Terakreditasi SINTA Peringkat 3
Surat Keputusan Direktur Jenderal Pendidikan Tinggi, Riset, dan Teknologi Nomor 225/E/KPT/2022
masa berlaku mulai Vol.7 No. 1 tahun 2022 s.d Vol. 11 No. 2 tahun 2026

Terbit online pada laman web jurnal:
http://publishing-widyagama.ac.id/ejournal-v2/index.php/jointecs

JOINTECS

JOINTECS
(Journal of Information Technology and Computer Science)

Vol. x No. x (20xx) xx - xx
e-ISSN:2541-6448 p-ISSN:2541-3619

Pendekatan NLP untuk Deteksi Ambiguitas User Story dengan Syntactic Ambiguity dan Text Similarity

Defina Putri Alvira[1], Luvia Friska Narulita*[2]
[1]Teknik Informatika, Teknik, Universitas 17 Agustus 1945 Surabaya, [2]Sistem dan Teknologi Informasi, Teknik, Universitas 17 Agustus 1945 Surabaya
[1]definaputrialvira@gmail.com, [2] luvia@untag-sby.ac.id

Abstract

A user story serves as a means of communication for requirements in Agile software development. However, employing non-standardized natural language can create ambiguities in user stories. This research aims to develop a system that detects these ambiguities using a Natural Language Processing (NLP) approach, which incorporates text similarity analysis and syntactic ambiguity identification. The system assesses sentence structure, identifies non-atomic actions, and recognizes user stories with high semantic similarity. The detection outcomes are used to provide automated improvement recommendations. This study demonstrates that utilizing NLP techniques, specifically spaCy and BERT, effectively identifies potential ambiguities, which is expected to enhance the clarity of software requirements and minimize interpretation errors.

Keywords: User Story, Ambiguity, Natural Language Processing, Text Similarity, Syntactic Ambiguity

Abstrak

User story digunakan sebagai media komunikasi kebutuhan dalam pengembangan perangkat lunak berbasis Agile. Namun, penggunaan bahasa alami yang tidak terstandarisasi dapat menyebabkan ambiguitas dalam penulisan *user story*. Penelitian ini bertujuan untuk membangun sistem deteksi ambiguitas pada *user story* menggunakan pendekatan *Natural Language Processing* (NLP), yang menggabungkan analisis *text similarity* dan *syntactic ambiguity*. Sistem ini mengevaluasi struktur kalimat, mendeteksi keberadaan aksi ganda (*non-atomic*), serta mengidentifikasi *user story* yang memiliki kemiripan makna tinggi. Hasil deteksi digunakan untuk menghasilkan rekomendasi perbaikan secara otomatis. Studi ini menunjukkan bahwa penerapan metode NLP dengan spaCy dan BERT efektif dalam mengidentifikasi potensi ambiguitas, sehingga diharapkan dapat meningkatkan kejelasan kebutuhan perangkat lunak dan mengurangi risiko kesalahan interpretasi.

Kata kunci: User Story, Ambiguitas, Natural Language Processing, Text Similarity, Syntactic Ambiguity

CC BY SA

1. Pendahuluan

Dalam pengembangan perangkat lunak berbasis Agile, *user story* menjadi elemen penting dalam menyampaikan kebutuhan pengguna secara ringkas dan komunikatif. Format umum seperti "As a, I want, so that" dirancang untuk memfasilitasi komunikasi antara pemangku kepentingan dan tim pengembang tanpa memerlukan penjelasan teknis yang kompleks [1][2]. Namun, karena ditulis menggunakan bahasa alami yang bebas struktur, *user story* berisiko mengandung

Diterima Redaksi : xx-xx-20xx | Selesai Revisi : xx-xx-20xx | Diterbitkan Online : xx-xx-20xx

ambiguitas yang dapat menghambat pemahaman bersama dalam tim pengembang [3]

Ambiguitas dalam *user story* dapat bersifat sintaktik (struktur kalimat tidak lengkap), semantik (makna ganda), maupun fungsional (terdapat lebih dari satu aksi utama). Jika tidak ditangani sejak awal, ambiguitas tersebut dapat menyebabkan kesalahan implementasi, biaya tambahan, serta kualitas perangkat lunak yang rendah[4]. Studi-studi terkini menunjukkan bahwa pendekatan berbasis *Natural Language Processing* (NLP) mampu membantu mendeteksi potensi ambiguitas dalam teks kebutuhan perangkat lunak secara otomatis [5]. Salah satu pendekatan yang terbukti efektif adalah penggunaan representasi kontekstual berbasis transformer seperti BERT (*Bidirectional Encoder Representations from Transformers*) yang dapat memahami makna kalimat berdasarkan konteks secara mendalam[6], [7]. Penelitian ini jugayang menekankan pentingnya analisis linguistik terstruktur untuk mengidentifikasi potensi ambiguitas dalam *user story* serta kebutuhan akan dukungan otomatis berbasis NLP[8]. Penelitian ini bertujuan untuk mengembangkan sistem deteksi ambiguitas *user story* menggunakan pendekatan NLP yang menggabungkan analisis struktur sintaksis menggunakan spaCy dan representasi semantik dengan BERT. Sistem ini dirancang untuk mendeteksi struktur yang tidak lengkap, aktivitas non-atomik, serta kemiripan makna antardokumen, dan memberikan rekomendasi perbaikan dalam bentuk naratif otomatis

2. Metode Penelitian

Penelitian ini menggunakan pendekatan rekayasa perangkat lunak berbasis analisis linguistik dan pembelajaran mesin untuk mendeteksi ambiguitas pada user story. Terdapat lima tahapan utama dalam metode ini, mulai dari pengumpulan data hingga pemberian rekomendasi[5].

2.1. Pengumpulan Data dan Pra-pemrosesan

Dataset *user story* diperoleh dari berbagai repositori terbuka seperti Zenodo dan Kaggle, yang menyediakan deskripsi *user story* dalam format tabel (.xlsx) sebelum kemudian dikonversi ke bentuk teks [9] [10], [11]. Pada tahap awal, data melalui proses pra-pemrosesan menggunakan teknik seperti normalisasi (mengubah semua huruf menjadi huruf kecil), tokenisasi, penandaan *part-of-speech* (POS), *dependency parsing*, serta penghapusan *stopword* dengan bantuan pustaka spaCy, guna menyederhanakan struktur kalimat dan mempermudah analisis berikutnya [12], [13], [14], [15]. Sistem ini juga menerapkan pendekatan linguistik lanjutan, termasuk segmentasi kalimat berdasarkan elemen WHO-WHAT-WHY dan evaluasi kualitas *atomicity*, untuk mengurangi kemungkinan ambiguitas struktural yang muncul dari kalimat majemuk atau penggunaan konjungsi dengan lebih dari satu aksi utama [8], [10], [16], [17]. Analisis sintaktik ini turut mempertimbangkan struktur *dependency* serta identifikasi kata kerja utama dan frasa penghubung, yang sering menjadi penyebab ambiguitas dalam penulisan *user story* [10], [18], [19].

```
Program Jurnal
Input: user_stories
Output: result_dataframe

Inisialisasi embeddings
embeddings ← BERT(user_stories)

For i ← 1 to length(user_stories) do
    who,what,why ← spaCy(user_stories[i])
    similarity←
cosine_similarity(embeddings[i],
embeddings[j])
    if similarity ≥ 0.7 then
        Tambahkan ke daftar kemiripan
    end if
    Struktur ← deteksi struktur kalimat
    Saran ← hasil rekomendasi
End for

Return dataframe hasil
```

2.2. Ekstraksi Fitur dan Deteksi Ambiguitas

Setiap *user story* yang telah melalui proses pra-pemrosesan kemudian diubah menjadi vektor embedding menggunakan model BERT [6], [7]. Vektor embedding ini dimanfaatkan untuk menghitung tingkat kemiripan antar *user story* menggunakan metode *cosine similarity* [4]. Jika skor kemiripan melebihi ambang batas 0,7, maka user story tersebut dianggap berpotensi memiliki *ambiguity similarity*. Nilai *threshold* ini ditentukan berdasarkan analisis distribusi skor kemiripan dalam dataset, dengan tujuan menjaga sensitivitas sistem terhadap makna yang serupa tanpa menimbulkan terlalu banyak hasil kemiripan palsu (*false positive*) [20], [21]. Secara bersamaan, sistem juga menjalankan analisis sintaksis untuk mengidentifikasi potensi ambiguitas struktural, termasuk pendeteksian jumlah kata kerja utama untuk menilai aspek *non-atomic*, identifikasi konjungsi yang menghubungkan dua aksi, serta verifikasi kelengkapan struktur “who-what-why” [8], [12], [16], [22]. Strategi ini merujuk pada pendekatan linguistik dalam mendeteksi ambiguitas yang telah dikaji dalam berbagai penelitian sebelumnya [10], [16], [18], [23], [24], [25].

2.3. Rekomendasi Perbaikan.

Jika ditemukan indikasi ambiguitas, sistem akan memberikan rekomendasi perbaikan secara otomatis, seperti: menyarankan pemisahan aksi ganda, penambahan elemen yang hilang, atau penyusunan ulang struktur kalimat. Rekomendasi ditampilkan dalam format naratif berbasis template yang memudahkan pemahaman pengguna non-teknis [2], [26]. Sistem ini meniru prinsip sistem semi-otomatis

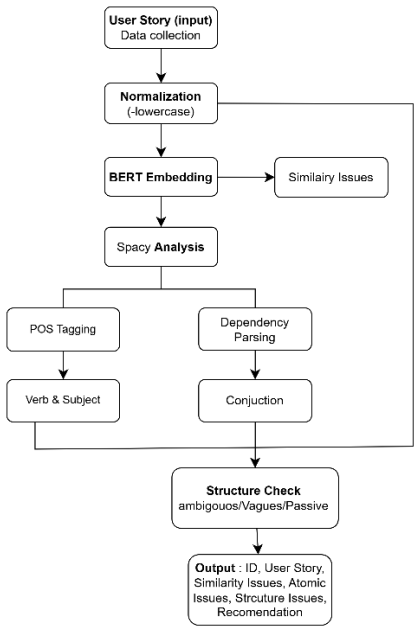
yang telah diterapkan pada proyek serupa dalam ranah analisis kebutuhan perangkat lunak [27][8].

Tabel 1.Tabel Software Pendukung

Komponen	Teknologi
Bahasa Pemrograman	Python
Library NLP	spaCy
Model Embedding	BERT-base-uncased
Framework Web	Flask
Lingkungan Pengujian	Jupyter Notebook

Pemilihan spaCy dan BERT dalam penelitian ini didasarkan pada efektivitas keduanya dalam memproses teks bahasa alami, khususnya dalam konteks bahasa Inggris. spaCy dipilih karena memiliki kemampuan parsing sintaksis yang andal serta efisien dalam melakukan analisis struktur kalimat seperti tokenisasi, *POS tagging*, dan *dependency parsing* [9], [12]. Sementara itu, BERT (*Bidirectional Encoder Representations from Transformers*) mampu menghasilkan representasi kontekstual yang mendalam dari teks melalui mekanisme *self-attention*, sehingga memungkinkan sistem mengenali makna kalimat dengan akurasi tinggi [6], [7], [28]. Kombinasi kedua alat ini telah terbukti efektif dalam berbagai studi NLP untuk mendeteksi ambiguitas dan meningkatkan pemahaman semantik dalam dokumen berbasis bahasa alami[3].

2.4. Visualisasi Sistem



Gambar 1. Alur Pra-pemrosesan dan Analisis Sintaksis User Story

Penelitian ini dimulai dengan tahap pengumpulan data berupa *user story*, yang diperoleh dari *backlog* proyek, wawancara dengan pemangku kepentingan, dan dokumen kebutuhan sistem[11], [29]. Setelah data dikumpulkan, dilakukan proses normalisasi dengan mengonversi seluruh teks ke huruf kecil (lowercase) untuk menjaga konsistensi dalam proses analisis berikutnya [12]. Selanjutnya, *user story* diubah menjadi representasi vektor menggunakan BERT embedding [6], [7], [9], yang memungkinkan sistem untuk mengukur kesamaan makna antar *user story* dan mendeteksi kemungkinan duplikasi atau kemiripan [30]. Secara paralel, sistem juga menjalankan analisis linguistik dengan menggunakan spaCy, termasuk *POS tagging* untuk mengidentifikasi subjek dan kata kerja utama, serta *dependency parsing* untuk memetakan hubungan antar kata dan mengenali konjungsi yang mengindikasikan adanya lebih dari satu aksi [16], [22], [31]. Hasil dari kedua jenis analisis ini dimanfaatkan dalam proses *structure checking*, yang bertujuan mengidentifikasi potensi permasalahan seperti penggunaan kata ambigu, istilah yang tidak spesifik (vague), dan kalimat pasif [10], [25]. Semua temuan ini dirangkum dalam output yang berisi konten *user story*, isu-isu yang ditemukan, serta rekomendasi perbaikannya [29].

2.5. Jenis Ambiguitas Dan Strategi Deteksi

Ambiguitas merupakan salah satu tantangan utama dalam rekayasa kebutuhan perangkat lunak, termasuk pada penulisan *user story*. Menurut [32], ambiguitas didefinisikan sebagai kata atau ungkapan yang dapat ditafsirkan dalam dua atau lebih cara yang berbeda. Dalam konteks *Requirements Engineering* (RE), ambiguitas sering disebut sebagai “tumit Achilles” dari dokumen kebutuhan karena dapat menyebabkan perbedaan ekspektasi maupun implementasi yang tidak sesuai akibat interpretasi yang bervariasi [16], [19]. juga menegaskan bahwa ambiguitas tetap dapat muncul meskipun pembaca memiliki pemahaman yang baik terhadap konteks [16], [19], [33].

Tabel 2.1 Tabel contoh epik dan user story

Epic	As a user, I want to view detailed information about the company offering the job, so that I can assess whether the company is a good fit for me.
User Story 1	As a job seeker, I want to view th company’s profile details so that I ca understand their background and industr focus.
User Story 2	As a job seeker, I want to read employee reviews about the company so that I can evaluate their work culture.

Dalam *user story*, ambiguitas biasanya dianalisis dari dua perspektif: linguistik dan rekayasa perangkat lunak[16], [17], [33] . Pendekatan linguistik mencakup isu-isu seperti ambiguitas leksikal, sintaktis, dan ketidakkonsistenan dalam penggunaan format template [15], [16], [33] sementara perspektif rekayasa perangkat lunak lebih menyoroti perbedaan pemahaman terhadap istilah dan konteks di antara

anggota tim pengembang yang berasal dari latar belakang yang beragam [17], [19].

Penelitian ini mengidentifikasi dua kategori utama ambiguitas yang dapat dideteksi secara otomatis: *Ambiguitas similarity* ini terjadi ketika dua *user story* yang berbeda memiliki makna semantik yang sangat serupa. Jenis ambiguitas ini bersifat implisit dan sulit dikenali tanpa bantuan sistem, sebagaimana dijelaskan oleh [16], [17], [33]. Proses deteksi dilakukan melalui representasi vektor menggunakan BERT dan pengukuran *cosine similarity*[6], [7]. Jika nilai kesamaan ≥ 0,7, maka user story tersebut dianggap memiliki potensi duplikasi makna[20]. Ambiguitas sintaktik muncul akibat struktur kalimat yang memungkinkan berbagai interpretasi gramatikal. Penelitian ini membaginya ke dalam dua subkategori, berdasarkan klasifikasi [16]

Atomic adalah kriteria kualitas yang mengharuskan sebuah *user story* ditulis sebagai kalimat sederhana, dengan satu aktor dan satu permintaan aksi terhadap satu objek tertentu. Tujuannya agar setiap *user story* hanya menghasilkan satu *parse tree* sehingga meminimalkan kemungkinan munculnya interpretasi berbeda akibat kalimat majemuk atau kompleks[16], [19]. Dalam praktik agile, kriteria ini mendukung prinsip "Small" dalam INVEST [12]. Kesalahan terhadap kriteria atomic umumnya terjadi akibat penggunaan konjungsi (seperti "and") di segmen WHAT, yang membuat satu *user story* memuat lebih dari satu aksi atau objek. Hal ini jarang terjadi di segmen WHO, tetapi sangat sering muncul di WHAT dan WHY. Untuk menilai apakah konjungsi menimbulkan ambiguitas, analisis harus memeriksa relasi antar aksi dan keterkaitannya dengan tujuan (WHY).

Tabel 2.2 Tabel ilustrasi non-atomic dapat memicu ambiguitas sintaksis.

US^1	*As a content manager, I want to publish new company updates, remove outdated announcements, and refresh the employee recognition section, so that visitors always see relevant and current information.*
US^2	*As a content manager, I want to publish the latest company milestones, remove inactive job listings, and refresh the gallery section, so that the website reflects our most current status.*
US^3	*As a content manager, I want to edit the CEO's welcome message, archive past newsletters, and highlight current projects on the homepage, so that visitors always see up-to-date and relevant content.*

Poor structure adalah salah satu indikator ambiguitas dalam *user story* yang muncul ketika segmen aksi (WHAT) mencakup syarat atau kondisi eksplisit yang seharusnya diletakkan di bagian motivasi (WHY) [16], [17]. Penggunaan konjungsi subordinatif seperti *after*, *when*, atau *if* dalam segmen WHAT berpotensi menimbulkan ambiguitas sintaktik karena mengaburkan batas antara aksi utama dan kondisi yang mendahuluinya [19], [33]. Kriteria ini sejalan dengan prinsip verifiabilitas dalam kerangka Agile *Requirements Verification Framework*, serta berkaitan dengan prinsip INVEST yang menekankan pada kejelasan dan dapat diuji [22], [34]. Meskipun terdapat diskusi mengenai pemenuhan seluruh aspek SMART atau INVEST secara bersamaan, menjaga konsistensi struktur tetap menjadi kunci untuk meningkatkan kualitas dan keterukuran *user story* [24].

Tabel 2.3 Tabel ilustrasi poor structure dapat memicu ambiguitas sintaksis.

US^1	*As a content manager, I want to archive old content when it's no longer needed, so that the website can be better.*
US^2	*As a content manager, I want to change the header image sometimes, so that people don't get bored.*
US^3	*As a content manager, I want to organize everything on the homepage, so that the page feels more organized.*

Pendekatan ini merujuk pada klasifikasi dari [33]serta didukung oleh pemrosesan NLP menggunakan spaCy dan representasi semantik berbasis BERT untuk mendeteksi serta memberikan saran perbaikan bagi *user story* yang mengandung ambiguitas.

3. Hasil dan Pembahasan

Penelitian ini mengkaji tiga komponen utama dalam sistem deteksi ambiguitas pada *user story*, yakni *similarity*, *atomicity*, dan *structure*. Evaluasi dilakukan dengan menghitung metrik seperti *precision*, *recall*, dan *F1-score* untuk masing-masing jenis isu, serta menganalisis fluktuasi nilai antar dataset guna menilai kestabilan performa sistem. Tujuan dari proses ini adalah untuk mengetahui sejauh mana sistem mampu mengenali potensi ambiguitas yang bervariasi tergantung pada konteks dan gaya penulisan *user story*.

Sebanyak 1034 *user story* digunakan dalam pengujian untuk memastikan sistem diuji dalam berbagai situasi representatif—mulai dari *user story* yang mengikuti pola template hingga gaya penulisan naratif yang lebih bebas. Evaluasi ini tidak hanya menitikberatkan pada kinerja numerik, tetapi juga mengeksplorasi pola prediksi yang tepat maupun keliru, serta mengidentifikasi kemungkinan bias terhadap jenis ambiguitas tertentu.

3.1. Penyajian Hasil Pengujian Sistem

Pengujian dilakukan untuk menilai kinerja sistem dalam mengenali tiga tipe ambiguitas pada *user story*,

yaitu *similarity*, *atomicity*, dan *structure*. Setiap tipe isu dianalisis menggunakan *confusion matrix*, yang mencakup empat parameter utama: *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Data pengujian dikumpulkan dari 1034 *user story* yang dirancang untuk mewakili variasi gaya penulisan, struktur kalimat, serta konteks penggunaan di dunia nyata. Secara keseluruhan, terdapat **3.327** *instance* hasil prediksi sistem yang dibandingkan dengan label manual (*ground truth*).

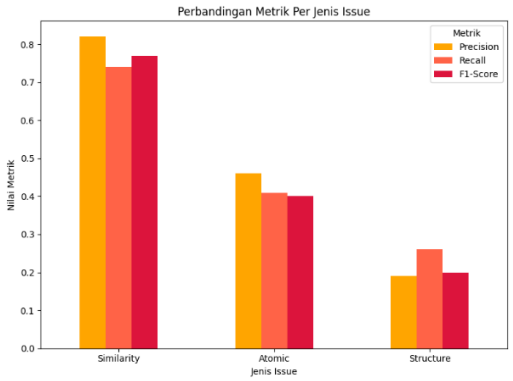
Berdasarkan hasil tersebut, sistem memperoleh akurasi total sebesar **60,65%**, yang dihitung dengan rumus berikut:

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN}$$

Untuk memberikan gambaran yang lebih menyeluruh, dilakukan pula penghitungan nilai *precision*, *recall*, dan *F1-score* pada masing-masing kategori ambiguitas. Nilai-nilai tersebut ditampilkan dalam bentuk **grafik batang dan garis** untuk mempermudah perbandingan performa tiap kategori secara kuantitatif serta untuk menunjukkan kestabilan sistem dalam berbagai variasi dataset.

3.2. Analisis Performa Berdasarka Kategori Ambiguitas

Secara keseluruhan, sistem mencatat performa terbaik pada deteksi isu *similarity*. *Precision* sebesar 0,82 menunjukkan bahwa sebagian besar prediksi positif tepat sasaran, sedangkan *recall* 0,74 mengindikasikan kemampuan sistem dalam menangkap mayoritas kasus similarity yang relevan. Nilai *F1-score* sebesar 0,77 mencerminkan keseimbangan antara ketepatan dan cakupan prediksi. Untuk isu *atomicity*, performa sistem menurun dengan *precision* 0,46, recall 0,41, dan *F1-score* 0,40. Hal ini menunjukkan ketidakkonsistenan dalam mengenali keberadaan lebih dari satu aksi dalam satu *user story*, kemungkinan disebabkan oleh beragamnya cara penyampaian aksi, baik secara eksplisit maupun implisit. Deteksi isu *structure* menunjukkan kinerja paling lemah. *Precision* rendah (0,19) dan *recall* yang juga rendah (0,26) menandakan bahwa sistem sering salah mengklasifikasikan serta gagal mendeteksi struktur kalimat yang bermasalah. Nilai *F1-score* 0,20 memperkuat indikasi bahwa model kesulitan dalam mengenali pola struktur yang tidak eksplisit atau tidak sesuai format baku.



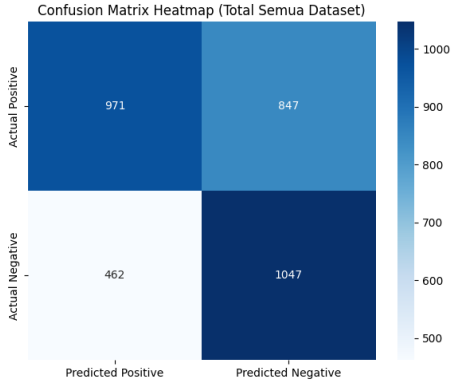
Gambar 2. Perbandingan Precision, Recall, dan F1-score

Secara umum, sistem lebih unggul dalam menangani isu berbasis **semantik** dibanding **sintaktik**, selaras dengan karakteristik model BERT yang lebih kuat dalam memahami konteks makna daripada menganalisis pola struktural yang kompleks.

3.3. Perbandingan Visual Kinerja Deteksi Tiap Kategori

Confusion matrix hasil perbanding antar kategori, menunjukkan bagaimana sistem mengklasifikasikan 3.327 *user story* yang diuji. Dari jumlah tersebut, sistem berhasil mengenali 971 instance sebagai positif secara tepat, dan 1.047 instance sebagai negatif yang memang tidak mengandung isu. Dengan kata lain, sekitar 60% dari hasil prediksi sesuai dengan kondisi sebenarnya. Namun, satu hal yang cukup mencolok adalah jumlah **false negative** yang tinggi, yaitu sebanyak 847 kasus. Ini berarti masih banyak *user story* yang sebenarnya bermasalah tetapi tidak terdeteksi oleh sistem. Hal ini kemungkinan besar terjadi pada kasus-kasus ambiguitas yang tidak disampaikan secara eksplisit—misalnya pada kalimat yang panjang, bersifat naratif, tidak mengikuti struktur template, atau menyisipkan tujuan secara tidak langsung.

Selain itu, tercatat sebanyak 462 **false positive**, di mana sistem menganggap user story bermasalah padahal tidak. Hal ini menunjukkan bahwa sistem terkadang terlalu sensitif terhadap pola tertentu, seperti penggunaan konjungsi atau ungkapan yang umum ditemukan pada kalimat ambigu.

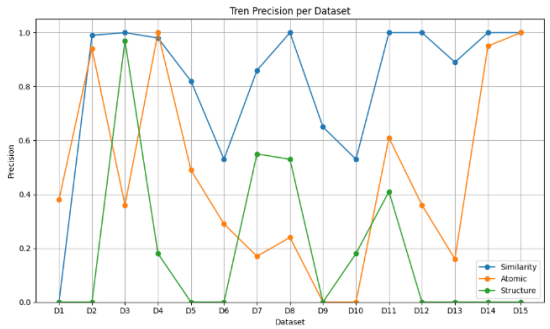


Gambar 3. Distribusi Klasifikasi Berdasarkan Confusion Matrix

Dari hasil ini dapat disimpulkan bahwa sistem sudah cukup mampu mendeteksi sebagian besar *user story* yang bermasalah, terutama yang ditulis dengan struktur yang jelas. Meski demikian, masih diperlukan peningkatan agar sistem dapat menangkap ambiguitas yang tersirat tanpa mudah terjebak oleh pola-pola yang sebenarnya masih tergolong valid.

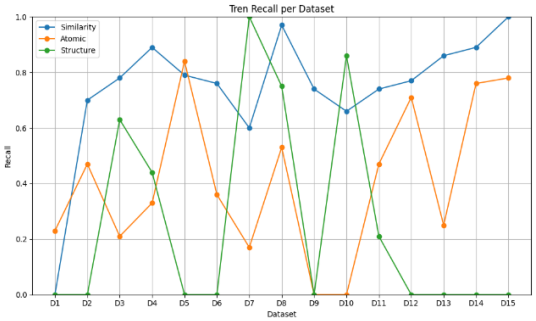
3.4 Tren Evaluasi per Dataset

Tren *precision* per dataset memperlihatkan bahwa sistem paling konsisten dalam mendeteksi isu *similarity*, dengan nilai yang tinggi dan stabil di sebagian besar dataset. Sebaliknya, *precision* untuk *atomicity* dan *structure* cenderung fluktuatif dan sering kali rendah. Hal ini menunjukkan bahwa pendekatan berbasis semantik (seperti BERT) cukup efektif untuk mendeteksi kemiripan makna, namun masih ada keterbatasan dalam mengenali struktur kalimat dan aksi ganda, terutama ketika *user story* ditulis dalam format yang tidak eksplisit. Temuan ini menjadi dasar penting untuk pengembangan lanjutan, khususnya pada aspek analisis sintaksis dan pemrosesan kalimat kompleks.



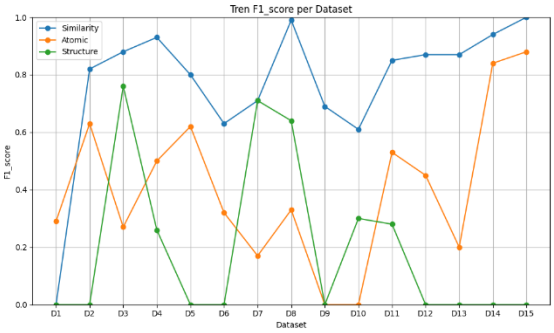
Gambar 4. Grafik Tren Precision per Dataset

Tren *recall* per dataset menunjukkan bahwa sistem paling andal dalam mengenali isu *similarity*, dengan nilai *recall* yang cenderung tinggi dan stabil dari D2 hingga D15. Untuk *atomicity*, performa cukup beragam, dengan puncak recall di D5, D6, dan D14, namun sangat rendah di dataset lainnya. Sementara itu, *structure* mencatat recall yang paling tidak stabil, bahkan mencapai titik nol di beberapa dataset. Temuan ini menegaskan bahwa sistem masih kesulitan menangkap ambiguitas struktur yang bersifat implisit atau tidak mengikuti pola baku. Diperlukan pendekatan tambahan agar deteksi struktur lebih adaptif terhadap variasi gaya penulisan *user story*.



Gambar 5. Grafik Tren Recall per Dataset

Tren *F1-score* memperlihatkan bahwa performa deteksi isu *similarity* cenderung stabil dan mendekati optimal di hampir seluruh dataset. Nilai F1-score yang tinggi ini menunjukkan keseimbangan antara *precision* dan *recall* dalam mengenali *user story* yang bermakna serupa. Sementara itu, performa pada isu *atomicity* masih sangat bervariasi, dengan beberapa dataset menunjukkan hasil baik (misalnya D2, D5, D14), namun menurun drastis di dataset lainnya. Untuk *structure*, *F1-score* sangat rendah dan tidak stabil, menandakan bahwa sistem belum cukup mampu mengatasi variasi struktur kalimat yang tidak eksplisit. Hal ini mengindikasikan perlunya peningkatan pada strategi deteksi berbasis sintaks dan rule adaptif.



Gambar 6. Grafik Tren F1-Score per Dataset

3.5 Hasil Analisis Sistem Deteksi Ambiguitas

Salah satu hasil analisis sistem menunjukkan bahwa *user story* dengan ID **US-1** terdeteksi memiliki **isu** *similarity*, yaitu tingkat kemiripan yang tinggi dengan sejumlah user story lain, seperti US-5 (skor 0.92), US-52, US-9, hingga US-10 dan lainnya. Meskipun tidak teridentifikasi adanya masalah *atomicity* maupun *structure*, kemiripan yang signifikan ini dapat menandakan potensi duplikasi atau tumpang tindih fungsi dalam backlog. Oleh karena itu, disarankan dilakukan peninjauan ulang untuk memastikan keunikan dan kejelasan tiap kebutuhan.

4. Kesimpulan

Penelitian ini berhasil membangun sistem deteksi ambiguitas pada *user story* menggunakan pendekatan *Natural Language Processing* (NLP) dengan kombinasi analisis sintaksis dan kemiripan makna. Sistem yang dikembangkan mampu mengidentifikasi struktur *user story* yang tidak lengkap, mendeteksi keberadaan lebih

dari satu aksi utama (*non-atomic*), serta mengenali kemiripan makna antar *user story*. Proses ini memungkinkan dihasilkannya rekomendasi perbaikan secara otomatis dalam bentuk narasi yang mudah dipahami. Sistem ini berpotensi diimplementasikan dalam proses penulisan kebutuhan perangkat lunak berbasis *Agile* untuk membantu *product owner* dan analis dalam menjaga kejelasan dokumentasi. Selain itu, sistem dapat dikembangkan lebih lanjut agar mendukung analisis terhadap *user story* dalam bahasa lain atau diperluas cakupannya untuk kebutuhan teknis di luar *Agile*. Sistem mendeteksi isu *similarity* terbaik dengan *F1-score* 0,77 dari 3.327 cerita pengguna. Ini menunjukkan bahwa model efektif dalam memahami kesamaan makna antar kalimat. Sebaliknya, performa pada masalah *atomicity* (F1-score 0,40) dan *structure* (F1-score 0,20) masih rendah, menunjukkan bahwa sistem sulit untuk memeriksa pola sintaktik yang kompleks dan struktur kalimat yang tidak jelas. Secara keseluruhan, temuan penelitian ini menunjukkan bahwa metode berbasis semantik lebih baik daripada metode sintaktik dalam mendeteksi ambiguitas makna. Sistem ini dapat digunakan sebagai alat bantu untuk meningkatkan dokumentasi kebutuhan perangkat lunak di lingkungan *Agile*. Pengembangan lebih lanjut dapat dilakukan untuk memperluas cakupan bahasa (misalnya, Bahasa Indonesia), menangani input panjang yang mencakup lebih dari 512 token, dan meningkatkan deteksi struktur dengan menggunakan pendekatan berbasis aturan dan pembelajaran semantik yang lebih mendalam dengan dukungan data yang lebih bervariasi.

Daftar Pustaka

[1] A. Amna, “Identifying Ambiguity Problems in User Stories : A Proposed Framework,” Leuven,Belgium, 2022.

[2] M. A. Kuhail and S. Lauesen, “User Story Quality in Practice: A Case Study,” *Software*, vol. 1, no. 3, pp. 223–243, Jun. 2022, doi: 10.3390/software1030010.

[3] S. Ezzini, S. Abualhaija, C. Arora, and M. Sabetzadeh, “TAPHSIR: Towards AnaPHoric Ambiguity Detection and ReSolution In Requirements,” 2022, doi: 10.5281/zenodo.5902117.

[4] S. Agarwala, A. Anagawadi, and R. M. Reddy Guddeti, “Detecting Semantic Similarity of Documents Using Natural Language Processing,” in *Procedia CIRP*, Elsevier B.V., 2021, pp. 128–135. doi: 10.1016/j.procs.2021.05.076.

[5] I. K. Raharjana, D. Siahaan, and C. Fatichah, “User Stories and Natural Language Processing: A Systematic Literature Review,” *IEEE Access*, vol. 9, pp. 53811–53826, 2021, doi: 10.1109/ACCESS.2021.3070606.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Oct. 2019, [Online]. Available: http://arxiv.org/abs/1810.04805

[7] H. Zhang and M. O. Shafiq, “Survey of transformers and towards ensemble learning using transformers for natural language processing,” *J Big Data*, vol. 11, no. 1, Dec. 2024, doi: 10.1186/s40537-023-00842-0.

[8] A. R. Amna, *Enhancing requirements clarity : analyzing potential ambiguity in user stories*. Ghent, Belgium: Ghent University. Faculty of Economics and Business Administration, 2024.

[9] R. Sonbol, G. Rebdawi, and N. Ghneim, “The Use of NLP-Based Text Representation Techniques to Support Requirement Engineering Tasks: A Systematic Mapping Review,” Apr. 22, 2022, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2022.3182372.

[10] M. Urbieta, L. Antonelli, G. Rossi, and J. C. S. do Prado Leite, “The impact of using a domain language for an agile requirement management,” *Inf Softw Technol*, vol. 127, Nov. 2020, doi: 10.1016/j.infsof.2020.106375.

[11] F. Gilson and C. Irwin, “From user stories to use case scenarios towards a generative approach,” in *Proceedings - 25th Australasian Software Engineering Conference, ASWEC 2018*, Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 61–65. doi: 10.1109/ASWEC.2018.00016.

[12] A. R. Amna, Y. Wautelet, S. Poelmans, S. Heng, and G. Poels, “The AmbiTRUS Framework for Identifying Potential Ambiguity in User Stories,” 2022. [Online]. Available: https://ssrn.com/abstract=4890812

[13] R. Sonbol, G. Rebdawi, and N. Ghneim, “The Use of NLP-Based Text Representation Techniques to Support Requirement Engineering Tasks: A Systematic Mapping Review,” 2022, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2022.3182372.

[14] M. Trkman, J. Mendling, P. Trkman, and M. Krisper, “Impact of the conceptual model’s

representation format on identifying and understanding user stories," *Inf Softw Technol*, vol. 116, Dec. 2019, doi: 10.1016/j.infsof.2019.08.001.

[15] Y. Wautelet, S. Heng, M. Kolp, and I. Mirbel, "Unifying and extending user story models," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2014, pp. 211–225. doi: 10.1007/978-3-319-07881-6_15.

[16] E. Kamsties, "Understanding Ambiguity in Requirements Engineering."

[17] F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis, "Identifying Nocuous Ambiguities in Natural Language Requirements," 2006.

[18] A. ZEAARAOUI, *User Stories Template for Object-Oriented Applications*. MAROCCO: IEEE, 2013.

[19] V. Gervasi, A. Ferrari, D. Zowghi, and P. Spoletini, "Ambiguity in Requirements Engineering: Towards a Unifying Framework."

[20] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Aug. 2019, [Online]. Available: http://arxiv.org/abs/1908.10084

[21] T. Ambreen, N. Ikram, M. Usman, and M. Niazi, "Empirical research in requirements engineering: trends and opportunities," *Requir Eng*, vol. 23, no. 1, pp. 63–95, Mar. 2018, doi: 10.1007/s00766-016-0258-2.

[22] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Improving agile requirements: the Quality User Story framework and tool," *Requir Eng*, vol. 21, no. 3, pp. 383–403, Sep. 2016, doi: 10.1007/s00766-016-0250-x.

[23] M. Elallaoui, K. Nafil, and R. Touahni, "Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques," in *Procedia Computer Science*, Elsevier B.V., 2018, pp. 42–49. doi: 10.1016/j.procs.2018.04.010.

[24] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "The use and effectiveness of user stories in practice," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2016, pp. 205–222. doi: 10.1007/978-3-319-30282-9_14.

[25] J. Melegati, "QUESt: New Practices to Represent Hypotheses in Experiment-Driven Software Development," 2019, doi: 10.1145/3340481.

[26] A. Shaaban, H. Elazhary, and E. Hassanein, "An automated tool for detection and tutoring of sources of ambiguities in requirements documents," *International Journal of Recent Technology and Engineering*, vol. 8, no. 3, pp. 2371–2375, Sep. 2019, doi: 10.35940/ijrte.C4660.098319.

[27] L. Müter, T. Deoskar, M. Mathijssen, S. Brinkkemper, and F. Dalpiaz, "Refinement of User Stories into Backlog Items: Linguistic Structure and Action Verbs: Research Preview," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, Mar. 2019, pp. 109–116. doi: 10.1007/978-3-030-15538-4_7.

[28] M. Koroteev, "BERT: A Review of Applications in Natural Language Processing and Understanding," Mar. 2021, doi: 10.48550/arXiv.2103.11943.

[29] D. Hallmann, "'I Don't Understand!': Toward a Model to Evaluate the Role of User Story Quality," in *Lecture Notes in Business Information Processing*, Springer, 2020, pp. 103–112. doi: 10.1007/978-3-030-49392-9_7.

[30] T. R. Silva and B. Fitzgerald, "Empirical findings on bdd story parsing to support consistency assurance between requirements and artifacts," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jun. 2021, pp. 266–271. doi: 10.1145/3463274.3463807.

[31] H. Zhou, Y. Zhang, Z. Li, and M. Zhang, "Is POS Tagging Necessary or Even Helpful for Neural Dependency Parsing?," Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.03204

[32] M. Webster, "Ambiguity definition."

[33] D. M. Berry and E. Kamsties, "Chapter 2 AMBIGUITY IN REQUIREMENTS SPECIFICATION," 2004.

[34] L. Buglione and A. Abran, "Improving the user story Agile technique using the INVEST criteria," in *Proceedings - Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, IWSM-MENSURA 2013*, IEEE Computer Society, 2013, pp. 49–53. doi: 10.1109/IWSM-Mensura.2013.18.