PROPOSAL PENELITIAN

Automatic thesaurus construction for noise detection in requirement document

Ahmad Mustofa
NRP. 5116201054

DOSEN PEMBIMBING

Daniel Oranova Siahaan, S.Kom, M.Sc, P.D.Eng

NIP. 197411232006041001

PROGRAM MAGISTER

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2015

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
PROPOSAL TESIS

Judul :

Oleh :

NRP :

Telah diseminarkan pada:

Hari :

Tanggal :

Tempat :

Mengetahui / menyetujui

Dosen Penguji:	Dosen Pembimbing:
1.	1.
_____	
2.	2.
_____	
3.	
_____	
NIP.	

[Halaman ini sengaja dikosongkan]

Nama Mahasiswa :

NRP :

Pembimbing :

ABSTRAK

Kata kunci:

[Halaman ini sengaja dikosongkan]

Nama Mahasiswa :

NRP :

Pembimbing :

ABSTRACT

***Keywords**:*

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Tahap spesifikasi kebutuhan adalah tahap pertama yang dilakukan dalam proses pengembangan perangkat lunak. Sehingga jika terjadi kesalahan pada tahap ini, secara otomatis akan terjadi kesalahan pada tahap-tahap selanjutnya. Sudah banyak peneliti yang berusaha menemukan cara untuk mendeteksi kesalahan pada proses rekayasa kebutuhan. Bertrand meyer mengelompokkan kesalahan dalam spesifikasi kebutuhan menjadi tujuh kelompok yang kemudian dikenal dengan istilah *Meyer's seven sins* [1]. *Noise* adalah salah satu kesalahan dalam *Meyer's seven sins* yang disebabkan oleh adanya suatu elemen dalam teks yang memberikan informasi yang tidak relevan dengan domain masalah yang hendak diselesaikan [1]. Di sisi lain, perkembangan yang pesat pada bidang kecerdasan buatan telah mampu melakukan klasifikasi teks secara otomatis. Dalam klasifikasi teks, dibutuhkan sebuah metode untuk melakukan pembobotan pada data teks yang digunakan. Term frequency – invers document frequency (tf-idf) merupakan metode yang sering digunakan dalam pembobotan data teks [2]. Dalam tf-idf setiap term / kata yang unik akan diberikan bobot masing-masing tanpa memperhatikan unsur kesamaan antar kata. Untuk mengatasi kekurangan ini, Hao et al mengusulkan metode kontruksi thesaurus secara otomatis untuk menemukan kata-kata yang bermakna mirip. Metode ini terbukti menghasilkan performa yang baik dalam melakukan klasifikasi email spam [3]. Atas dasar itulah kami mengusulkan penggunaan metode *automatic thesaurus construction* yang dikombinasikan dengan metode tf-idf untuk sistem pendeteksi noise pada dokumen spesifikasi kebutuhan perangkat lunak.

1.2 Perumusan Masalah

Rumusan masalah yang diangkat dalam penelitian ini adalah:

1. Bagaimana mengaplikasikan *automatic thesaurus construction* pada data dokumen spesifikasi kebutuhan perangkat lunak?
2. Bagaimana mengekstraksi fitur-fitur dari dokumen spesifikasi kebutuhan perangkat lunak?

3. Bagaimana membedakan kebutuhan yang termasuk noise dengan yang tidak pada dokumen spesifikasi kebutuhan perangkat lunak?

1.3 Tujuan

Tujuan yang akan dicapai dalam pembuatan tesis ini adalah membangun sebuah sistem yang dapat mendeteksi noise secara otomatis pada dokumen spesifikasi kebutuhan perangkat lunak.

1.4 Manfaat

Manfaat dari penelitian ini adalah untuk membantu specifier dalam meningkatkan kualitas dokumen spesifikasi kebutuhan perangkat lunak yang hendak dibuat.

1.5 Kontribusi Penelitian

Kontribusi yang diharapkan dari penelitian ini adalah membantu pengembang perangkat lunak dalam mendeteksi kebutuhan fungsional yang tergolong *noise* dalam dokumen spesifikasi kebutuhan perangkat lunak.

Kebutuhan fungsional yang tergolong *noise* adalah kebutuhan yang memiliki kemiripan yang rendah dengan kebutuhan yang lain. Nilai kemiripan dapat diperoleh dengan melakukan analisa pada bobot masing-masing kebutuhan yang dihitung dengan menggunakan metode tf-idf.

Seperti yang telah dijelaskan pada subbab 1.1, metode tf-idf tidak memperhatikan kemiripan antar kata. Karenanya akan digunakan konstruksi thesaurus secara otomatis untuk mengetahui kata-kata yang memiliki makna mirip. Metode ini diharapkan mampu mendeteksi kebutuhan fungsional yang tergolong *noise* secara akurat.

1.6 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Dokumen yang digunakan menggunakan Bahasa Indonesia.
2. Kebutuhan yang diproses adalah kebutuhan fungsional saja.

BAB 2
KAJIAN PUSTAKA

2.1 Automatic Thesaurus Construction

Automatic thesaurus construction adalah proses pembentukan thesaurus secara otomatis yang berguna untuk mendapatkan kata-kata yang memiliki makna yang mirip. Nilai kemiripan dari dua kata yang berbeda didapatkan dengan menggunakan rumus

$$\mathrm{sim}(\mathrm{t_i},\ \mathrm{t_j}) = \frac{\sum_{k=1}^{N} \mathrm{w_{ik}} \ * \ \mathrm{w_{jk}}}{\sqrt{\sum_{k=1}^{N} \mathrm{w_{ik}}^{2}} \ * \sqrt{\sum_{k=1}^{N} \mathrm{w_{jk}}^{2}}} \tag{2.1}$$

dimana $\mathrm{w_{ik}}$ adalah bobot term i pada dokumen k.

2.2 TF-IDF

Term frequency – invers document frequency adalah metode pembobotan dalam pengolahan teks. Nilai term frequency (tf) didapat dengan menghitung jumlah kemunculan sebuah term dalam sebuah dokumen. Sedangkan nilai invers document frequency didapat dengan rumus

$$\mathrm{idf_i} = \log_2 \frac{N}{\mathrm{df_i}} \tag{2.2}$$

dimana $\mathrm{df_i}$ adalah jumlah dokumen yang mengandung term i. Nilai tf-idf dari term i kemudian dapat dihitung dengan mengalikan nilai tf dari term i dengan nilai idf dari term i [2].

2.3 Cosine Similarity

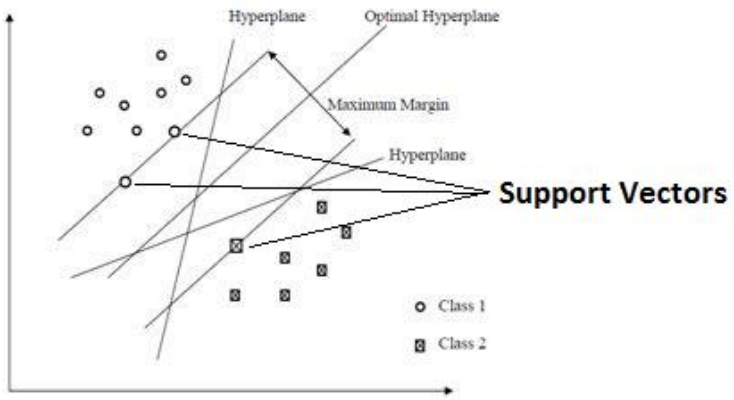
Dalam *text processing*, sebuah dokumen biasanya direpresentasikan sebagai sebuah vektor. Cosine similarity adalah sebuah metode yang digunakan untuk menghitung kemiripan antara dua vektor. Nilai cosine similarity dari vektor A dan vektor B didapat dengan menggunakan rumus

$$\mathrm{sim}(A,\ B) = \frac{\sum_{k=1}^{N} \mathrm{A_k} \ * \ \mathrm{B_k}}{\sqrt{\sum_{k=1}^{N} \mathrm{A_k}^{2}} \ * \sqrt{\sum_{k=1}^{N} \mathrm{B_k}^{2}}} \tag{2.3}$$

dimana $\mathrm{A_k}$ dan $\mathrm{B_k}$ adalah komponen dari vektor A dan B [4].

2.4 Support Vector Machine

Support vector machine (SVM) adalah metode klasifikasi *supervised* (data latih sudah diketaui kelasnya) yang mengklasifikasikan dua kelas. Pada metode ini, setiap data akan di-*plotting* ke dalam ruang *n*-dimensi (dimana *n* adalah jumlah fitur) dengan nilai masing-masing fitur menjadi nilai tertentu pada koordinat. Kemudian akan dilakukan klasifikasi dengan mencari *hyperplane* (bidang pembatas) untuk membedakan antara dua kelas sebaik mungkin. Gambar 2.1 merupakan beberapa contoh dari kemungkinan *hyperplane* yang digunakan untuk memisahkan kelas satu dengan lainnya. Metode SVM akan mencari h*yperplane* yang memiliki *margin* yang maksimal. *Support vector* adalah data-data yang dijadikan pembatas dengan kelas lain.



Sumber gambar : www.sine.ni.com

Gambar 2.1 Contoh alternatif hyperplane

Diberikan data masukan $(x_1, x_2, x_3, \ldots, x_n)$ dan masing-masing kelas dinotasikan $y_i \in \{-1, +1\}$ untuk $i = 1,2,3, \ldots, n$ dimana *n* adalah banyaknya data. Fungsi *hyperplane* dibuat dengan persamaan

$$w \,.\, x + b = 0 \tag{2.4}$$

dengan batasan yang ditulis dalam persamaan

$$y_i(wx_i + b) + \ t_i \ \geq 1 \tag{2.5}$$

untuk mencari nilai *w* dan *b* yang optimal digunakan persamaan

$$\min \frac{1}{2} ||w||^2 + C \sum_{i=1}^{l} t_i \tag{2.6}$$

dimana nilai *C* adalah nilai pinalti dari kesalahan klasifikasi. Fungsi tujuan persamaan di atas berbentuk kuadrat, sehingga untuk menyelesaikannya, bentuk tersebut ditransformasikan ke dalam bentuk *dual space*. Persamaan *dual space* dapat ditulis menggunakan persamaan

$$\max \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j x_i x_j \tag{2.7}$$

dengan batasan dalam persamaan berikut.

$$\alpha_i \geq 0 , \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.8}$$

Untuk mencari nilai α_i, digunakan *sequential minimal optimization* (SMO). Persamaan *hyperplane* dilakukan dengan persamaan:

$$f = w^T z + b = \sum_{i=1}^{s} \alpha_i y_i {x_i}^T z + b \tag{2.9}$$

dimana *z* adalah data masukan *support vector machine*. Pada banyak kasus, data yang diklasifikasikan tidak bisa langsung dipisahkan dengan garis yang linear. Oleh karena itu, digunakan metode kernel untuk mengatasi permasalahan tersebut. Dengan metode kernel, suatu data x di *input space* dipetakan ke fitur *space* F dengan dimensi yang lebih tinggi. Salah satu kernel yang biasa dipakai adalah kernel RBF dan *Polynomial*. Persamaan kernel RBF dan *Polynomial* berurutan dapat dilihat pada persamaan di bawah ini:

$$k(x, y) = \exp(-\gamma |x - y|^2) \tag{2.10}$$

$$k(x, y) = (\gamma\langle x^T\, y\rangle + r)^p \tag{2.11}$$

Penggunaan fungsi kernel mengubah persamaan *dual space* menjadi

$$\max \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \tag{2.12}$$

dan juga mengubah persamaan *hyperplane* menjadi

$$f = \sum_{i=1}^{s} \alpha_i y_i\, k(x_i, z) + b \tag{2.13}$$

[5].

2.5 Presisi dan Recall

Untuk menghitung tingkat performansi suatu sistem dapat digunakan perhitungan presisi dan *recall.* Secara matematis, rumus untuk menghitung presisi dan recall dapat dilihat pada persamaan berikut

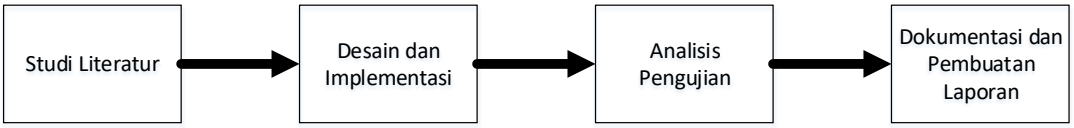
$$precision = \frac{TP}{TP + FP} \tag{2.14}$$

$$recall = \frac{TP}{TP + FN} \tag{2.15}$$

dimana TP adalah kebutuhan yang tergolong *noise* dan terklasifikasi sebagai *noise,* FP adalah kebutuhan yang bukan tergolong *noise* akan tetapi terklasifikasi sebagai *noise,* serta FN adalah kebutuhan yang tergolong *noise* akan tetapi tidak terklasifikasi sebagai *noise.*

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) analisis pengujian, dan (4) dokumentasi sistem dan jadwal kegiatan. Ilustrasi alur metodologi penelitian dapat dilihat pada **Error! Reference source not found.**.



Gambar 3.1 Alur metodologi penelitian

3.1 Studi Literatur

Penelitian diawali dengan proses pengkajian yang berkaitan dengan topik penelitian yang diambil. Pada penelitian ini, referensi yang digunakan adalah jurnal-jurnal yang berkaitan dengan metode-metode yang digunakan.

3.2 Desain dan Implementasi

Penilitian ini terdiri dari lima proses utama, yaitu praproses, thesaurus construction, pembobotan term, ekstraksi fitur, dan klasifikasi. Data masukan yang digunakan pada penelitian ini adalah kumpulan kebutuhan fungsional dari sebuah perangkat lunak, sedangkan data luarannya adalah label (*noise* / bukan) dari masing-masing kebutuhan fungsional. Satu kebutuhan fungsional akan dianggap sebagai satu dokumen.

3.2.1 Praproses

Pada tahap ini dilakukan tokenisasi, stopword removal, serta stemming. Tokenisasi adalah proses pemecahan masing-masing dokumen menjadi kumpulan term. Stopword removal adalah penghapusan kata penghubung serta tanda baca. Sedangkan proses stemming adalah proses pengubahan setiap term menjadi kata dasar.

3.2.2 Thesaurus Construction

Untuk masing-masing dokumen masukan, akan dihitung tingkat similaritasnya dengan semua term unik dalam korpus. Nilai similaritas term dengan dokumen dapat dihitung dengan rumus berikut

$$\mathrm{sim}(d_k,\ t_j) = \sum_{i=1}^{N} w_i\ *\ \mathrm{sim}(t_i,\ t_j) \qquad (3.1)$$

dimana t_i adalah term unik dalam dokumen k dan t_j adalah term unik dalam korpus. Term yang memiliki nilai similaritas melebihi batas yang ditentukan kemudian akan ditambahkan kedalam dokumen k.

3.2.3 Pembobotan Term

Masing-masing dokumen masukan kemudian akan dihitung bobotnya dengan menggunakan rumus tf-idf. Setelah proses ini, masing-masing dokumen masukan akan direpresentasikan sebagai sebuah vektor.

3.2.4 Ekstraksi Fitur

Untuk masing masing dokumen masukan, akan dihitung nilai similaritasnya dengan dokumen masukan yang lain dengan menggunakan *cosine similarity*. Setelah itu diambil tiga fitur yang kemudian akan digunakan dalam tahap klasifikasi. Tiga fitur tersebut adalah sebagai berikut.

a. Maximum of Similarity Measure

   Fitur ini adalah nilai maksimum dari nilai similaritas yang sudah didapatkan sebelumnya.

b. Mean of Similarity Measure

   Fitur ini adalah nilai rata-rata dari nilai similaritas yang sudah didapatkan sebelumnya.

c. Standard Deviation of Similarity Measure

   Fitur ini adalah nilai standar deviasi dari nilai similaritas yang sudah didapatkan sebelumnya.

3.2.5 Klasifikasi

Pada tahap ini akan dilakukan klasifikasi dari masing-masing dokumen masukan dengan menggunakan SVM.

3.3 Skenario Pengujian

a. Dataset

Dalam melakukan penelitian ini digunakan dataset berupa kumpulan kebutuhan perangkat lunak yang diambil dari beberapa dokumen SKPL yang ditambahkan beberapa *noise* untuk mencoba metode yang diajukan.

b. Sistem Penguji

Dalam melakukan pengujian, akan dilakukan variasi nilai batas / *threshold* dalam menentukan nilai similaritas dokumen dengan term. Penilaian akan dilakukan dengan memanfaatkan perhitungan presisi dan *recall* seperti yang telah diterangkan pada bab sebelumnya.

3.4 Dokumentasi Jadwal dan Pembuatan Sistem

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 **Error! Reference source not found.**

Tabel 3.1 Jadwal Kegiatan Penelitian Tesis

No.	Kegiatan	Bulan															
		Mei 2017				Juni 2017				Juli 2017				Agustus 2017			
1.	Studi Literatur																
2.	Analisa dan Perancangan																
3.	Implementasi																
4.	Pengujian dan Evaluasi																
5.	Penyusunan Buku Tugas Thesis																

DAFTAR PUSTAKA

[1] B. Meyer, "On Formalism in Specification," *IEEE Software,* 1985.

[2] B. Trstenjak, S. Mikac and D. Donko, "KNN with TF-IDF Based Framework for Text Categorization," *Procedia Engineering*, no. 69, pp. 1356-1364, 2014.

[3] H. Xu and B. Yu, "Automatic thesaurus construction for spam filtering using revised back propagation neural network," *Expert Systems with Applications,* no. 37, pp. 18-23, 2010.

[4] A. Singhal, "Modern Information Retrieval: A Brief Overview," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2001.

[5] J. C. Platt, "Sequential Minimal Optimization : A Fast Algorithm for Training Support Vector Machines," Technical Report MSR-TR-98-14, 1998.