KLASIFIKASI ULASAN APLIKASI PADA TOKO APLIKASI BERGERAK DENGAN MEMANFAATKAN ISSUE TRACKER GITHUB

Joko Prasetyo[1)] dan Daniel Oranova Siahaan[2)]
[1, 2)]Teknik Informatika, Institut Teknologi Sepuluh Nopember
Jl. Teknik Kimia, Gedung Teknik Informatika, Kampus ITS Sukolilo, Surabaya, 60111
e-mail: joko.prsty@gmail.com[1)], danielos@cs.its.ac.id[2)]

ABSTRAK

Dengan semakin maraknya aplikasi untuk perangkat bergerak, membuat para pengembang harus berkompetisi untuk membuat aplikasi sesuai dengan keinginan pengguna. Ulasan pengguna pada suatu aplikasi adalah salah satu cara untuk mencapainya. Terdapat penelitian yang memanfaatkan ulasan tersebut, yaitu rekomendasi fitur aplikasi untuk pengembang. Penelitian tersebut mengekstraksi fitur aplikasi dari suatu ulasan dan mengklasifikasi ke dalam dua tipe, yaitu laporan kesalahan atau permintaan fitur. Pada proses klasifikasi, dataset yang dipakai berasal dari hasil penelusuran ulasan toko aplikasi bergerak Google Play dan IOS App Store. Kekurangan dari dataset tersebut adalah perlunya pelabelan manual oleh para ahli, dimana hal tersebut membutuhkan waktu yang tidak sedikit, selain itu data yang dihasilkan tidak terlalu banyak sehingga hal tersebut akan berpengaruh pada hasil akurasi. Penelitian ini mengusulkan penggunaan dataset berlabel dari sumber lain yang jumlahnya melimpah, yaitu judul dari issue pada repository perangkat lunak terbuka Github, yang mana dataset tersebut akan digunakan sebagai data latih. Tujuan dari penelitian yang diusulkan adalah untuk mengetahui apakah penggunaan data dari sumber lain dapat dipakai sebagai dataset latih dengan akurasi yang lebih baik atau tidak. Dari hasil penelitian diharapkan keterlibatan ahli dalam proses pelabelan dapat dihilangkan. Selain itu, diharapkan model klasifikasi tersebut dapat digunakan untuk klasifikasi dengan domain yang lebih luas, misalnya klasifikasi perangkat lunak desktop. Metode klasifikasi yang dipakai adalah naïve bayes, karena telah dibuktikan pada penelitian sebelumnya, bahwa metode naïve bayes menghasilkan akurasi yang lebih tinggi daripada decision tree dan Max Ent. Hasil Penelitian menunjukkan bahwa model yang diusulkan menghasilkan precision sebesar 61-69% dan recall sebesar 30-88%.

***Kata Kunci**: klasifikasi teks, naïve bayes, pengolahan bahasa alamiah, spesifikasi kebutuhan perangkat lunak.*

ABSTRACT

The popularity of apps for mobile device make apps developers have to compete to make apps according to user's wishes. The user review of an app could be one of the ways to achieve it. There was a research that made use of the user review that was app feature recommendation for developers. The research was about extracting app's features of a review and classifying the review into two types, either bug report or feature request. In the classification process, dataset that were used came from crawling apps stores, Google Play and IOS. The drawback of the dataset is that it needs some experts to label them; it needs much time to complete them, other than that the dataset that are yielded are little. So that it will affect to the accuracy of the classification. This research proposes using labeled dataset from another source that have large amount of data, the dataset are taken from title of issues in Github's repositories. The purpose of the research is to know whether the data that are retrieved from another source can be used as training data with better accuracy or not. Hopefully, by using the training dataset, it can eliminate expert involvement in the data labeling. In addition, the proposed model can be used to classify vaster domain, such as desktop software. The classification method that is used in this research is Naïve Bayes; it was proved that it yields better accuracy than Decision Tree and Max Ent. The research results show that the proposed model yield precision about 61-69% and recall about 30-88%.

***Keywords**: text classification, naïve bayes, natural language processing, software requirement spesification.*

I. PENDAHULUAN

DEWASA ini perkembangan toko aplikasi bergerak sangat pesat. Pada tahun 2010-2016, toko aplikasi bergerak seperti *Android Play Store* mencapai unduhan sebanyak 60 miliar kali.[1] Hal ini tidak lain dikarenakan perangkat bergerak, seperti ponsel dan tablet yang telah terintegrasi Sistem Operasi, semakin terjangkau di kalangan masyarakat. Sehingga kebutuhan akan aplikasi bergerak semakin tinggi.

Toko aplikasi bergerak memberi kemudahan kepada para pengembang dalam menyediakan aplikasi kepada para pengguna. Untuk mendapatkan aplikasi yang diinginkan, pengguna cukup mengunduh aplikasi tersebut ke dalam perangkat bergerak mereka. Pengguna juga dapat memberikan ulasan mengenai aplikasi yang telah mereka unduh, seperti opini, rating, maupun permintaan. Ulasan tersebut sangat penting bagi pengembang, perangkat lunak dengan ulasan yang baik, dapat meningkatkan penjualan dan jumlah unduhan [1]. Selain itu, ulasan adalah hal yang penting dalam mendukung kesuksesan pengembangan perangkat lunak [2].

[1] https://www.statista.com/statistics/281106/number-of-android-app-downloads-from-google-play/

Beberapa penelitian memanfaatkan ulasan pengguna untuk memudahkan pengembang dalam mengembangkan aplikasi pada toko aplikasi bergerak. Salah satunya adalah [3] yang mengekstraksi fitur aplikasi dari ulasan pengguna dan mengklasifikasikannya ke dalam dua tipe, yaitu laporan kesalahan dan permintaan fitur.

Penelitian pada [3] menggunakan dataset ulasan dari [4] untuk digunakan sebagai dataset latih dan dataset uji. Dataset tersebut dihasilkan dari penelusuran ulasan pada toko perangkat lunak bergerak yaitu *Google Play* dan *IOS App Store*. Dalam penelitian tersebut, data ulasan pada dataset latih dan dataset uji dilakukan pelabelan secara manual oleh beberapa ahli yang kompeten di bidangnya. Pelabelan secara manual adalah hal yang tidak efisien, karena akan membutuhkan tenaga dan waktu yang sangat banyak, terlebih lagi banyaknya ulasan yang dikirim oleh pengguna untuk perangkat lunak yang populer [5].

Banyaknya tenaga dan waktu yang digunakan dalam proses pelabelan, membuat data yang dihasilkan untuk proses pelatihan dan pengujian terbatas. Sehingga hal tersebut kemungkinan dapat mempengaruhi akurasi dalam proses klasifikasi yang dilakukan.

Dari hal tersebut, penelitian ini mengusulkan perbaikan terhadap model klasifikasi ulasan yang dilakukan oleh [3]. Model klasifikasi ulasan yang diusulkan tidak lagi membangun dataset latih dan dataset uji dari ulasan yang melibatkan ahli dalam proses pelabelan, akan tetapi menggunakan pengetahuan berupa data berlabel yang berasal dari sumber lain sebagai dataset uji. Data dari sumber lain yang dimaksud adalah judul dari *issue* pada *issues tracker repository* proyek perangkat lunak kode terbuka, seperti *Github* (*Github.com*). Data yang akan dipakai dalam penelitian yang diusulkan adalah data dengan label laporan kesalahan dan permintaan fitur.

Keunggulan dataset uji dari sumber lain ini adalah data uji yang banyak dan dinamis. *Repository* data uji akan selalu berkembang, baik dari sisi kosa-kata, istilah, kasus, ataupun teknologi, dan sensitif terhadap perkembangan waktu, selain itu model klasifikasi tersebut dapat digunakan pada domain yang lebih luas, tidak hanya pada aplikasi bergerak saja, tetapi kemungkinan juga dapat digunakan pada perangkat lunak desktop.

Metode klasifikasi dalam penelitian yang diusulkan, menggunakan naïve bayes yaitu *binary classification*. Menurut [6], klasfikasi naïve bayes dengan menggunakan *binary classification*, menghasilkan akurasi yang lebih tinggi daripada metode klasifikasi yang lain, yaitu decision tree dan Max Ent.

II. Penelitian Terkait

Seksi ini akan menjelaskan tentang beberapa penelitian yang pernah dilakukan sebelumnya, untuk mendukung penelitian yang diusulkan.

A. Keikutsertaan Pengguna dalam Rekayasa Kebutuhan

Peninjauan dilakukan terhadapat beberapa literatur rekayasa perangkat lunak [7] untuk mengetahui pengaruh keterlibatan pengguna pada proses pengembangan perangkat lunak. Peninjauan dilakukan pada literatur yang diambil dari berbagai sumber, yaitu *IEEExplore, science direct, ACM DL, Springer link, Google Scholar* dan *MISQ*, dengan periode 1980-2012. Total literatur yang dipakai sebanyak 87 paper. Penelitian tersebut menggunakan metode Systematic Literatur Review (SLR) berdasarkan panduan dari *Evidence Based Software Engineering* (EBSE) [8]. Hasil penelitian menunjukkan bahwa keterlibatan pengguna memberikan efek positif pada kesuksesan sistem dari proses pengembangan perangkat lunak.

Dalam penelitian yang lain, keikutsertaan pengguna berpengaruh dalam proses rekayasa kebutuhan [9]. Penelitian tersebut dilakukan dengan melakukan survei dan wawancara. Survei dilakukan terhadap 22 pekerja profesional pengembangan perangkat lunak pada perusahaan multinasional. Para responden diberikan kuesioner yang berisi 10 seksi, yaitu *personal information, project information, requirement engineering process, user requirement, user involvement, customer requirement, requirement analysis, project evaluation,* dan *requirement process evaluation.* Sedangkan wawancara dilakukan terhadap 8 praktisi perangkat lunak. Topik yang diajukan kepada responden yaitu *stakeholder's involvement, requirement elicitation, effects of different user characteristics on their needs, problems and improvements on RE process*. Hasil dari penelitian tersebut menunjukkan bahwa keterlibatan pengguna dapat meningkatkan kualitas proyek perangkat lunak. Selain itu, pengguna yang memiliki keahlian serta berpengalaman dalam pengembangan proyek perangkat lunak, lebih memberikan manfaat pada proses rekayasa kebutuhan daripada pengguna yang tidak memiliki keahlian dan tidak berpengalaman.

Keikutsertaan pengguna juga dapat berupa *crowdsourcing* yaitu suatu bentuk kontribusi berbasis massa, seperti penelitian [10], yang melakukan penelitian mengenai rekayasa kebutuhan berbasis massa. Penelitian ini mengusulkan rekayasa kebutuhan semi otomatis yang menganalisa timbal balik pengguna, dengan tujuan untuk mendapatkan kebutuhan pengguna yang valid. Penelitian tersebut mengumpulkan timbal balik dari pengguna melalui interaksi langsung dan kolaborasi sosial dengan memanfaatkan metode *text mining* dan *data mining*.

Pada penelitian [4] yang dikembangkan pada penelitian [6], timbal balik pengguna yang berupa ulasan dari toko

aplikasi bergerak, dimanfaatkan untuk membantu pengembang dalam meningkatkan kualitas aplikasi bergerak. Penelitian tersebut bertujuan untuk mendeteksi secara otomatis, ulasan dari aplikasi yang bersangkutan, dengan cara mengklasifikasikan ulasan tersebut ke dalam 4 tipe, yaitu: *bug reports, feature requests, user experiences, and text ratings.* Pada penelitian tersebut, ulasan diklasifikasikan menggunakan 3 metode, yaitu naïve bayes, decision tree, dan max ent. Dari ketiga metode tersebut, naïve bayes menghasilkan akurasi yang lebih baik. Selain itu, penelitian ini juga membandingkan antara *string matching, text classification, natural language processing, sentiment analysis* dan *review metadata*. Data ulasan didapat dari hasil penelurusan toko aplikasi bergerak *Android* dan IOS. Data yang dipakai dalam proses klasifikasi adalah sebanyak 4400. Sebelum dilakukan pengklasifikasian, data ulasan dilakukan pelabelan oleh beberapa ahli yang kompeten di bidangnya. Label pada ulasan yang dipakai dalam penelitian tersebut yaitu: *bug reports, feature requests, user experiences,* dan *text ratings.*

Selain itu, penelitian [3] juga memanfaatkan timbal balik pengguna yang berupa ulasan dari aplikasi toko bergerak, sebagai rekomendasi untuk pengembang aplikasi, dalam meningkatkan kualitas dalam prses pengembangan aplikasi bergerak. Pada penelitian tersebut, data ulasan diekstraksi untuk diambil fitur aplikasinya, kemudian diklasifikasikan ke dalam dua tipe: yaitu laporan kesalahan dan permintaan fitur. Dataset yang dipakai adalah dataset dari [4].

B. Keikutsertaan Pengguna dalam Rekayasa Kebutuhan

Penelitian yang dilakukan pada [11], membahas mengenai *issues tracker* pada proyek perangkat lunak, yang memberikan dampak besar pada pengembangan perangkat lunak, dalam kesalahan mengindentifikasi *bug* atau *feature*. Kemudian, Penelitian yang dilakukan pada [12], mengklasifikasikan timbal balik pengguna pada *bug*/*Issue tracker* secara otomatis. Model klasifikasi pada penelitian tersebut menentukan apakah suatu timbal balik dapat diklasifikasikan untuk digunakan dalam aktivitas pengembangan perangkat lunak. Hasil dalam penelitian ini menunjukkan bahwa *bug/issue* dapat diklasifikasikan secara tepat sebesar 77-82%.

C. Klasifikasi Lintas Domain

Penggunaan dataset lintas domain, biasanya dipakai untuk mengatasi kurangnya data berlabel yang digunakan dalam proses klasifikasi [13][14]. Dataset diambil dari domain lain yang telah memiliki label, dan memiliki data dalam jumlah besar. Dataset tersebut kemudian digunakan sebagai dataset latih.

III. METODOLOGI DAN DATA

Dalam seksi ini akan dijelaskan mengenai metodologi serta data yang dipakai dalam penelitian yang diusulkan.

A. Penggalian Data

Data untuk proses pelatihan diambil dari hasil penelusuran pada situs proyek perangkat lunak sumber terbuka yaitu *Github.* Data tersebut berupa judul dari *issue* yang ditulis oleh pengembang atau pengguna perangkat lunak yang bersangkutan. *Github* memberikan tempat bagi pengembang perangkat lunak untuk mengembangkan perangkat lunak yang mereka buat. Perangkat lunak tersebut tersimpan pada *Github* dalam sebuah *repository*. Terdapat *repository* publik dan privat yang disediakan oleh *Github*. *Repository* publik dapat diakses oleh semua pengguna selain anggota, sedangkan *repository* privat, hanya dapat diakses oleh pengguna yang bersangkutan. Penelitian ini hanya menggunakan *repository* publik karena dapat mudah diakses untuk proses pengambilan data.

Saat ini, *Github* memiliki *repository* publik lebih dari 1.000.000. Untuk mengakses *repository* tersebut, *Github* menyediakan *API* yang dapat diakses menggunakan *web service.*[2] Namun ada juga beberapa *repository* populer yang telah didaftar oleh *Github,* yang dikelompokkan ke dalam *showcase.*[3] *Showcase* tersebut berjumlah 54 buah. Penelitian ini menggunakan *repository* yang terdapat pada *showcase tersebut*, karena *repository* populer biasanya memiliki data *issue* yang lebih banyak. *Showcase* tersebut berisi *repository* dengan kategori campuran, yaitu *repository* dari perangkat lunak sistem dan perangkat lunak aplikasi. Penelitian yang diusulkan, akan menggunakan dua jenis kategori *repository*, yaitu *repository* campuran yang terdapat pada *Showcase,* dan repository aplikasi bergerak Android dan IOS.[4] *Repository* aplikasi bergerak berjumlah 765 buah. Pengguanaan kedua jenis kategori *repository* tersebut dimaksudkan untuk menguji pengaruh domain pada akurasi dari hasil proses klasifikasi.

Data yang dipakai dalam penelitian yang diajukan, adalah judul *issue* dengan label *bug* dan *feature*. Kedua label tersebut dipakai karena sesuai dengan penelitian [3], yang mana label *bug* akan diasosiasikan dengan laporan

[2] *https://developer.github.com/v3/repos/*
[3] *https://github.com/showcases*
[4] *https://github.com/pcqpcq/open-source-android-apps/tree/master/categories, dan https://github.com/dkhamsing/open-source-ios-apps/blob/master/APPSTORE.md*

TABEL I
HASIL PEROLEHAN DATA JUDUL *ISSUE* PADA *GITHUB* SETELAH PROSES PENYARINGAN

No	*Repository*	Jumlah		Total
		Feature	*Bug*	
1	Campuran	45.242	95.634	140.876
2	Aplikasi bergerak	3.508	11.791	15.299

TABEL II
JUMLAH DATASET LATIH

No	*Repository*	Jumlah		Total
		Feature	*Bug*	
1	Campuran	45.000	45/000	90.000
2	Aplikasi bergerak	3.500	3.500	7.000

TABEL III
JUMLAH DATASET UJI

No	*Repository*	Jumlah
1	*Feature*	295
2	*Bug*	370
	Total	665

kesalahan, sedangkan label *feature* akan diasosiasikan dengan permintaan fitur. Data yang berhasil dikumpulkan kemudian disimpan ke dalam database SQL dengan menggunakan database *sqlite3*.

Data yang didapat dalam proses penelusuran *repository* campuran sebanyak 1.321.274, dengan berbagai macam label. Sedangkan dari *repository* aplikasi bergerak sebanyak 17.839, dengan label *bug* dan *feature*. Dalam data yang telah didapat, terdapat beberapa data yang memiliki label lebih dari satu, sehingga terdapat beberapa data yang redundan.

Data tersebut kemudian dilakukan penyaringan ke dalam dua label yaitu *bug* dan *feature*. Setelah dilakukan penyaringan data, data dengan *repository* campuran dengan label *bug*, didapat sebanyak 95.634 data, dan dengan label *feature* didapat sebanyak 45.242. Sedangkan data dengan *repository* aplikasi bergerak, dengan label *bug* didapat sebanyak 11.791, dan dengan label *feature* didapat sebanyak 3.508. Tabel I menunjukkan banyak data setelah proses penyaringan.

Untuk mempermudah proses pelatihan dan untuk meningkatkan relevansi data, serta mencegah ketimpangan hasil akurasi dalam proses klasifikasi, maka data latih untuk kedua kelas dibuat 50:50. Data pada Tabel I dilakukan pengurangan menjadi 45.000 untuk data dalam *repository* campuran, dan 3500 untuk data dalam *repository* aplikasi bergerak. Penyesuaian data ditunjukkan pada Tabel II.

Untuk dataset uji, diambil dari [6], dengan jumlah data sebanyak 665. Dataset dengan label *bug* sebanyak 370, sedangkan dataset dengan label *feature* sebanyak 295. Jumlah dataset uji ditunjukkan pada Tabel III.

B. Ekstraksi Data

Tahap selanjutnya adalah proses ekstraksi informasi data, dengan keluaran berupa *Bag of Words*. *Bag of words* adalah kumpulan kata atau frase yang didapat dari kemunculan pada suatu data. Proses ekstraksi informasi data melibatkan pemrosesan bahasa alamiah yaitu *lemmatization* [3]. *Lemmatization* adalah proses untuk mengubah sebuah kata ke dalam bentuk dasar dari kata tersebut, misalnya *"fixing"*, *"fixed"* dan *"fixes"* menjadi *"fix"*. Proses *lemmatization* menggunakan *library* pemrosesan bahasa alamiah yaitu *Freeling*. *Freeling* adalah *library* yang ditulis dalam bahasa C++ yang telah dilengkapi beberapa fungsionalitas diantaranya adalah *tokenization*, *lemmatization*, dan lain – lain.

Setelah data dilakukan pemrosesan bahasa alamiah, selanjutnya adalah proses *tokenization* yaitu memecah kalimat ke dalam kata atau frase [3]. Kemudian setelah itu, dilanjutkan proses *stopword removal* yaitu penghapusan kata yang sering muncul, tetapi tidak berpengaruh pada hasil akurasi klasifikasi [3].

Daftar *stopword* diambil dari [6]. Selain itu, proses penghapusan juga dilakukan pada kata yang tidak dimulai dengan karakter a sampai z (contoh [,], !, dan %).

Pada penelitian ini, dipakai dua bentuk fitur klasifikasi, yaitu *unigram* dan *bigram* [3]. *Unigram* adalah kata atau frase yang berjumlah 1, sedangkan *bigram* adalah kumpulan dari dua kata atau frase. Langkah selanjutnya adalah menghitung jumlah kemunculan kata untuk fitur klasifikasi *unigram* dan *bigram*. Proses ini akan menghasilkan keluaran *Bag of words unigram* dan *bigram*.

C. Model Klasifikasi

Metode klasifikasi pada penelitian ini menggunakan algoritma naïve bayes [15], dengan *binary classification*. Metode tersebut menghasilkan dua macam kelas, yaitu "ya" dan "tidak". Pada penelitian ini, terdapat dua tipe ulasan yang akan diklasifikasikan, yaitu laporan kesalahan, dan permintaan fitur. Oleh karena itu diperlukan dua model klasifikasi, yaitu model klasifikasi untuk tipe laporan kesalahan, dan model klasifikasi untuk tipe permintaan fitur. Untuk model permintaan fitur, kelas "ya" menggunakan fitur klasifikasi pada tipe *feature*. Sedangkan untuk kelas "tidak", menggunakan fitur klasifikasi pada tipe *bug.*

D. Pengujian dan Evaluasi

Pada tahap ini akan dilakukan pengujian dan evaluasi terhadap model yang yang diusulkan, dengan tujuan untuk mengetahui tingkat keberhasilan dari model tersebut. Dataset yang dipakai untuk tahap pengujian adalah dataset dari [6]. Dataset tersebut terbagi sebanyak empat tipe yaitu: *bug*, *feature*, *rating*, dan *user experience*. Namun, dataset yang dipakai dalam proses pengujian adalah dataset dengan tipe *bug* dan *feature*, sesuai dengan tipe dataset yang dipakai pada proses pelatihan yaitu tipe *bug* dan *feature*. Tujuan dari penggunaan kedua label tersebut adalah, agar relevan dengan dataset latih.

Skenario pengujian dilakukan berdasarkan [6], yaitu hanya diambil skenario dengan akurasi tinggi yang memungkinkan untuk diuji. Terdapat dua skenario pengujian di dalam penelitian ini. Skenario pertama adalah kombinasi fitur *Bag of Words unigram, lemmatization, stopword removal* dan skenario kedua adalah kombinasi fitur *Bag of Words bigram, lemmatization, stopword removal*.

Terdapat dua model pengujian, yaitu model pengujian untuk tipe laporan kesalahan dan model pengujian untuk tipe permintaan fitur. Hasil dari pengujian kemudian akan dievaluasi menggunakan *precision* dan *recall,* dan juga *F-measure (F1)* [15].

IV. HASIL PENELITIAN

Seksi ini akan membahas mengenai hasil penelitian dan evaluasi.

A. Hasil Penelitian

Tabel IV menunjukkan hasil evaluasi untuk *repository* campuran dan *repository* aplikasi bergerak, dengan perbandingan antara teknik *Bag of Words unigram*, dan *Bag of Words bigram.* Hasil evaluasi berupa nilai *precision*, *recall*, dan *F-measure* (*F1)* [15], dari tipe permintaan fitur dan laporan kesalahan.

Pada Tabel IV terlihat bahwa, untuk *repository* aplikasi bergerak, tipe permintaan fitur menghasilkan akurasi yang jauh lebih kecil dibandingkan dengan tipe laporan kesalahan. Permintaan fitur menghasilkan nilai *F1* sebesar 40% untuk *Bag of words unigram*, dan 42% untuk *Bag of words bigram*. Sedangkan laporan kesalahan menghasilkan nilai *F1* sebesar 72% untuk unigram, dan 70% untuk bigram. Untuk *repository* campuran, *Bag of Words unigram* memiliki nilai *F1* yang lebih tinggi daripada *Bag of Words bigram.*

Nilai *F1* pada tipe permintaan fitur untuk bigram dan unigram sama, sebesar 54%, sedangkan pada tipe laporan kesalahan, nilai unigram 74%, lebih tinggi daripada nilai bigram, yaitu 71%. Keunggulan *Bag of word unigram* dibandingan *Bag of word bigram* dalam model ini adalah, kemungkinan disebabkan oleh panjang kalimat ulasan yang kecil. Sehingga hal tersebut menghasilkan kombinasi kata *bigram* yang tidak relevan dengan dataset uji.

Jika dibandingkan nilai *repository* aplikasi bergerak dengan *repository* campuran, terlihat bahwa nilai *F1* pada *repository* campuran lebih tinggi dari pada *repository* aplikasi bergerak. Hal tersebut menunjukkan bahwa, meskipun ulasan dataset uji memiliki kategori yang sama dengan ulasan pada dataset latih, tidak serta merta membuat nilai *F1* dari hasil klasifikasi menjadi tinggi.

Untuk mengetahui berapa banyak data yang teridentifikasi benar dan salah, maka perlu dilakukan proses identifiksi kelas, yang ditunjukkan pada Tabel V. Untuk tipe permintaan fitur, nilai *true positif* lebih kecil daripada *false negative*, artinya banyak ulasan dengan kelas "ya", teridentifikasi ke dalam "tidak", hal ini akan mempengaruhi nilai *recall*, sehingga nilai *recall* bernilai kecil. Nilai *true positif* semakin bertambah tinggi pada *repository* campuran dengan teknik *Bag of words bigram*, artinya nilai *recall* semakin tinggi apabila menggunakan teknik tersebut. Sedangkan untuk tipe permintaan fitur, nilai *true positive* mengalami penurunan pada teknik *Bag of words bigram*.

Banyaknya identifikasi *false negative* dan *false positive*, bisa disebabkan karena data uji memiliki kesalahan pelabelan, yaitu data yang seharusnya memiliki label bug, ditulis dengan label *feature*, begitu juga sebaliknya, data dengan label *feature*, ditulis dengan label *bug*. Tabel VI menunjukkan data yang kemungkinan memiliki kesalahan pelabelan.

TABEL IV
HASIL EVALUASI

Teknik	Permintaan Fitur			Laporan Kesalahan		
	Precision	*Recall*	*F1*	*Precision*	*Recall*	*F1*
Aplikasi bergerak						
Bag of Words unigram	0.67	0,30	0,40	0,61	*0,88*	0.72
Bag of Words bigram	0,64	0,32	0,42	0,61	0,80	0,70
Campuran						
Bag of Words unigram	*0.69*	0,44	*0,54*	*0,66*	0,85	*0.74*
Bag of Words bigram	0,66	*0,46*	*0,54*	0,65	0,78	0,71

TABEL V
HASIL IDENTIFIKASI KELAS

Teknik	Permintaan Fitur				Laporan Kesalahan			
	False positive	*True negative*	*True positif*	*False negatif*	*False positive*	*True negative*	*True positive*	*False negative*
Aplikasi bergerak								
Bag of Words unigram	41	*329*	85	*210*	*209*	86	*327*	43
Bag of Words bigram	53	317	93	202	187	100	295	75
Campuran								
Bag of Words unigram	59	311	131	164	166	129	316	54
Bag of Words bigram	*69*	301	*135*	160	155	*140*	289	*81*

TABEL VI
CONTOH DATA YANG KEMUNGKINAN MEMILIKI KESALAHAN PELABELAN

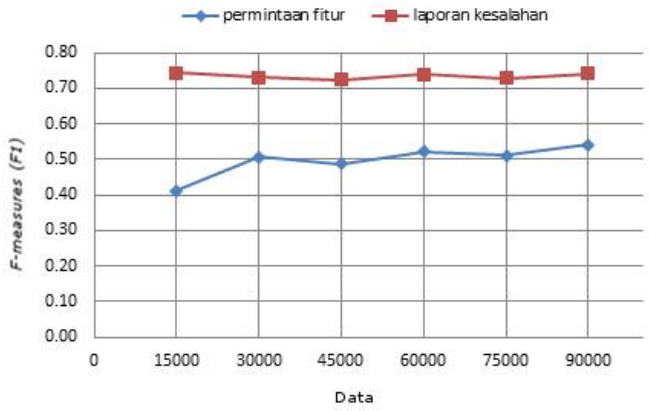
No	Tipe	Data
1	*feature*	*Can't open the apps\tWhenever I open the apps it shut down after few seconds. I tried uninstall and reinstall, and its the same. It was ok before the last update.*
2	*feature*	*App runs too slow!!!*
3	*feature*	*Keeps auto-deleting every 3 days\tThis app is quite nice but ever since the last update it keeps auto deleting. I have re-installed it thrice including today. That is just wrong!*
4	*feature*	*Keeps auto-deleting every 3 days\tThis app is quite nice but ever since the last update it keeps auto deleting. I have re-installed it thrice including today. That is just wrong!*
5	*feature*	*Wont load\tI have been playin this for few weeks now all of sudden it wont load*
6	*bug*	*Would like it to have more edit items..*
7	*bug*	*The update has been installing for a week*
8	*bug*	*Plz more backgrounds and frame*
9	*bug*	*You need to improve the hebrew editing..*

TABEL VII
RATA-RATA PANJANG KALIMAT DATASET UJI DAN DATASET LATIH

Data	Rata-Rata Panjang Kalimat (Julah Kata)	
	feature	*bug*
Dataset Latih *Github*		
Campuran	6,1176	7,1773
Aplikasi bergerak	5,3942	6,6971
Dataset Uji Toko Aplikasi Bergerak		
Feature	27,427	
Bug	29,024	

TABEL VIII
KATA-KATA PALING INFORMATIF PADA *BAG OF WORDS UNIGRAM*

No	Kata	Mutual Info
1	*add*	0,04107
2	*not*	0,03415
3	*support*	0,02282
4	*feature*	0,02214
5	*request*	0,01635
6	*do*	0,01572
7	*fail*	0,01456
8	*error*	0,01387
9	*when*	0,01345
10	*be*	0,01164
11	*break*	0,01105
12	*work*	0,01019
13	*allow*	0,00973
14	*crash*	0,00936
15	*with*	0,00814
16	*can*	0,00583
17	*bug*	0,00558
18	*fix*	0,00554
19	*cause*	0,00552
20	*incorrect*	0,00542



Gambar 1. Perbandingan jumlah data latih dalam proses klasifikasi

B. Akurasi Jumlah Data

Untuk mengetahui hubungan antara akurasi dan banyak data, maka dilakukan pengujian terhadap jumlah data latih yang berbeda-beda. Nilai yang dibandingkan untuk masing-masing jumlah data, adalah nilai F1 .Komposisi data yang dipakai adalah 50:50, 50% untuk data kelas "ya" dan 50% untuk data dengan kelas "tidak". Banyak data latih yang dipakai, mulai dari 15.000 sampai 90.000, dengan rentang antara data sebesar 15.000.

Dari Gambar 1 menunjukkan bahwa jumlah data mempengaruhi akurasi dari masing-masing model klasifikasi, yaitu model permintaan fitur, dan model laporan kesalahan. Nilai *F1* untuk model permintaan fitur, mengalami fluktuasi setiap penambahan data sebanyak 15.000 data, tetapi mengalami kecenderungan naik setiap penambahan data sebanyak 30.000 data. Sedangkan nilai *F1* pada model laporan kesalahan tidak menunjukkan tren naik atau tren turun. Nilai *F1* pada model laporan kesalahan mengalami fluktuasi untuk setiap penambahan data.

C. Panjang Kalimat

Keragaman kosa-kata dari *Bag of words*, dapat dipengaruhi oleh panjang kalimat dari dataset yang dipakai. Rata-rata panjang kalimat dari dataset, dapat dilihat pada Tabel VII. Dataset latih memiliki rata-rata panjang kalimat yang lebih kecil dibandingkan dataset uji. Hal ini dikarenakan dataset *Github* yang berupa judul. Biasanya judul memiliki karakteristik jelas, singkat, dan padat.

Jika dibandingkan antara rata-rata panjang kalimat *feature* dan *bug*, *bug memiliki* rata-rata panjang kalimat yang lebih besar daripada bug. Hal ini juga memungkinkan keragaman kosa-kata pada dataset *bug* lebih banyak daripada dataset *feature*. Sehingga kemungkinan menyebabkan akurasi pada tipe laporan kesalahan lebih tinggi daripada permintaan fitur.

D. Kata Informatif

Tabel VIII menunjukkan kata-kata yang paling informatif [16] yang terdapat pada *Bag of word unigram.* Kata-kata tersebut mempengaruhi akurasi dari proses klasifikasi yang dilakukan. Semakin tinggi nilai mutual infonya, maka pengaruhnya semakin besar dalam proses klasifikasi.

Terlihat bahwa kata-kata pada Tabel VIII tidak berpengaruh pada domain yang spesifik. Sebagian besar kata-kata tersebut umum digunakan pada domain manapun dalam *issue* perangkat lunak. Hal tersebut menunjukkan bahwa data *issue* dengan domain apapun, dapat digunakan sebagai data latih pada proses klasifikasi.

V. DISKUSI

Pada seksi ini akan dibahas mengenai temuan dan kendala selama melakukan penelitian:

- Tidak semua *repository* memiliki *issue* didalamanya. Sehingga pada proses penggalian data, data yang didapat sedikit meskipun banyak *repository* yang telah ditelusuri.
- Data dengan label *bug* selalu lebih banyak daripada data dengan label *feature.* Hal ini dikarenakan kesalahan dari perangkat lunak, akan selalu muncul selama proses pengembangan perangkat lunak masih terus berjalan. Sedangkan fitur perangkat lunak hanya akan muncul apabila ada permintaan dari pengembang atau pengguna perangkat lunak yang bersangkutan. Ketimpangan data antara label *bug* dan *feature* akan menyebabkan akurasi dari proses klasifikasi dengan model laporan kesalahan, lebih tinggi daripada permintaan fitur.
- Banyak kata-kata teknis yang berhubungan dengan kode program atau modul-modul dari perangkat lunak

pada judul *issue* di *repository Github*, seperti kata *AddInterfaceController* dan *__NSCFString replaceCharactersInRange:withString.* Hal ini dapat mengurangi keragaman kosa-kata umum yang terdapat pada *Bag of words*. Untuk mengatasi hal tersebut, dibutuhkan proses pengambilan data *issue* yang sangat banyak, yang mana hal tersebut akan mempengaruhi waktu proses klasifikasi, yaitu waktu yang dibutuhkan dalam proses tersebut akan semakin lama.

- Kesalahan kemungkinan bisa dilakukan oleh pengembang dalam melakukan pelabelan pada *issue* dari perangkat lunak yang bersangkutan. Data yang seharusnya memiliki label *bug*, pada kenyataanya memiliki label *feature*, begitu juga sebaliknya. Biasanya data dengan label *feature* cenderung lebih banyak memiliki kesalahan pelabelan. Selain itu data dengan label lain, terkadang dilakukan pelabelan dengan label *bug* atau *feature*. Kesalahan pelabelan tersebut akan berefek pada kurangnya akurasi pada proses klasifikasi.

VI. KESIMPULAN

Hasil pengujian data ulasan pada *repository* dengan kategori aplikasi bergerak dan *repository* campuran, menunjukkan bahwa nilai *F-measure (F1)* dari data *repository* aplikasi bergerak, tidak menunjukkan hasil yang lebih baik dari data *repository* campuran, meskipun data *repository* aplikasi bergerak memiliki kategori yang sama dengan data ulasan yang terdapat pada toko aplikasi bergerak Android dan IOS. Hasil *F1* pada *repository* aplikasi bergerak sekitar 40-42% untuk tipe permintaan fitur, dan 70-72% untuk tipe laporan kesalahan. Sedangkan *F1* pada *repository* campuran sekitar 54% untuk tipe permintaan fitur, dan 71-74% untuk tipe laporan kesalahan.

Jika dilihat dari jumlah data yang dipakai dalam proses pelatihan, diketahui bahwa banyaknya data latih berpengaruh pada nilai akurasi dari model klasifikasi. *Bag of words unigram* menghasilkan akurasi yang lebih baik daripada *Bag of word bigram*. Hal ini kemungkinan terjadi karena panjang kalimat yang sedikit, sehingga kombinasi *bigram* pada data latih, tidak relevan dengan data uji.

Rata-rata panjang kalimat dari dataset dari *Github* dan toko aplikasi bergerak, berbeda cukup signifikan, yaitu sekitar 5-7 kata untuk dataset dari *Github*, dan 26-29 kata untuk dataset dari toko aplikasi bergerak. Selain itu, rata-rata panjang kalimat pada dataset dengan tipe *bug*, lebih panjang dari dataset dengan tipe *feature*, yaitu beda selisih 1-2 kata. Hal tersebut kemungkinan yang menyebabkan akurasi pada tipe laporan kesalahan lebih tinggi daripada permintaan fitur. Karena terdapat kemungkinan kosa-kata pada tipe *bug* lebih banyak daripada tipe *feature.*

Berdasarkan kata paling informatif, menunjukkan bahwa dataset dari *Github* memiliki kata-kata umum yang biasa dipakai pada ulasan perangkat lunak, ini menunjukkan bahwa dataset dari *Github* dapat dipakai sebagai dataset latih untuk proses klasifikasi pada ulasan perangkat lunak aplikasi bergerak Android dan IOS.

DAFTAR PUSTAKA

[1] A. Finkelstein *et al.*, "App Store Analysis : Mining App Stores for Relationships between Customer , Business and Technical Characteristics," *UCL Res. Note*, vol. 14/10, pp. 1–24, 2014.

[2] H. Li, L. Zhang, L. Zhang, and J. Shen, "A user satisfaction analysis approach for software evolution," *2010 IEEE Int. Conf. Prog. Informatics Comput.*, vol. 2, no. 2007, pp. 1093–1097, 2010.

[3] P. D. G. Putri and D. O. Siahaan, "EKSTRAKSI FITUR PRODUK DAN BUG POTENSIAL DARI DATA OPINI PENGGUNA," Institut Teknologi Sepuluh Nopember, 2016.

[4] W. Maalej and H. Nabil, "Bug Report , Feature Request , or Simply Praise ? On Automatically Classifying App Reviews," 2015.

[5] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," *2013 21st IEEE Int. Requir. Eng. Conf. RE 2013 - Proc.*, pp. 125–134, 2013.

[6] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requir. Eng.*, vol. 21, no. 3, pp. 311–331, 2016.

[7] M. Bano, "User Involvement in Software Development and System Success : A Systematic Literature Review," pp. 125–130, 2013.

[8] B. Kitchenham *et al.*, "Systematic literature reviews in software engineering – A tertiary study," *Inf. Softw. Technol.*, vol. 52, no. 8, pp. 792–805, 2010.

[9] V. V Das, "Involvement of Users in Software Requirement Engineering," pp. 230–233, 2007.

[10] S. A. Fricker, I. W. Conference, and D. Hutchison, *Requirements Engineering : Foundation*. 2015.

[11] K. Herzig, S. Just, and A. Zeller, "It ' s not a Bug , it ' s a Feature : How Misclassification Impacts Bug Prediction," no. Section XII.

[12] G. Antoniol, K. Ayari, M. Di Penta, and F. Khomh, "Is it a Bug or an Enhancement ? A Text-based Approach to Classify Change Requests."

[13] L. Li and X. Jin, "Multi-Domain Active Learning for Text Classification," 2012.

[14] Q. Lin, S. Liu, Z. Yang, A. Jami, and A. Saxena, "Cross-Domain Text Understanding in Online Social Data," pp. 1–6, 2014.

[15] D. Jurafsky and J. Martin, H, "Naive Bayes and Sentiment Classification," 2016.

[16] D. Mining, "Finding Informative Features 36-350:," vol. 1, no. September, pp. 1–12, 2009.