melakukan klasterisasi menggunakan metode yang diusulkan terhadap data set yang disediakan. Poses klasterisasi ini akan dilakukan sebanyak 10 kali pada setiap data set, dengan nilai ambang batas (*threshold*) kemiripan kata dari 0.0 sampai dengan 0.9. Dari hasil tersebut akan dihitung nilai kesepakatan dengan hasil dari para ahli. Perhitungan kesepakatan ini akan menggunakan kappa [7]. Dari hasil pengujian ini, akan ditentukan nilai ambang batas paling optimal (to) untuk menghasilkan kesepakatan terbaik. Dari hasil ini, maka akan diuji kembali sebanyak 10 kali dengan nilai ambang batas gradual mulai dai to-0.05 sampai dengan to+0.05. hasil ini kemudian diolah lagi untuk menentukan nilai ambang batas paling optimal (to) terbaru. Nilai ini akan menjadi nilai ambang batas kemiripan kata terbaik untuk pendeteksian noise menggunakan metode yang diusulkan.
3. Dokumentasi dan Jadwal Penelitian

Pada tahap dokumentasi sistem ini akan dilakukan penulisan laporan hasil penelitian dari setiap tahapan yang dilakukan. Tujuan dari tahapan ini adalah menghasilkan dokumentasi tertulis dari penelitian yang dilakukan. Jadwal penelitian yang dilakukan dapat dilihat pada Tabel 3.9

Tabel .10 Jadwal Kegiatan Penelitian

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Kegiatan | Bulan ke-1 | | | | Bulan ke-2 | | | | Bulan ke-3 | | | | Bulan ke-4 | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1. | Studi Literature |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2. | Analisis dan Desain |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3. | Implementasi Perangkat Lunak |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4. | Uji Coba dan Analisis |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5. | Penulisan Laporan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**DAFTAR PUSTAKA**

[1] Sommerville, *Software Enginnering, 8th Edition, Addison Wesley, Boston*. 2007.

[2] B. Meyer, “on Formalism in Specifications.,” *IEEE Softw.*, vol. 2, no. 1, pp. 6–26, 1985.

[3] D. Siahaan, *Analisis Kebutuhan dalam Rekayasa Perangkat Lunak*. 2012.

[4] A. Rossi, “Incentives in managerial compensation: a survey of experimental research,” *ROCK Work. Pap.*, 1999.

[5] E. Prasetyo, *Data Mining: Mengolah Data Menjadi Informasi dengan Matlab*. 2014.

[6] © E Garcia and A. Co, “Cosine Similarity Tutorial,” pp. 4–10, 2015.

[7] J. Cohen, “A Coefficient of Agreement for Nominal Scales,” *Educ. Psychol. Meas.*, vol. 3, 1960.

[8]

A. Yusuf and H. Tjandrasa, “Prediksi Nilai Dengan Metode Spectral Clustering Dan Clusterwise Regression,” *SimanteC*, vol. 4, no. 1, p. 8, 2014.

[9] U. Von Luxburg, “A Tutorial on Spectral Clustering,” pp. 1–32, 2007.

[10] X. Cai, R. Zhang, D. Gao, and W. Li, “Simultaneous Clustering and Noise Detection for Theme-based Summarization,” *Proc. 5th Int. Jt. Conf. Nat. Lang. Process.*, pp. 491–499, 2011.