MATICS : Jurnal Ilmu Komputer dan Teknologi Informasi ISSN : 1978-161X(p); 2477-2550(e)
Volume 9, No. 1(2017), pp 67-71 DOI : 10.18860/mat.v9i1.4534
Received : July 15th 2017; **Accepted** : August 9th 2017 ; **Avalaible Online** : September 15th 2017

Pendeteksian Ketidak lengkapan Kebutuhan Dengan Teknik Klasifikasi Pada Dokumen Spesifikasi Kebutuhan Perangkat Lunak

Suci Nurfauziah, Daniel Oranova Siahaan, Fajar Baskoro

***Abstract*—Software Requirements Specification (SRS) document was produced by software requirements and this is a critical stage in Software Development. Errors that occur in the software requirements will affect the failure of the product. SRS often written in natural language. One of the characteristics of good requirements specification is complete. The quality requirements specification can be assessed based on the statement or requirements document. Requirement specification is complete that defines precisely all the situations confronting the system and can be understood without related another requirements. This research purposed to establish a classification model incompleteness detection requirements in software requirements specification document written in natural language. This study makes corpus that contain statements of requirement complete and incomplete. Corpus manually written by three experts. There will be features extraction, features validation dan keyword generation from corpus. Gwet's AC1 performance value will be used to determine whether the classifier reliable and detect the presence of incompleteness in SRS.**

Based on the results of the experiment using a combination of adaboost and C4.5 methods, the average of the agreement index is obtained at the moderate level with the best value of 0.52 at the time of use of the top six features. The top six features is bad_jj, bad_rb, jml_kt_penegasan, jml_kt_penghubung, bad_prp dan jml_kt_negatif.

***Keyword*— Tesk Classification, Natural Language Processing, Software Requirement Specification, Completeness.**

***Abstrak*–- Dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL) dihasilkan dari proses rekayasa kebutuhan dan merupakan tahapan yang kritis pada pengembangan perangkat lunak. Kesalahan yang terjadi pada proses rekayasa kebutuhan akan mempengaruhi ketidakberhasilan produk tersebut. Dokumen SKPL sering kali ditulis dengan bahasa alamiah. Salah satu karakteristik spesifikasi kebutuhan yang baik adalah lengkap. Kualitas spesifikasi kebutuhan bisa dinilai berdasarkan pernyataan kebutuhan atau dokumen kebutuhan. Spesifikasi kebutuhan yang lengkap secara jelas mendefinisikan semua situasi yang dihadapi sistem dan dapat dipahami tanpa melibatkan atau terkait pada kebutuhan lain. Penelitian ini bertujuan untuk membangun model klasifikasi pendeteksian ketidaklengkapan kebutuhan pada dokumen spesifikasi kebutuhan perangkat lunak yang ditulis dengan bahasa alamiah. Penelitian ini membuat *corpus* kebutuhan yang berisi pernyataan kebutuhan lengkap dan pernyataan kebutuhan tidak lengkap. *Corpus* ditulis secara manual oleh tiga orang ahli. Dari *Corpus* akan dilakukan ekstraksi fitur, pemilihan fitur yang valid, dan pembangkitan kata kunci. Nilai performansi Gwet's AC1 digunakan untuk mengetahui apakah *classifier* yang dibangun dapat diandalkan dan dapat mendeteksi adanya ketidaklengkapan pada dokumen spesifikasi kebutuhan perangkat lunak.**

Berdasarkan hasil ujicoba dengan menggunakan kombinasi metode adaboost dan C4.5 diperoleh rata-rata indek kesepakatan pada level *moderate* dengan nilai tertinggi 0.52 pada saat penggunaan enam fitur teratas. Enam fitur teratas yang paling berpengaruh antara lain bad_jj, bad_rb, jml_kt_penegasan, jml_kt_penghubung, bad_prp dan jml_kt_negatif.

***Kata Kunci*— Klasifikasi Teks, Pemrosesan Bahasa Alami, Spesifikasi Kebutuhan Perangkat Lunak, Kelengkapan.**

I. PENDAHULUAN

Rekayasa kebutuhan merupakan tahap awal dalam proses pengembangan perangkat lunak. Tahapan yang terjadi dalam proses rekayasa kebutuhan akan mempengaruhi proses pengembangan perangkat lunak. Rekayasa kebutuhan menghasilkan dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL). Pembuatan dokumen SKPL haruslah berhati-hati agar menghasilkan sebuah perangkat lunak yang berkualitas. Kesalahan yang terjadi pada rekayasa kebutuhan akan mempengaruhi ketidakberhasilan produk tersebut, tidak peduli seberapa baik tahapan selanjutnya[1].

Penelitian pada bidang rekayasa kebutuhan khususnya mendeteksi kualitas kebutuhan telah banyak dilakukan, beberapa jurnal menjelaskan tentang kerancuan seperti yang dilakukan oleh [2], [3], dan [4], ketidakkonsistensian oleh [5], mendeteksi konflik pada kebutuhan perangkat lunak oleh [6], dan kebenaran kebutuhan perangkat lunak [7]. Pada penelitian ini akan

Manuscript received July 15, 2017. This work was supported in part by Informatics Department of Institut Teknologi Sepuluh Nopember, Surabaya.

Suci Nurfauziah is with the Informatics Departement of Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia (email suci.fauziah15@mhs.if.its.ac.id)

Daniel O. Siahaan is the the Informatics Departement of Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia.

Fajar Baskoto is the Informatics Departement of Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia.

difokuskan untuk mendeteksi ketidaklengkapan kebutuhan perangkat lunak.

Catarina Gralha(2015) mengukur kompleksitas dan kelengkapan dokumen spesifikasi kebutuhan yang ditulis dengan i* model[8]. I* model merupakan salah satu bentuk bahasa formal, sedangkan dokumen SKPL seringkali ditulis dengan bahasa alamiah. Pada penelitian lain, Genova menyebutkan beberapa karakteristik untuk menilai kualitas kebutuhan yang ditulis dengan bahasa alamiah, salah satunya adalah kelengkapan. Beberapa fitur yang diusulkan oleh Genova untuk menilai kelengkapan spesifikasi kebutuhan perangkat lunak bersifat tidak langsung dan tidak ada cara untuk mengukur kelengkapan tersebut dengan menggunakan fitur itu[9].

Penelitian lain dari Hussain, memanfaatkan *text mining* untuk membangun *classifier* yang dapat mendeteksi adanya kerancuan pada dokumen SKPL[10]. Sehingga pada penelitian ini berusaha mengembangkan teknik yang digunakan oleh Hussain untuk mendeteksi ketidaklengkapan pada dokumen SKPL.

II. KAJIA PUSTAKA

A. Kelengkapan Pada Kebutuhan Perangkat Lunak

Menurut [11] urntuk mengevaluasi kebutuhan, maka terdapat dua aspek kualitas SKPL, yaitu :

- Kualitas pernyataan kebutuhan yaitu kualitas dari kalimat tunggal yang dinilai secara terpisah.
- Kualitas dokumen kebutuhan yaitu kualitas dari beberapa kalimat dipertimbangkan dalam konteks dokumen secara keselutuhan.

Sebuah kebutuhan yang lengkap menunjukkan setiap pernyataan kebutuhan dapat dipahami tanpa melibatkan atau terkait pada kebutuhan lain. Pernyataan kebutuhan seharusnya kalimat lengkap yang tidak memerlukan referensi pada pernyataan lain untuk dipahami sebagai bentuk dasar. Sedangkan dokumen kebutuhan dinyatakan lengkap adalah semua fungsi, fitur didefinisikan secara jelas sesuai dengan keinginan stakeholder[12].

Sebuah kebutuhan juga dikatakan lengkap jika semua informasi terhindar dari kerancuan dan tidak perlu penjelasan tambahan[13]. Pernyataan kebutuhan lengkap merupakan kemampuan setiap kebutuhan untuk menjelaskan dengan tepat kebutuhannya. Sedangkan dokumen kebutuhan yang lengkap adalah dokumen spesifikasi kebutuhan yang terhindar dari kemungkinan atau ketidakcocokan diantara pernyataan kebutuhan.

Pada penelitian [9] menyebutkan beberapa fitur untuk menilai kualitas kebutuhan yang berpengaruh secara tidak langsung terhadap kelengkapan. Fitur tersebut antara lain :

a. Berdasarkan *morfologi* :
   - Ukuran terkait dengan jumlah kata pada pernyataan kebutuhan.

b. Berdasarkan *lexical* :
   - Kata sambung
   - Kata rancu/tidak jelas meliputi bentuk ekspresi subjective seperti kata sifat serta kata keterangan.

c. Berdasarkan *analytical* :
   - *Verbal tense and mood* meliputi aturan sebagai berikut :
     - ✓ Kebutuhan seharusnya memiliki minimal kata yang menunjukkan penegasan (*imperative verb*).
     - ✓ Menghindari penggunaan kata kerja secara implisit, maksudnya kata kerja yang menunjukkan ekspresi kondisional dalam kalimat. Ekspresi kondisional dengan menerapkan kata-kata *modal* seperti *should, can, may*, kecuali kata *must* yang identic dengan penegasan.
     - ✓ Menghindari penggunaan kata kerja pasif.

   – *Domain terms* : Penggunaan kata kerja dan kata benda dalam satu kalimat secara berlebihan.

d. Berdasarkan *relational* :
   - Ketergantungan antar kebutuhan
   - *Overlapping* pada seluruh kebutuhan : adanya dua konflik antara dua kebutuhan dan *redundant* pada seperangkat kebutuhan.

Fitur pada point ke a sampai c merupakan fitur kualitas untuk setiap pernyataan kebutuhan, sedangkan fitur pada poin ke d merupakan fitur kualitas untuk dokumen kebutuhan[9].

B. Adaboost dan C4.5

Boosting yang digunakan pada peneliatian ini adalah adaboost. Adaboost merupakan pengembangan dari boosting. Metode ini menghasilkan serangkaian base klasifikasi. Proses latih yang digunakan untuk setiap klasifikasi berdasarkan peforma klasifikasi sebelumnya.

Boosting merupakan metode yang umum digunakan untuk meningkatkan learning algortithm. Schapire pertama kali memperkenalkan algoritma boosting pada tahun 1990. Pada tahun 1995 Freund dan Schapire mengembangkannya kemudian disebut AdaBoost. Ide utama algoritma ini adalah menentukan bobot pada setiap set data latih. Pada awalnya semua bobot adalah sama, selanjutnya bobot akan diperbaharui. Jika prediksi data latih tersebut benar maka bobot akan dikurangi dan sebaliknya jika salah maka bobot klasifikasi akan ditambah[14]. Penerapan boosting pada C4.5 memberikan akurasi yang lebih baik dari pada versi standar (C4.5) dengan menambahkan bobot pada setiap kelas data latih[15].

Algoritma ini mengasumsikan bahwa seperangkat data latih terdiri dari m *instances*, diberi label -1 atau 1. Klasifikasi dari *instance* dibuat dari semua klasifikasi $\{C_t\}$ yang masing-masing memiliki bobot α_t. Secara matematik dapat ditulis[14]:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t . C_t(x)\right) \quad (1)$$

III. TAHAP PENELITIAN

Pengembangan metode pendeteksi ketidaklengkapan kebutuhan terdiri dari tiga sub-proses, yaitu pembuatan *corpus*, *preprocessing* dan pengembangan klasifikasi.

A. *Pembuatan Corpus*

Pada penelitian ini terdapat *corpus* yang berisi kalimat-kalimat spesifikasi kebutuhan yang bersifat lengkap dan kalimat spesifikasi kebutuhan yang tidak lengkap. Akan dipilih beberapa orang ahli yang menguasahi dan memahami mengenai *Requirements Engineering* atau *Software Engineering* serta menguasahi bahasa Inggris. Sehingga apabila ketiga ahli sepakat tidak lengkap maka pernyataan kebutuhan tersebut berlabel tidak lengkap, dan bila ketiganya sepakat lengkap maka berlabel lengkap. Namun, bila terdapat ahli yang berbeda pendapat maka diambil suara terbanyak dari dua orang ahli yang sependapat.

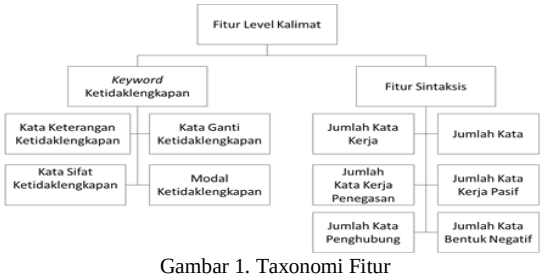
Dokumen Spesifikasi Kebutuhan Perangkat Lunak yang akan dijadikan *corpus* dalam penelitian ini merupakan potongan *"problem description"* yang diambil dari *ACM's OOPSLA DesignFest®* (http://designfest.acm.org/) secara acak. *Problem description* berkaitan dengan spesifikasi perangkat lunak akan diekstraksi menjadi pernyataan-pernyataan kebutuhan. Kemudian pernyataan kebutuhan tersebut akan dianotasi oleh ahli yang dibekali dengan dokumen panduan anotasi. *Corpus* yang telah terbentuk dipilah berdasarkan anotasinya. $Corpus_1$ terdiri dari kalimat yang beranotasi lengkap dan $corpus_2$ terdiri dari kalimat yang beranotasi tidak lengkap. $Corpus_1$ dan $corpus_2$ secara keseluruhan akan digunakan pada proses pemilihan fitur.

B. *Preprocessing*

Tahap ini bertujuan untuk membangun dataset pelatihan dan pengujian. Dataset pelatihan dan pengujian dibangun dari *corpus* yang sudah dihasilkan dari proses sebelumnya. Pada proses preprocessing terdiri dari beberapa sub proses, yaitu : proses ekstraksi fitur dan perankingan fitur dengan metode *information gain*.

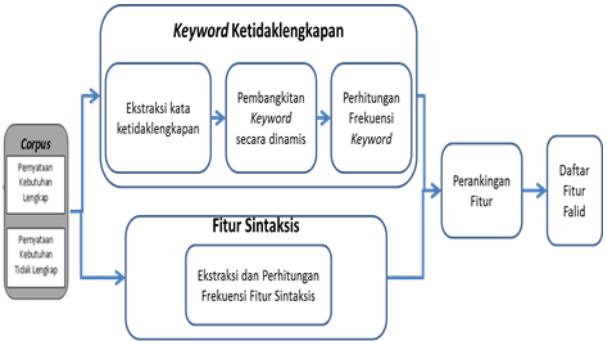
Pada tahap ini proses dimulai dengan mengidentifasi kandidat fitur pernyataan kebutuhan yang dapat digunakan untuk mendeteksi ketidaklengkapan kebutuhan. Gambar 3.3 menunjukkan kandidat fitur pernyataan kebutuhan yang digunakan. Ekstraksi Fitur pada penelitian ini menggunakan model kualitas pernyataan kebutuhan yang diperkenalkan oleh [9] dan [12] khususnya bagian untuk menilai kualitas kebutuhan secara individu. Pendeteksian lengkap atau tidak pernyataan kebutuhan akan dilihat dari kata kunci dan fitur-fitur sintaksis.

Berdasarkan taxonomi fitur pada gambar 1 maka pada tahap ini akan dibagi menjadi dua tahapan, yaitu



Gambar 1. Taxonomi Fitur

ekstraksi fitur kata kunci ketidaklengkapan dan fitur sintaksis. Selanjutnya untuk proses dari kedua tahapan tersebut dijelaskan pada Gambar 2. Gambar 2 merupakan alur kerja tahapan ekstraksi dan pemilihan



Gambar 2. Ekstraksi Fitur

fitur secara umum.

Seperti yang ditunjukkan pada gambar 2, tahap fitur kata kunci ketidaklengkapan terdiri dari tiga sub-proses, yaitu ekstraksi kata ketidaklengkapan, pembangkitan kata kunci secara dinamis, dan perhitungan frekuensi kata kunci. Sedangkan fitur sintaktis terdiri dari sub-proses yaitu ekstraksi dan perhitungan frekuensi fitur sintaktis.

Untuk proses ekstraksi dari masing-masing tahapan sama-sama melakukan proses *tokenizing* dan *POS tagging*. *Tokenizing* merupakan proses memecah dokumen maupun kalimat menjadi kata atau term. *POS tagging* merupakan proses yang dilakukan untuk mendapatkan informasi tag dari setiap kata yang ada pada dokumen. Proses *tokenizing* dan *POS tagging* menggunakan *library Stanford Natural Language Processing.*

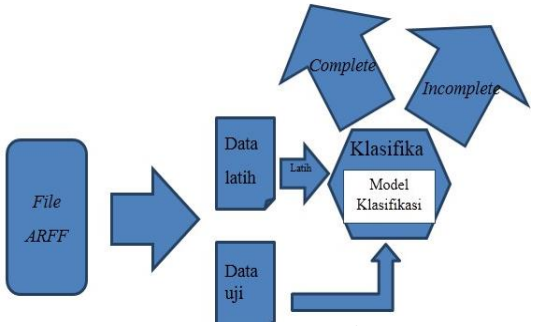
Pada tahap fitur sintaksis setelah didapat tag dari setiap kata maka selanjutnya dihitung jumlah frekuensi kata untuk setiap fitur (jumlah kata kerja, jumlah kata, jumlah kata penegasan, dan jumlah kata kerja pasif). Sedangkan pada tahap fitur kata kunci ketidaklengkapan setelah didapat tag dari setiap kata maka dilakukan proses pembangkitan kata kunci dan perhitungan jumlah frekuensi kata untuk setiap kata kunci.

Setelah kedua tahapan ekstraksi fitur maka dilakukan proses perankingan dengan menggunakan metode *information gain.*

C. *Pengembangan Klasifikasi*

Tahap ini bertujuan untuk membangun klasifikasi yang menggunakan kombinasi adaboost dengan C4.5.

Gambar 3 merupakan gambaran tahapan klasifikasi. Setelah *corpus* di POS tagger dan dihitung freuensi kata kunci serta fitur valid maka akan menghasilkan data yang berformat ARFF. Data selanjutnya dibedakan menjadi data latih dan data uji. Data latih dilakukan klasifikasi terlebih dahulu untuk membentuk model klasifikasi. Selanjutnya data uji akan dilakukan klasifikasi sesuai hasil model dari proses latih sebelumnya. Klasifikasi akan dibangun memanfaatkan perangkat lunak *Weka* yang menyediakan implementasi kombinasi antara adaboost dengan C4.5.



Gambar 3. Tahapan Klasifikasi

D. *Metode Evaluasi*

Pada penelitian ini akan digunakan *Gwet's AC1* untuk menilai hasil klasifikasi yang telah dibuat. Nilai AC1 akan menjadi tolok ukur apakah kerangka kerja yang diusulkan dapat diandalkan untuk mendeteksi adanya ketidaklengkapan pada pernyataan kebutuhan perangkat lunak.

Untuk menghitung nilai AC1, hasil observasi pengamat ditulis dalam bentuk tabel 2x2 seperti yang ditunjukkan pada Gambar 4.

Pengamat 1	Pengamat 2		
	Ya	Tidak	Total
Ya	A	B	B1=A+B
Tidak	C	D	B2=C+D
Total	A1=A+C	A2=B+D	N

Gambar 4. Kesepakatan Pengamat

Untuk menghitung AC1-statistic, maka terlebih dahulu menghitung kesepakatan yang terobservasi (P) ditunjukkan pada persamaan 2 berikut ini :

$$P = \frac{A+D}{N} \quad (2)$$

Selanjutnya menghitung probabilitas chance-agreement dengan persamaan 3 berikut :

$$e(y) = 2P_1(1 - P_1) \quad (3)$$

dimana $P_1 = \frac{(A1+B1)/2}{N}$ merepresentasikan perkiraan kemungkinan seorang pengamat (1 atau 2) mengelompokkan data ke dalam kategori "Ya". A_1 dan B_1 masing-masing adalah jumlah pengamat 1 atau 2 mengelompokkan data ke dalam kategori "Ya", sedangkan N adalah jumlah data. Maka hasil AC1-statistic dihitung dengan persamaan 4, berikut :

$$AC1 = \frac{P-e(Y)}{1-e(Y)} \quad (4)$$

IV. UJI COBA DAN HASIL

Dari hasil pembuatan corpus oleh para ahli, maka didapat 141 pernyatan kebutuhan berlabel tidak lengkap dan 286 pernyataan kebutuhan berlabel lengkap. Sehingga total data sebanyak 427 pernyatan kebutuhan.

Selanjutnya maka dilakukan pembangkitan kata kunci yang mungkin mengindikasi sebagai tidak lengkap. Hasil pembangkitan kata kunci ketidaklengkapan ditunjukkan pada tabel 1. Pembangkitan kata kunci ini dikelompokkan sesuai jenisnya, yaitu kata sifat(bad_JJ), kata keterangan(bad_RB), kata ganti(bad_PRP) dan modal.

Tabel 1. Pembangkitan Kata Kunci

Kelompok Kata	Daftar Kata Kunci Ketidaklengkapan
bad_JJ	*equivalent, accurate, bad, desirable, inactive, real-time, on-line, above, similar, large, old, lower, personal, optional, easy, lowest, slow, appropriate, accessible, internal, other, alive, least, manual, current, same, overdue, established, outstanding, necessary, efficient, downstream, possible, token, single, connected, such, fast, able, certain, first, essential, scalable, small, acceptable, compact, adequate, own, several, simple, good, centralized, different, new, individual, particular, less, multi-byte, read-out, most, formal, safe, final, fixed, basic, present, largest, inexpensive, financial, proper, random, tunable, total, public, parallel, responsible, original, application-specific, multiple, clear, enough, specific, intelligent, third, detailed, corresponding, due, various, groupable, multi-lingual, online, compliant, chronological, base*
bad_RB	*very, quickly, back, recursively, periodically, immediately, elsewhere, intricately, improperly, precisely, not, later, easily, usually, so, gracefully, directly, just, fully, otherwise, successfully, unfrequently, recently, also, often, then, statistically, dynamically, as, currently, strongly, efficiently, relatively, significantly*
bad_PRP	*his, themselves, its, them, it, he, they*
bad_MD	*can, may, will, could, might, should*

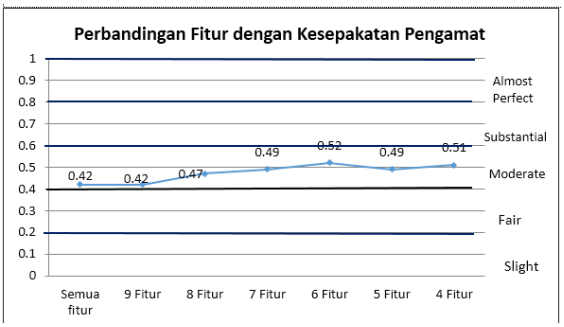
Hasil proses ekstraksi fitur disimpan dengan format file ARFF (*Attribute-Relation File Format).*

Tabel 2 merupakan hasil peranking fitur dengan menggunakan metode Information Gain. Perangkingan fitur digunakan untuk memilih fitur yang valid dan fitur yang berpengaruh pada hasil klasifikasi.

Tabel 2. Perankingan Fitur

No	Fitur	Nilai
1	BAD_JJ	0.091332
2	BAD_RB	0.034328
3	jml_KT_PENEGASAN	0.019380
4	jml_KT_PENGHUBUNG	0.001703
5	BAD_PRP	0.001259
6	jml_KT_NEGATIF	0.001030
7	BAD_MD	0.000652
8	jml_KT_PASIF	0.000248
9	jml_KT_KERJA	0.000002
10	jml_KATA	0

Berdasarkan hasil ujicoba dengan menggunakan iterasi sebanyak 40 dan menerapkan hasil perankingan, maka hasilnya dapat dilihat pada gambar 5. Hasil penelitian menunjukkan indek kesepakatan dengan Gwet AC1 static nilai rata-ratanya 0.47 yang berada pada tingkat *moderate*(cukup). Gambar 5 juga mengambarkan kenaikan hasil klasifikasi dari 0.42(10 fitur) berangsur-angsur naik sampai pada puncak kenaikan dengan penggunaan enam fitur teratas sebesar 0.52. Setelah pengurangan fitur berikutnya mengalami penurunan. Jadi dapat disimpulkan bahwa penggunaan enam fitur teratas sangat berpengaruh terhadap hasil klasifikasi. 6 fitur teratas tersebut antara lain bad_jj, bad_rb, jml_kt_penegasan, jml_kt_penghubung, bad_prp dan jml_kt_negatif.



Gambar 5. Tahapan Klasifikasi

V. KESIMPULAN

Dari penelitian ini bahwa kerangka kerja yang diusulkan mampu mendeteksi ketidaklengkapan dokumen spesifikasi kebutuhan perangkat lunak dengan nilai kesepakatan rata-rata 0.47 yaitu berada tingkat *moderate* (cukup) dan nilai tertinggi 0.52 pada saat penggunaan enam fitur teratas. Enam fitur teratas yang paling berpengaruh antara lain bad_jj, bad_rb, jml_kt_penegasan, jml_kt_penghubung, bad_prp dan jml_kt_negatif.

Penerapan metode *information gain* untuk perankingan dapat meningkatkan hasil klasifikasi dan mengetahui fitur-fitur mana saja yang berpengaruh. Beberapa fitur yang berpengaruh dan menunjukkan hasil maksimal antara lain bad_jj, bad_rb, jml_kt_penegasan, jml_kt_penghubung, bad_prp dan jml_kt_negatif.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada M.Ainul Yaqin, Galuh Sari, Devi Karolita, Hendy Dwi Harfianto, dan Tandhy Simanjuntak yang merupakan para ahli *Software Engineering* atas bantuannya dalam penyusunan *corpus.*

DAFTAR PUSTAKA

[1] R. Pressman, *Rekayasa Perangkat Lunak*, 7th ed. Yogyakarta: Andi, 2012.

[2] I. Hussain, O. Ormandjieva, and L. Kosseim, "Automatic quality assessment of SRS text by means of a decision-tree-based text classifier," in *Proceedings - International Conference on Quality Software*, 2007, no. Qsic, pp. 209–218.

[3] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Requirements for tools for ambiguity identification and measurement in natural language requirements specifications," *Requir. Eng.*, vol. 13, no. 3, pp. 207–239, 2008.

[4] D. Popescu, S. Rugaber, N. Medvidovic, and D. M. Berry, "Reducing ambiguities in requirements specifications via automatically created object-oriented models," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5320 LNCS, pp. 103–124, 2008.

[5] T. C. de Sousa, J. R. Almeida, S. Viana, and J. Pavón,"Automatic analysis of requirements consistency with the B method," *ACM SIGSOFT Softw. Eng. Notes*, vol. 35, no. 2, p. 1, 2010.

[6] A. Sardinha, R. Chitchyan, N. Weston, P. Greenwood, and A. Rashid, "EA-Analyzer: Automating conflict detection in a large set of textual aspect-oriented requirements," *Autom. Softw. Eng.*, vol. 20, no. 1, pp. 111–135, 2013.

[7] E. Parra, C. Dimou, J. Llorens, V. Moreno, and A. Fraga, "A methodology for the classification of quality of requirements using machine learning techniques," *Inf. Softw. Technol.*, vol. 67, pp. 180–195, 2015.

[8] C. Gralha, J. Araújo, and M. Goulão, "Metrics for measuring complexity and completeness for social goal models," *Inf. Syst.*, vol. 53, pp. 346–362, 2015.