pISSN: 2442-3386 eISSN: 2442-4293
Vol 1 No 2 Juli 2015, 1 - 6
Melek IT
INFORMATION TECHNOLOGY JOURNAL

KLASIFIKASI KEBUTUHAN NON-FUNGSIONAL MENGGUNAKAN FSKNN DENGAN PENGEMBANGAN SINONIM DAN HIPERNIM BERBASIS ISO/IEC 9126

Denni Aldi Ramadhani[1], Siti Rochimah[2], Umi Laili Yuhana[3]
[1,2,3] Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
dennialdi@gmail.com, siti@its-sby.edu, yuhana@if.its.ac.id

Abstrak

Kebutuhan non-fungsional merupakan salah satu faktor penting yang berperan dalam kesuksesan pengembangan perangkat lunak yang sering kali dilupakan oleh pengembang sehingga menimbulkan efek yang merugikan. Untuk mendapatkan kebutuhan non-fungsional dibutuhkan suatu sistem otomatisasi identifikasi kebutuhan non fungsional. Dalam penelitian ini diusulkan suatu sistem otomatisasi identifikasi kebutuhan non fungsional dari kalimat kebutuhan berbasis algoritma klasifikasi FSKNN dengan pengembangan term pada data latih menggunakan sinonim dan gabungan hipernim dan sinonim berbasis ISO/IEC 9126. Dalam pengujiannya dataset kalimat kebutuhan yang digunakan adalah 1342 kalimat dari 6 dataset yang berbeda. Hasil dari penelitian ini adalah pengembangan term dengan menggunakan gabungan hipernim dan sinonim dapat memperbaiki kinerja *accuracy* sebesar 8.1%, *precision* sebesar 13% *dan recall* sebesar 4.9%.
Kata Kunci: kebutuhan non-fungsional, klasifikasi multi-label, Fuzzy Similarity based K-Nearest Neighbour.

Abstract

Non-functional requirements is one of the important factors that play a role in the success of software development that is often forgotten by the developer, causing adverse effects. To obtain the non-functional requirements needed an automation system identification non-functional requirements. In this study proposed an an automation system of identification of non-functional requirements from the requirement sentences based classification algorithms FSKNN with term enrichment on training data using synonyms and combined hypernym and synonyms based on ISO / IEC 9126. In the testing phase the dataset of requirement sentence used was 1342 sentences from six different datasets Results of this research is the term enrichment by using a combination hypernym and synonyms can improve performance accuracy by 8.1%, precision by 13% and a recall by 4.9%.
Keywords: *Non-Functional Requirement, multi-label classification, Fuzzy Similarity based K-Nearest Neighbour.*

I. PENDAHULUAN

Mengetahui kebutuhan non fungsional yang berkaitan erat dengan aspek kualitas perangkat lunak merupakan sesuatu yang penting karena aspek kualitas dalam kebutuhan non fungsional merupakan salah satu faktor yang berperan dalam kesuksesan mengembangkan sebuah sistem. Dunia industri dalam prakteknya juga terlalu fokus pada faktor kebutuhan fungsional dan sering melupakan faktor kebutuhan non fungsional hingga mencapai pada tahap desain dan pengembangan [1]. Pada kenyataannya jika aspek kualitas kebutuhan non fungsional tidak dipertimbangkan dan tidak diketahui maka menimbulkan efek yang merugikan, baik untuk kesuksesan pengembangan sistem perangkat lunak tersebut dan juga bagi para pemangku kepentingan.

Masalah yang timbul karena kurang diperhatikannya identifikasi kebutuhan non-fungsional dalam pengembangan perangkat lunak adalah kegagalan kinerja ketika perangkat lunak telah disebarkan (*deployment*) ke lingkungan masyarakat [2] [3]. Efek merugikan juga dapat berdampak untuk para pemangku kepentingan, biasanya karena para pemangku kepentingan telah mengeluarkan dana besar untuk suatu proyek pengembangan perangkat lunak yang pada akhirnya perangkat lunak tersebut tidak berguna secara maksimal [4].

Banyak penelitian yang telah dilakukan untuk mengidentifikasi perangkat lunak, salah satu penelitian seperti yang dilakukan oleh Slankas dan Williams mengajukan sebuah sistem yang disebut dengan *NFR* Locator. Sistem ini terdiri dari dua proses yaitu pada proses pertama melakukan *parsing* dokumen kebutuhan menjadi kalimat – kalimat yang tersusun dalam bentuk *directed graph*, proses *parsing* ini menggunakan *Stanford Natural Language Parser*. Proses kedua melakukan klasifikasi terhadap kalimat hasil proses pertama dengan menggunakan pendekatan

pembelajaran diawasi berbasis algoritma KNN dan memanfaatkan fungsi *distance metric Levenshtein* untuk mengukur jarak data uji dengan data tetangga terdekatnya [5].

Pada penelitian lainnya juga mengajukan sebuah cara untuk menyelesaikan permasalahan identifikasi aspek kualitas kebutuhan non fungsioanl dengan pendekatan pembelajaran semi-diawasi (semi-supervised learning) berbasis algoritma Naive Bayes. Algoritma Naive Bayes digunakan untuk proses pelatihan yang digabungkan dengan metode Expectation Maximization (EM) yang bertujuan untuk mengurangi jumlah data latih. Proses pertama yaitu dengan melakukan praproses terhadap dokumen kebutuhan untuk mendapatkan term kunci dari dokumen, dan proses kedua adalah melakukan klasifikasi dengan metode gabungan tersebut. Dokumen yang tidak terlabeli dianggap memiliki kehilangan data karena kurangnya label kelas [6].

Beberapa penelitian diatas tidak mempertimbangkan pengembangan pada fitur atau term yang digunakan untuk melakukan proses klasifikasi. Fitur atau term pada data latih dapat dikembangkan dengan menggunakan sinonim dan hipernim dengan memanfaatkan *dictionary* Wordnet, sehingga fitur atau term yang akan digunakan pada data latih dapat berkembang maknanya secara lebih luas.

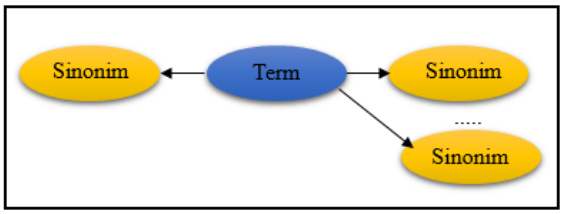
Pada penelitian ini akan mengusulkan sebuah sistem otomatisasi indentifikasi kalimat kebutuhan dengan menerapkan metode klasifikasi *multi-label* berbasis algoritma FSKNN dengan menambahkan pengembangan *term* dalam data latih berdasarkan hubungan gabungan hipernim dan sinonim berbasis Wordnet Kategori aspek kualitas yang digunakan berbasis model kualitas standar internasional ISO/IEC 9126.

II. METODE

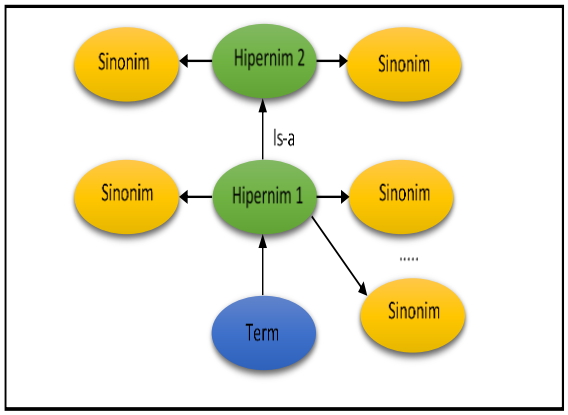
Metode penelitian ini terdiri atas 3 tahap utama yaitu : otomatisasi pelabelan data latih dengan pengembangan term sinonim dan gabungan hipernim sinonim, tahap pelatihan FSKNN yang terdiri dari subproses pengelompokan pola pelatihan dan penghitungan prior probability dan nilai likelihoods, dan tahap klasifikasi FSKNN.

2.1 Tahap Otomatisasi Pelabelan Data Pelatihan

Pada fase data pelatihan otomatisasi label terdiri dari empat tahap: preprocessing, pengembangan term dalam data pelatihan berdasarkan hubungan gabungan antara hipernim dan sinonim, pembobotan tf-idf, dan pengukuran nilai kemiripan. Semua tahapan otomatisasi data training pelabelan didasarkan pada penelitian yang telah dilakukan oleh Suharso dan Rochimah [7]. Kecuali pada fase kedua adalah kontribusi dari penelitian ini yaitu mengembangkan term pada data pelatihan menggunakan sinonim dan gabungan pola hipernim dan sinonim seperti pada Gambar 1 dan Gambar 2.



Gambar 1. Pola Pengembangan Term dengan Sinonim



Gambar 2. Pola Pengembangan Term dengan Gabungan Hipernim dan Sinonim

2.2 Tahap Pelatihan FSKNN

Tahap pelatihan Semantic-FSKNN terdiri dari dua tahap yaitu pengelompokan pola pelatihan dan penghitungan prior probability [8].

1. Pengelompokan Pola Pelatihan

Pengelompokan dokumen pelatihan $d_1, d_2, \dots, d_l$ ke dalam p kluster berbasis pada *fuzzy similarity measure.* Diberikan $dt(t_i, c_j)$ dan $dd(t_i, c_j)$ yang merupakan distribusi *term* t_i pada kategori c_j, yang didefinisikan sebagai berikut:

$$dt(t_i, c_j) = \frac{\sum_{v=1}^{l} w_{iv} y_{jv}}{\sum_{v=1}^{l} w_{iv}} \quad (1)$$

$$dd(t_i, c_j) = \frac{\sum_{v=1}^{l} sgn(w_{iv}) y_{jv}}{\sum_{v=1}^{l} y_{iv}} \quad (2)$$

Untuk $1 \le i \le m$ dan $1 \le j \le p$, dimana :

$$sgn(x) = \begin{cases} 1, & jika\ x > 0 \\ 0, & jika\ x = 0 \end{cases} \quad (3)$$

Sehingga tentunya akan didapatkan $0 \le dt(t_i, c_j)$ dan $dd(t_i, c_j) \le 1$. Tahapan selanjutnya menghitung derajat keanggotaan t_i terhadap kategori c_j, pada proses penghitungan derajat keanggotaan t_i terhadap kategori c_j diberikan suatu penambahan formula baru agar nilai yang didapatkan dari proses pengukuran keterkaitan semantik dapat ikut diperhitungkan.

pISSN: 2442-3386 eISSN: 2442-4293
Vol 1 No 2 Juli 2015, 3 - 6
Melek IT
INFORMATION TECHNOLOGY JOURNAL

Penambahan formula tersebut adalah sebagai berikut :

$$\mu_R(t_i, c_j) = \frac{dt(t_i,c_j)}{\max_{1\le u\le m, 1\le v\le p} dt(t_u,c_v)} \times \frac{dd(t_i,c_j)}{\max_{1\le u\le m, 1\le v\le p} dd(t_u,c_v)} \quad (4)$$

Untuk $1 \le i \le m$ dan $1 \le j \le p$. Tahapan selanjutnya adalah menentukan fuzzy similarity dari setiap dokumen d, d =$\langle w_1, w_2, \dots, w_m \rangle$, terhadap kategori c_j sebagai berikut :

$$sim(d, c_j) = \frac{\sum_{i=1}^{m} \mu_R(t_i,c_j) \otimes \mu_d(t_i)}{\sum_{i=1}^{m} \mu_R(t_i,c_j) \oplus \mu_d(t_i)} \quad (5)$$

Dimana $\otimes$ dan $\oplus$ merupakan *fuzzy t-norm* dan *t-conorm* yang secara berurutan didefinisikan sebagai berikut :

$$x \otimes y = x \times y \quad (6)$$

$$x \oplus y = x + y - x \times y \quad (7)$$

Dan

$$\mu_d(t_i) = \frac{w_i}{\max_{1\le v\le m} w_v} \quad (8)$$

$\mu_d(t_i)$ merupakan derajat keanggotaan dari term t_i terhadap d. Tahapan terakhir adalah mendefinisikan derajat keanggotaan dari dokumen d terhadap kategori c_j sebagai berikut :

$$\mu_{c_j}(d) = \frac{sim(d,c_j)}{\max_{1\le v\le p} sim(d,c_v)} \quad (9)$$

Untuk $1 \le j \le p$. Untuk setiap dokumen pelatihan d_i, $1 \le i \le l$, akan dilakukan perhitungan $\mu_{c_j}(d_i)$, $1 \le j \le p$, dengan menggunakan persamaan 15. Untuk mendefinisikan p kluster, $S_1, S_2, \dots, S_p$, adalah sebagai berikut :

$$S_v = \{d_u | \mu_{c_v}(d_u) \ge \alpha,\ 1 \le u \le l \quad (10)$$

Untuk $1 \le v \le p$, dimana α adalah threshold yang didefinisikan oleh user untuk digunakan dalam proses pelatihan.

Untuk setiap d_i, $1 \le i \le l$, didefinisikan *search set* G_i untuk setiap $S_v \subseteq G_i$ jika dan hanya jika $d_i \in S_v$, $1 \le v \le p$. Proses berikutnya akan dijelaskan pada *pseudo-code* yang ditunjukkan pada Gambar 3, dengan catatan pada mulanya $S_v = \emptyset$, $1 \le v \le p$, dan $G_u = \emptyset$, $1 \le u \le l$.

Keluaran yang berupa *search set* $G_1, G_2, \dots, G_l$ selanjutnya akan digunakan untuk menentukan data tetangga terdekat yang dapat membantu melakukan perhitungan nilai *prior probability* dan nilai *likelihood* pada tahapan berikutnya.

```
for untuk setiap dokumen pelatihan d_i, 1≤i≤l
    for untuk setiap kategori c_j, 1≤j≤p
        if(μ_cj(d_i) ≥ α)
        then S_j ← S_j ∪ {d_i}
for untuk setiap dokumen pelatihan d_u, 1≤u≤l
    for untuk setiap kluster S_v, 1≤v≤p
        if(d_u ∈ S_v)
        then G_u = G_u ∪ S_v
```

Gambar 3. Pseudo-code proses pengelompokan dan proses pendefinisian search set G_i [8]

2. Penghitungan Prior Probability

Diberikan $P(H_j)$ yang merupakan *prior probability* yang harus diketahui nilainya terlebih dahulu sebelum melanjutkan ke dalam setiap observasi, sedangkan $P(E|H_j)$ merupakan suatu kelas *likelihood* dan merupakan sebuah *conditional probability* bahwa H_j memiliki keterkaitan dengan observasi E. Perhitungan probabilitas ini dilakukan dari pola pelatihan yang didapatkan sebelumnya, sebagai berikut :

$$P(H_j = 1) = \frac{s + \sum_{i=1}^{l} y_{ji}}{2s + l} \quad (11)$$

$$P(H_j = 0) = 1 - P(H_j = 1) \quad (12)$$

Dimana s merupakan nilai *smoothing constant*, yang biasanya bernilai real positif kecil. Tahapan berikutnya melakukan perhitungan kelas *likelihood* $P(E|H_j)$. E dapat bernilai 0,1, ... , atau k. Untuk setiap dokumen pelatihan d_i, $1 \le i \le l$, dimana $N^i = \langle d_{v1}, d_{v2}, \dots, d_{vk} \rangle$ merupakan *k-nearest neighbor* yang diambil dari *search set* G_i dan $n^i = \langle n_1^i, n_2^i, \dots, n_p^i \rangle$, yang merupakan vektor jumlah label yang didefinisikan :

$$n_j^i = \sum_{r=v_1}^{v_k} y_{jr} \quad (13)$$

Untuk $1 \le j \le p$. Tahap selanjutnya didefinisikan :

$$Z(e,j) = \sum_{i=1}^{l} y_{ji}\, \delta_{ei}(j) \quad (14)$$

$$\tilde{Z}(e,j) = \sum_{i=1}^{l} \tilde{y}_{ji}\, \delta_{ei}(j) \quad (15)$$

Dimana $\tilde{y}_{ji} = 1 - y_{ji}$ dan

$$\delta_{ei}(j) = \begin{cases} 1, jika\ e = n_j^i \\ 0, jika\ e \ne n_j^i \end{cases} \quad (16)$$

Kemudian didefinisikan *likelihoods*, sebagai berikut :

$$P(E = e|H_j = 1) = \frac{s+Z(e,j)}{(k+1)s+\sum_{v=0}^{k} Z(v,j)} \quad (17)$$

$$P(E = e|H_j = 0) = \frac{s+\bar{Z}(e,j)}{(k+1)s+\sum_{v=0}^{k} \bar{Z}(v,j)} \quad (18)$$

Untuk setiap e = 0, 1, ... , k dan j = 1, 2, ... , p, karena ukuran dari G_i, $1 \leq i \leq l$ selalu lebih kecil daripada jumlah pola pelatihan awal, menghitung *likelihoods* dapat dilakukan secara efisien.

2.3 Tahap Klasifikasi FSKNN

Proses *testing* dalam algoritma FSKNN dilakukan dengan menggunakan estimasi *maximum a posteriori* (MAP). Dimisalkan $N^i = \{d_{v1}, d_{v2}, \dots, d_{vk}\}$ merupakan satu set *k-nearest neighbors* untuk dokumen testing d^t , dan $n^t = \langle n_1^t, n_2^t, \dots, n_p^t \rangle$ merupakan vektor jumlah label untuk d^t (persamaan 2.15), untuk menentukan kategori mana yang memiliki hubungan dengan d^t yaitu dengan cara menghitung vektor label $y^t = \langle y_1^t, y_2^t, \dots, y_p^t \rangle$ dari dokumen d^t dengan menggunakan estimasi *maximum a posteriori* (MAP) sebagai berikut :

$$y_j^t \begin{cases} 1, jika\ P(H_j = 1|E = n_j^t) > P(H_j = 0|E = n_j^t) \\ 0, jika\ P(H_j = 0|E = n_j^t) > P(H_j = 1|E = n_j^t) \\ R[0,1], otherwise \end{cases} \quad (19)$$

Untuk $1 \leq j \leq p$, dimana H_j adalah variabel acak untuk mengetahui masuk ke dalam kategori c_j atau tidak ($H_j = 1$ untuk iya, dan $H_j = 0$ untuk tidak), E merupakan variabel untuk jumlah dokumen dalam N^t yang berhubungan dengan kategori c_j , dan R[0,1] mengindikasikan 0 atau 1 dipilih secara acak. Dengan menggunakan *Bayes Rule* didapatkan :

$$P(H_j = b|E = n_j^t) = \frac{P(H_j=b)P(E=n_j^t|H_j=b)}{P(E=n_j^t)} \quad (20)$$

Untuk b = 0, 1. Oleh karena itu persamaan 19 akan berubah menjadi :

$$y_j^t \begin{cases} 1, jika\ P(H_j = 1)P(E = n_j^t|H_j = 1) > P(H_j = 0)P(E = n_j^t|H_j = 0) \\ 0, jika\ P(H_j = 0)P(E = n_j^t|H_j = 0) > P(H_j = 1)P(E = n_j^t|H_j = 1) \\ R[0,1], otherwise \end{cases} \quad (21)$$

Untuk $1 \leq j \leq p$. Tentunya untuk menghitung y_j^t harus ditemukan N^t, dan menghitung $P(H_j)$ (Persamaan 11 dan 12) dan juga $P(E|H_j)$ (Persamaan 17 dan 18). Perhitungan yang tidak tergantung pada d^t dapat dilakukan pada proses pelatihan dan sisanya dilakukan ketika proses klasifikasi. Berikut tahapan selama proses klasifikasi tersebut :

1. Hitung $\mu_{c_j}(d^t)$, $1 \leq j \leq p$ dengan Persamaan 9 menggunakan informasi atau hasil perhitungan yang dilakukan dengan Persamaan 4, 5 dan 8.
2. Periksa jika $\mu_{c_j}(d^t) \geq \alpha$ untuk $1 \leq j \leq p$, maka akan didapatkan *search set* untuk d^t yaitu $G^t = S_{j1} \cup S_{j2} \cup \dots \cup S_{jh}$.
3. Temukan set N^t dari *k-nearest neighbors* terhadap d^t dari G^t, dan dapatkan vektor jumlah label n^t.
4. Hitung y_j^t, $1 \leq j \leq p$ menggunakan Persamaan 21 dengan menggunakan informasi yang telah dihitung dalam proses pelatihan dengan Persamaan 11, 12, 17 dan 18.
5. Jika y_j^t yang didapatkan bernilai 1, maka d^t *term*asuk ke dalam kategori c_j , jika sebaliknya maka d^t tidak *term*asuk ke dalam c_j.

III. HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas tentang dataset uji coba, skenario uji coba dan analisis dari hasil uji coba.

3.1 Dataset

Dalam pengujian yang akan dilakukan pada penelitian ini menggunakan dataset dari Promise [9], Itrust, CCHIT, World Vista US Veterans Health Care System Documentation, Online Project Marking System SRS, Mars Express Aspera-3 Processing and Archiving Facility SRS dengan total keseluruhan jumlah kalimat kebutuhan mencapai 1342 kalimat. Dimana 60% (805 kalimat) akan digunakan sebagai data latih dan 40% (537 kalimat) akan digunakan sebagai data uji. Hasil klasifikasi dari data uji nantinya akan dievaluasi kinerjanya berdasarkan *accuration, precision, recall, and hamming loss* [10].

3.2 Skenario Uji Coba

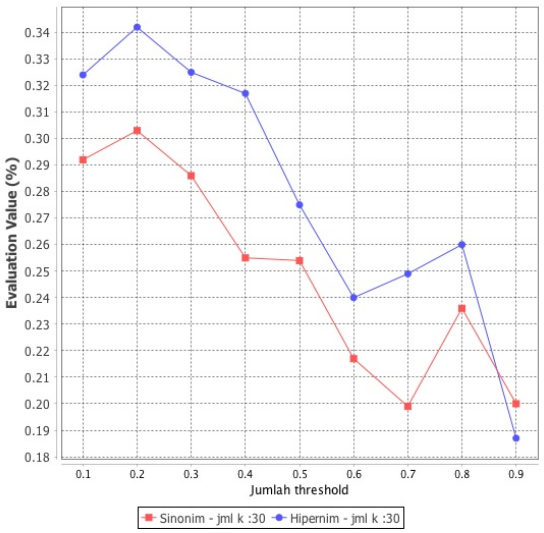
Tahap uji coba pada penelitian ini akan dilakukan dengan dua skenario yang berbeda yaitu uji coba klasifikasi FSKNN menggunakan data latih yang dikembangkan dengan sinonim dan skenario kedua yaitu uji coba klasifikasi FSKNN menggunakan data latih yang dikembangkan dengan gabungan hipernim dan sinonim. Masing – masing skenario akan diuji dengan jumlah tetangga terdekat (k) antara 10 hingga 55 dan setiap k diuji dengan nilai ambang batas antara 0.1 hingga 0.9. Skenario pengujian ini ditujukan untuk mengetahui pola pengembangan term data latih terbaik antara sinonim dan gabungan hipernim dan sinonim.

3.3 Hasil Uji Coba dan Analisis

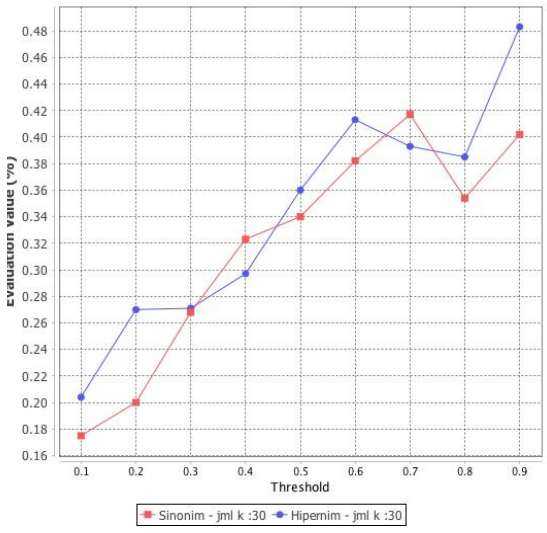
Dari hasil uji coba pada kedua skenario diketahui bahwa jumlah tetangga terdekat terbaik berada pada k = 30. Dengan hasil evaluasi *accuration, precision, recall, and hamming loss* seperti yang ditunjukkan pada Tabel 1 dan perbandingan masing – masing kinerja dapat dilihat pada Gambar 4, 5, 6 dan 7.

pISSN: 2442-3386 eISSN: 2442-4293
Vol 1 No 2 Juli 2015, 5 - 6
Melek IT
INFORMATION TECHNOLOGY JOURNAL

Threshold	FSKNN - Sinonim				FSKNN - Gabungan Hipernim - Sinonim			
	k=30				k=30			
	Hammingloss	Accuracy	Precision	Recall	Hammingloss	Accuracy	Precision	Recall
0,1	0,292	0,175	0,396	0,225	0,324	0,204	0,435	0,276
0,2	0,303	0,2	0,411	0,308	0,342	0,27	0,426	0,43
0,3	0,286	0,268	0,447	0,412	0,325	0,271	0,453	0,413
0,4	0,255	0,323	0,503	0,499	0,317	0,297	0,462	0,461
0,5	0,254	0,34	0,501	0,526	0,275	0,36	0,53	0,546
0,6	0,217	0,382	0,581	0,528	0,24	0,413	0,621	**0,577**
0,7	**0,199**	**0,417**	0,659	**0,553**	0,249	0,393	0,616	0,547
0,8	0,236	0,354	0,537	0,531	0,26	0,385	0,604	0,542
0,9	0,2	0,402	**0,662**	0,539	**0,187**	**0,483**	**0,792**	0,559



Gambar 4. Perbandingan kinerja Hammingloss antara pengembangan Sinonim dan Gabungan Hipernim Sinonim



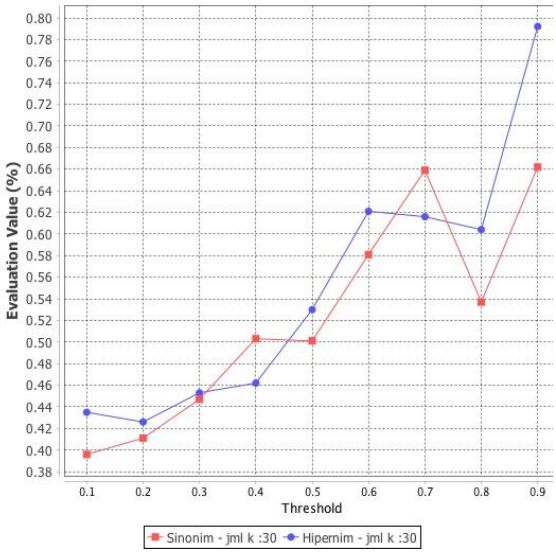
Gambar 5. Perbandingan kinerja Accuracy antara pengembangan Sinonim dan Gabungan Hipernim Sinonim

Pada Gambar 4 diatas dapat diketahui bahwa kinerja hamming loss (tingkat kesalahan) pada klasifikasi FSKNN dengan data latih yang dikembangkan menggunakan sinonim lebih rendah dibandingkan klasifikasi FSKNN dengan data latih pengembangan gabungan hipernim dan sinonim. Dari grafik pada Gambar 4 diatas dapat diketahui bahwa pengembangan dengan menggunakan gabungan hipernim dan sinonim tidak efektif untuk kinerja hammingloss karena dapat memicu tingkat kesalahan yang lebih tinggi. Tingkat kesalahan yang lebih tinggi dikarenakan pengembangan dengan menggunakan gabungan hipernim dan sinonim dapat mencakup makna term atau fitur yang lebih luas dibandingkan hanya dengan menggunakan sinonim saja.

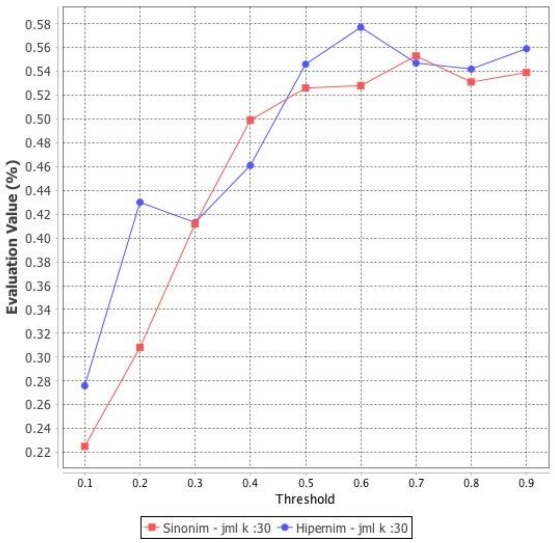
Pada Gambar 5 dapat diketahui bahwa secara mayoritas kinerja *accuracy* pada klasifikasi FSKNN dengan menggunakan pengembangan data latih gabungan hipernim dan sinonim lebih tinggi dibandingkan klasifikasi FSKNN dengan menggunakan pengembangan data latih dari sinonim saja. Titik optimum terletak pada ambang batas 0.9 dengan nilai 48.3% dan dapat diketahui kenaikan nilainya sebesar 8.1% dari nilai FSKNN dengan pengembangan sinonim pada ambang batas yang sama.

Pada Gambar 6 dibawah dapat diketahui kinerja *precision* untuk klasifikasi FSKNN dengan pengembangan gabungan hipernim dan sinonim lebih tinggi secara mayoritas dengan titik optimum terletak pada ambang batas 0.9 dengan nilai 79.2% dan kenaikan nilainya sebesar 13% dari nilai FSKNN dengan pengembangan sinonim pada ambang batas yang sama.

Pada Gambar 7 dibawah dapat diketahui kinerja *recall* untuk klasifikasi FSKNN dengan pengembangan gabungan hipernim dan sinonim lebih tinggi secara mayoritas dengan titik optimum terletak pada ambang batas 0.6 dengan nilai 57.7% dan kenaikan nilainya sebesar 4.9% dari nilai kinerja *recall* FSKNN dengan pengembangan sinonim pada ambang batas yang sama.



Gambar 6. Perbandingan kinerja Precision antara pengembangan Sinonim dan Gabungan Hipernim Sinonim



Gambar 7. Perbandingan kinerja Precision antara pengembangan Sinonim dan Gabungan Hipernim Sinonim

IV. KESIMPULAN

Dari hasil uji coba pada penelitian ini dapat disimpulkan bahwa pengembangan term atau fitur pada data latih dengan menggunakan gabungan hipernim dan sinonim terbukti dapat meningkatkan kinerja accuracy, precision, dan recall dibandingkan dengan pengembangan term atau fitur menggunakan sinonim saja, tetapi pengembangan term atau fitur dengan menggunakan data latih juga memberikan kinerja hamming loss yang lebih tinggi dibandingkan pada pengembangan term menggunakan sinonim.

DAFTAR PUSTAKA

[1] S. Ullah, M. Iqbal and A. M. Khan, "A Survey on Issues in Non-Functional Requirements Elicitation," in 2011 International Conference on Computer Networks and Information Technology (ICCNIT), Peshawar, Pakistan, 2011.

[2] Finkelstein and J. Dowel, "A Comedy of Errors: the London Ambulance Service case study," in Proceedings of the 8th International Workshop on Software Specification and Design, 1996.

[3] J. Bertman and N. Skolnik, "EHRs Get a Failing Grade on Usability," Internal Medicine News, vol. 43, p. 50, 2010.

[4] Hoskinson, "Politico," News, 29 Juni 2011. [Online]. Available: http://www.politico.com/news/stories/0611/58051.html. [Accessed 20 Oktober 2014].

[5] J. Slankas and L. Williams, "Automated Extraction of Non-Functional Requirements in Available Documentation," in 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE), San Francisco, CA, USA, 2013.

[6] Casamayor, D. Godoy and M. Campo, "Identification of non-functional requirements in textual specifications : A semi-supervised learning approach," Information and Software Technology, vol. 52, pp. 436-445, 2010.

[7] W. Suharso and S. Rochimah, SIstem Deteksi dan Klasifikasi Otomatis Kebutuhan Non Fungsional Berbasis ISO/IEC 9126, Surabaya: Institut Teknologi Sepuluh November, 2013.

[8] J.-Y. Jiang, S.-C. Tsai and S.-J. Lee, "FSKNN: Multi-label text categorization based on fuzzy similarity and k nearest neighbors," Expert Systems with Applications, vol. 39, pp. 2813-2821, 2012.

[9] PROMISE, "tera-PROMISE," 1 April 2010. [Online].Available:http://openscience.us/repo/requirements/other-requirements/nfr.html. [Accessed 20 Januari 2015].

[10] P. Prajapati, A. Thakkar and A. Ganatra, "A Survey and Current Research Challenges in Multi-Label Classification Methods," International Journal of Soft Computing and Engineering (IJSCE), vol. 2, no. 1, pp. 248-252, 2012.