Abstrak

Menu tanya jawab sering kali menampilkan informasi sebagai pertanyaan umum tentang layanan dan produk perusahaan. Selain itu, terkadang pengguna harus menelusuri semua halaman untuk mendapatkan informasi yang mereka butuhkan, yang memakan waktu lama. Alternatif yang ditawarkan perusahaan perbankan seperti BRI adalah Sabrina yaitu chatbot yang dapat menjawab pertanyaan seputar produk BRI. Namun, chatbot dibuat dengan metode dialog decision sehingga tidak memiliki pengetahuan terhadap sentimen kalimat. Metode tradisional seperti word2vec yang langsung melakukan training data tidak lagi efisien karna representasi terhadap kalimatnya yang terbatas. Oleh karena itu dibutuhkan suatu metode yang dapat menghubungkan kata antar kata agar lebih bisa dimengerti oleh model yaitu memakai graph-based representation. Data yang diubah akan diproses dengan model berbasis graph. Oleh karena itu, otomatisasi ini diperlukan untuk dapat mengetahui kinerja chatbot sekaligus memberikan pengetahuan sentiment melalui percakapannya dengan user. Dalam penelitian ini, diterapkan tiga model rekomendasi berbasis graf dan tiga model rekomendasi berbasis pohon. Model berbasis graf yang digunakan adalah GCN, Chebycev-Graph dan Transformer-Graph. Model berbasis pohon yang digunakan adalah Random Forest, LightGBM dan XGBoost. Dari keenam model yang dibuat, dilakukan perbandingan terhadap performa dan waktu inferensi. Hasil eksperimen menunjukkan bahwa model rekomendasi berbasis graf menghasilkan nilai Akurasi mirip dengan XGBoost yaitu 0,7173. Pada sisi waktu inferensi model berbasis graph lebih cepat 3 kali lipat daripada model berbasis pohon.

Kata kunci : Chatbot, Perbankan, Graph

Abstract

The frequently asked questions menu displays information as frequently asked questions about the company's services and products. Also, sometimes users have to browse through all the pages to find the information they need, which takes a long time. An alternative offered by banking companies such as BRI is Sabrina, a chatbot that can answer questions about BRI products. However, the chatbot is made using the decision dialogue method so it has no knowledge of sentence sentiment. Traditional methods such as word2vec which directly perform data training are no longer efficient because of the limited representation of sentences. Therefore we need a method that can connect words between words so that the model can understand it better, namely using graph-based representation. Changed data will be processed with a graph-based model. Therefore, this automation is needed to be able to find out the performance of chatbots while at the same time providing sentiment knowledge through conversations with users. In this study, three graph-based recommendation models and three tree-based recommendation models were applied. The graph-based models used are GCN, Chebycev-Graph and Transformer-Graph. The tree-based models used are Random Forest, LightGBM and XGBoost. Of the six models made, a comparison was made of performance and inference time. The experimental results show that the graph-based recommendation model produces an Accuracy value similar to that of XGBoost, namely 0.7173. In terms of inference time, graph-based models are 3 times faster than tree-based models.**Keywords: Chatbots, Banking, Graph**

1. Pendahuluan

Teknologi World Wide Web telah berkembang pesat, menciptakan revolusi dalam pertukaran informasi. Salah satunya adalah penyediaan menu tanya jawab di situs perusahaan. menu tanya jawab sering kali menampilkan informasi sebagai pertanyaan umum tentang layanan dan produk perusahaan. Namun, menu tanya jawab sering kali berisi terlalu banyak informasi karena mencakup semua informasi produk secara detail. Hal ini membuat mengakses informasi kurang interaktif dan lebih nyaman. Selain itu, terkadang pengguna harus menelusuri semua halaman untuk mendapatkan informasi yang mereka butuhkan, yang memakan waktu lama. Alternatif yang ditawarkan perusahaan adalah menawarkan layanan tanya jawab online, seperti kotak obrolan. Namun, jenis fungsi ini membutuhkan lebih banyak staf untuk menjawab pertanyaan pengguna individu. Masalah muncul ketika jumlah pengguna yang ingin mendapatkan informasi sangat banyak, tetapi jumlah karyawan terbatas, menyebabkan banyak pengguna yang tidak bertanya-tanya. Oleh karena itu, otomatisasi ini diperlukan untuk menjawab pertanyaan pengguna tentang informasi yang dibutuhkan. Oleh karena itu, chatbots dapat digunakan sebagai alternatif untuk mengatasi masalah ini. [1]

Chatbots telah banyak digunakan dalam banyak hal termasuk hiburan, pendidikan, perjalanan, dan sebaginya. Chatbot dapat digunakan sebagai alat untuk mempelajari bahasa baru, alat untuk mengakses sistem informasi, alat untuk memvisualisasikan konten korpus, dan alat untuk menjawab pertanyaan, pertanyaan tentang domain tertentu, dan dapat dilatih dalam beberapa bahasa. [2]

Di bidang perbankan, beberapa penelitian telah dilakukan untuk mengembangkan chatbots dalam menyediakan informasi perbankan. Chatbot telah dikembangkan di sektor perbankan menggunakan pemrosesan bahasa alami. Chatbots juga digunakan di area lain, seperti memberikan bimbingan belajar untuk siswa, layanan konsultasi, layanan pengetahuan modular, penyedia layanan humor, dan banyak lagi.[3]

BRI adalah salah satu perusahaan yang menawarkan Chatbots di situs web mereka yang menampilkan informasi produk yaitu Sabrina. Sabrina adalah chatbot perbankan digital terintegrasi Whatsapp yang dapat digunakan melalui smartphone. Chatbot ini di desain menggunakan metode pohon dialog sehingga setiap inputan user sudah terdapat jawabannya yang telah dibuat pada sistem chatbot tersebut.

Latar Belakang

Terlepas dari potensi besar chatbot, banyak chatbot layanan pelanggan tidak memenuhi harapan pelanggan dan menyebabkan kegagalan layanan. Akibatnya, banyak user layanan tidak puas terhadap chatbot, karena layanan online yang tidak memuaskan memiliki efek negatif pada word-of-mouth, loyalitas, dan niat untuk membeli suatu produk. Selain itu, menurut penelitian manusia menulis secara berbeda ketika mereka senang atau frustrasi dan dengan demikian, teks dengan sendirinya menyampaikan banyak informasi tentang emosi manusia [4] Pengguna yang kurang senang dengan chatbot menggunakan lebih sedikit persetujuan, lebih sedikit kata-kata positif, dan lebih banyak kata-kata yang berhubungan dengan kemarahan dan dengan demikian, mengekspresikan lebih banyak sentimen negatif [5]. Oleh karena itu dibutuhkan sentimen analisis untuk membantu mengevaluasi kepuasan pelanggan menggunakan chatbot.

Topik dan Batasannya

Berdasarkan latar belakang di atas, agar perancangan ini tidak menyimpang dari pokok pembahasan, maka penulis membuat batasan masalah sebagai berikut :

1. Data yang digunakan adalah data nasabah BRI yang berbentuk tabular dengan campuran fitur kategorikal dan numerik yang sudah dianonimisasi untuk menjaga privasi pengguna.
2. Model pembuatan sentiment analisis terbatas pada model pemelajaran mesin berbasis graf dan model berbasis pohon.
3. Model berbasis graf melakukan prediksi multi label yaitu prediksi graph untuk menentukan apakah chat dari pengguna termasuk sentiment positif, negatif, atau netral.

Tujuan

Berdasarkan latar belakang masalah di atas, maka tujuan yang ingin dicapai oleh penulis dalam penelitian ini adalah dapat merancang model pembelajaran mesin berbasis graf pada chatbot perbankan yang dapat menganalisis sentimen dari percakapan dengan user dengan mempertimbangkan relasi antar kata sehingga dapat digunakan untuk menilai kepuasan pengunaan chatbot perbankan.

Organisasi Tulisan

Penjelasan mengenai organisasi tulisan seperti pada bab 2 berisi pembahasan mengenai studi literatur yang berkaitan dengan penelitian pada tugas akhir penulis. Untuk bab 3 berisi penjelasan mengenai teori dan rancangan

desain sistem yang dibangun oleh penulis. Pada bab 4 membahas hasil evaluasi terhadap sistem yang berhasil dibangun. Pada bab 5 berisi rangkuman hasil evaluasi pada penelitian berupa kesimpulan.

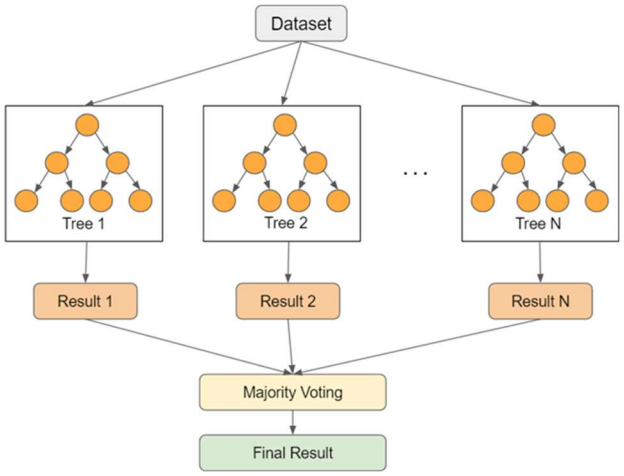
2. Studi Terkait

Penelitian ini mengacu pada beberapa penelitian sebelumnya yang berkaitan dengan penggunaan model machine learning berbasis graf dalam melakukan prediksi terhadap kasus tertentu dan membandingkannya dengan model terdahulu yaitu pembelajaran mesin berbasis graf.

2.1 Pembelajaran Mesin Berbasis Pohon

Metode pemelajaran mesin berbasis pohon merupakan salah satu cabang dari supervised learning yang menawarkan berbagai metode fleksibel yang dibangun di atas kerangka umum (Kern, Klausch, & Kreuter, 2019). Metode berbasis pohon mampu menangani data yang beragam tanpa memerlukan pra-pemrosesan yang mendalam. Metode berbasis po- hon juga memiliki komputasi yang cepat dan akurat. Pada penelitian ini akan digunakan tiga metode pembelajaran mesin berbasis pohon antara lain Random Forest, LightGBM, dan XGBoost.

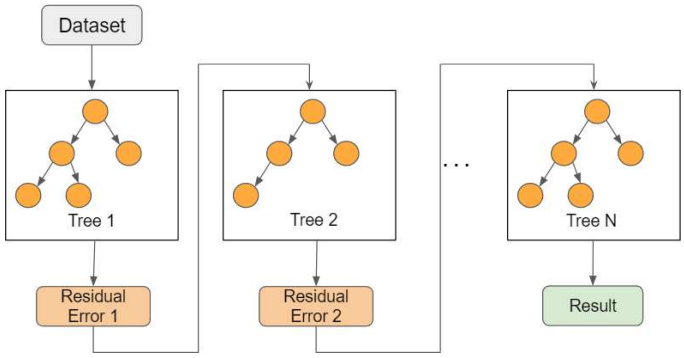
2.1.1 Random Forest



Gambar 1. Mekanisme Kerja Random Forest

Random Forest (RF) merupakan salah satu metode ensamble learning yang dibangun atas kumpulan dari Decision Tree (DT). DT bekerja dengan cara membuat keputusan yang berupa sebuah simpul yang akan mengarah ke simpul yang tidak mempunyai anak (daun) yang merupakan hasil dari prediksi [19]. RF bekerja dengan cara membuat sebanyak n DT yang akan dilatih pada data pelatihan berbeda yang telah dilakukan random sampling untuk tiap pohonnya. Tahap pada RF akan dilakukan secara paralel untuk semua DT yang telah dibuat. Hasil prediksi dari DT akan disatukan dengan cara mengambil keputusan akhir terbanyak untuk kasus klasifikasi dan rata-rata untuk kasus regresi (lihat gambar 1).

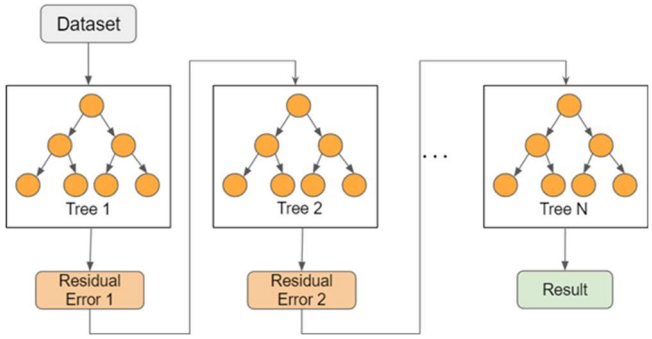
2.1.2 LightGBM



Gambar 2. Mekanisme Kerja LightGBM

LightGBM merupakan salah satu gradient boosting framework yang menggunakan algoritma berbasis pohon. LightGBM akan tumbuh secara vertikal, yaitu LightGBM memilih simpul dengan delta loss besar untuk tumbuh [20]. Algoritma lain akan tumbuh secara horizontal yaitu menambahkan simpul hingga penuh sebelum pindah ke simpul di level berikutnya. Saat menumbuhkan simpul, algoritma leaf-wise dapat men- gurangi lebih banyak kerugian daripada algoritma level-wise (lihat gambar 2). Proses yang terjadi pada LightGBM dilakukan secara berurutan.

2.1.3 XGBoost



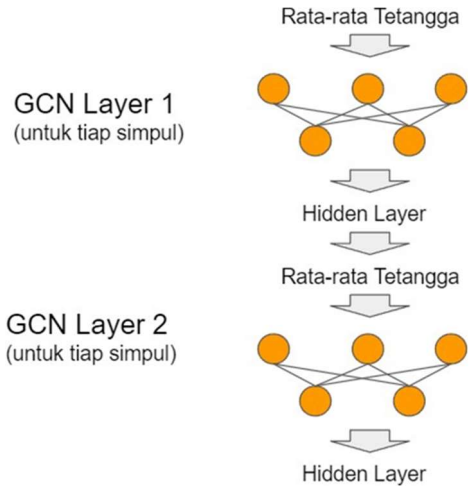
Gambar 3. Mekanisme Kerja XGBoost

XGBoost merupakan framework gradient boosting yang berbasis decision tree (DT). XGBoost akan bekerja secara level-wise. XGBoost bekerja dengan cara gradient boosting untuk mengurangi loss dari hasil prediksi [21]. XGBoost bekerja de- ngan cara membuat n DT. Pada klasifikasi yang dilakukan oleh DT, mungkin saja terjadi kesalahan klasifikasi terhadap suatu data. Untuk mengatasi hal tersebut, gradient boosting akan memberikan weight terhadap residual error yang terjadi dari kesalahan klasifikasi tersebut ke DT berikutnya sebagai input yang akan dilakukan pelatihan (lihat gambar 3). Proses yang terjadi pada gradient boosting dilakukan secara berurutan mulai dari DT 1 hingga ke DT ke N. Hasil model dari XGBoost didapatkan setelah pelatihan DT ke N. Poin utama yang membedakan XGBoost dengan gradient boosting tree lainnya adalah novel tree learning algorithm yaitu algoritma ini akan melakukan pembobotan terhadap data tiap kali dilakukan pelatihan untuk mengatasi data yang bernilai kosong. Selain itu, XGBoost juga dapat bekerja secara paralel dan terdistribusi agar pelatihan yang dilakukan lebih cepat.

2.2 Pembelajaran Mesin Berbasis Graf

Sejak keberhasilan deep learning, banyak penelitian yang telah mempelajari bagaimana struktur dari graf dapat diimplementasikan ke dalam jaringan saraf tiruan. Sistem berbasis GNN (Graph Neural Network) telah menunjukan banyak kegunaan sebab dapat diguna- kan dalam struktur data graf seperti struktur protein dan knowledge graph [18]. Pada penelitian ini akan digunakan tiga metode pembelajaran mesin berbasis graf yaitu Transformer-Graph, Chebycev-Graph, dan GCN.

2.2.1 GCN (Graph Convolutional Network)



Gambar 4. Mekanisme Kerja GCN

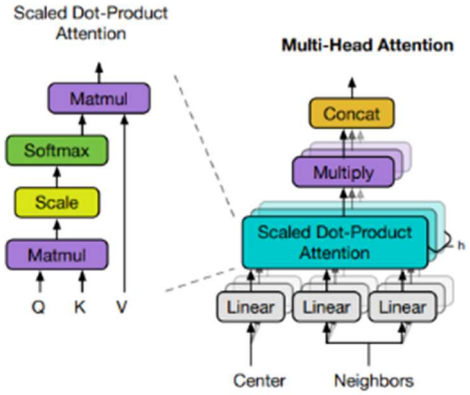
GCN merupakan jaringan saraf convolutional yang dapat mempelajari dan bekerja ter- hadap graf dengan memanfaatkan struktur dari graf. GCN menyelesaikan permasalahan seperti klasifikasi simpul maupun prediksi dari sisi pada suatu graf [22]. GCN bekerja dengan melakukan weighted terhadap rata-rata dari semua fitur sim- pul tetangga. Hal ini dilakukan terhadap semua simpul untuk dilakukan pembaharuan pada suatu layer (lihat Gambar 4). GCN akan bekerja dengan

melakukan fungsi agre- gat yang akan dilakukan ditiap layernya. Rumus dari fungsi agregat yang dimiliki GCN adalah sebagai berikut :

$$H^{(l+1)} = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

Dari persamaan diatas dapat dijabarkan bahwa A merupakan struktur dari graf yaitu matriks ketetanggaan, Dˆ merupakan matriks derajat simpul dari Aˆ, Aˆ merupakan hasil penjumlahan dari matriks A dan matriks I, I merupakan matriks identitas, W merupakan matriks weight, σ merupakan fungsi aktivasi relu. Dalam fungsi tersebut, l menandaan layer pada GCN. Saat l bernilai satu, maka H atau input yang masuk adalah fitur node dari graf. Saat l bernilai lebih dari atau sama dengan dua, maka input yang masuk adalah hasil dari layer sebelumnya.

2.2.2 Transformer-Graph

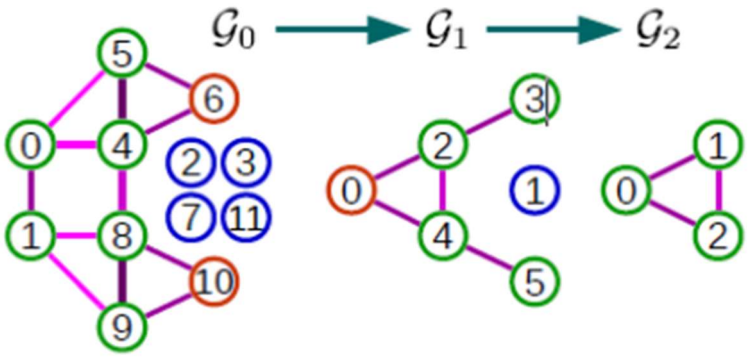


Gambar 5. Mekanisme Kerja Trasformer-Graph

Model Transformer didasarkan pada arsitektur encoder-decoder. Encoder adalah persegi panjang abu-abu di sebelah kiri dan decoder di sebelah kanan. Encoder dan decoder masing-masing terdiri dari dua dan tiga sublapisan. Attention multi-head, jaringan feedforward yang terhubung penuh, dan kesadaran diri decoder encoder dalam kasus decoder disebut multi-head attention. Encoder bertanggung jawab untuk melangkah melalui langkah-langkah waktu input dan menyandikan seluruh urutan menjadi vektor panjang tetap yang disebut vektor konteks. Decoder bertanggung jawab untuk melewati langkah-langkah waktu keluaran saat membaca dari vektor konteks.[17]

Pada gambar diatas akan diinputkan center node dan neigbor dari node center. Node-node tersebut akan dimasukan ke fungsi linear yang memiliki beberapa hidden layer. Hasil dari fungsi linear tadi akan masuk ke proses yang bernama Scaled-Dot Product Attention dimana proses ini akan mengubah ketiga masukan tadi menjadi satu vektor yang menjadi penentuan terhadap prediksi sentimennya nanti.

2.2.3 Chebynev-Graph

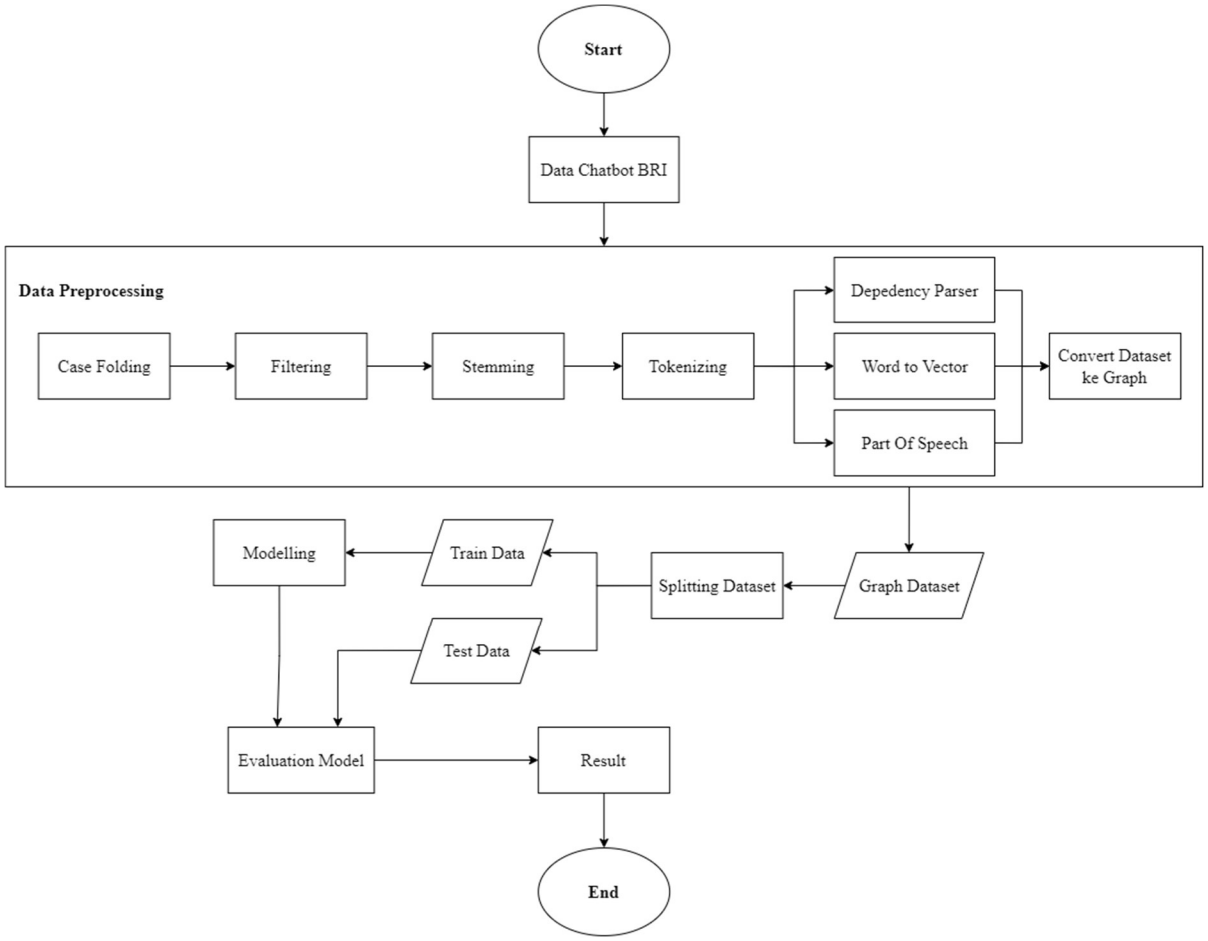


Gambar 6. Mekanisme Kerja Chebynev-Graph

Karena graf skala besar menjadi semakin umum, ini menimbulkan tantangan komputasi yang signifikan untuk memproses, mengekstrak, dan menganalisis data graf besar. Pengkasaran graf adalah salah satu teknik populer untuk memperkecil ukuran graf sambil mempertahankan sifat-sifat esensialnya. Meskipun literatur kasar grapf kaya, hanya ada eksplorasi terbatas metode berbasis data di lapangan. Dalam pekerjaan ini, kami memanfaatkan kemajuan terkini dari deep learning pada graf untuk pengasaran graph. Kami pertama-tama mengusulkan kerangka kerja untuk mengukur kualitas algoritme pengasaran dan menunjukkan bahwa tergantung pada tujuannya, kami perlu memilih operator Laplace dengan hati-hati pada graf kasar dan operator proyeksi/pengangkatan terkait. Termotivasi oleh pengamatan bahwa pilihan bobot tepi saat ini untuk graf kasar mungkin kurang optimal, kami membuat parameter peta penugasan bobot dengan jaringan saraf graf dan melatihnya untuk meningkatkan kualitas pengasaran dengan cara yang tidak diawasi. Melalui percobaan ekstensif pada jaringan sintetik dan nyata, kami menunjukkan bahwa metode kami secara signifikan meningkatkan metode pengasaran graf umum di bawah berbagai metrik, rasio reduksi, ukuran graf, dan jenis graf. [23]

Pada gambar berikut node yang memiliki kemiripan misalkan node 2,3,7,dan 11 dari segi nilai chebycevnya maka node-node tersebut akan dilakukan proses graf coarsening menjadi node baru yaitu node 1 dan jika node 1,2,dan 3 memiliki kemiripan juga maka akan dilakukan proses graf coarsening lagi dan seterusnya.

3. Sistem yang Dibangun



Gambar 7. Sistem Kerja

Berikut merupakan penjelasan detail untuk masing-masing proses penelitian ini.

3.1. Dataset

Penelitian ini menggunakan dataset dari nasabah BRI yang melakukan percakapan dengan Sabrina Chatbot BRI. Jumlah data yang didapatkan dari dataset tersebut adalah 40.000 data text nasabah. Dataset ini memiliki attribute waktu pengiriman pesan, id pesan, dan lain lain. Dengan label sentiment positif, netral, dan negative. Pada penelitian ini, pemilihan atribut merupakan tahap awal untuk membuat model analisis sentiment. Atribut dalam machine learning disebut sebagai feature. Feature memiliki karakteristik dapat diukur dan relevan dengan tujuan penelitian. Untuk penelitian ini diambil 2 attribut yaitu text dan label dari dataset.

3.2. Data Preprocessing

1. Case Folding

   Setiap kata dari teks ulasan yang sudah diseleksi sebelumnya disamakan bentuk case nya menjadi huruf kecil semua untuk meminimalisir terjadi nya error ketika memproses data.

2. Filtering

   Setelah teks ulasan dirubah menjadi huruf kecil semua, sekarang dihilangkan tanda baca seperti !.,? dan lain-lain. Stopwords Removal ini akan menhhilangkan setiap kata yang tidak relevan topik sentimen dengan cara proses stringmatching dengan stopwords yang tersimpan pada file .txt lain.

3. Stemming

   Teks ulasan yang sudah tidak mengandung kata stopwords sekarang diproses untuk mengubah setiap kata nya menjadi kata baku agar mudah diproses.

4. Tokenization

Kata-kata yang ada pada teks ulasan sekarang dipecah menjadi bentuk token untuk memudahkan proses pemberian bobot dan pembuatan word vector

5.Random Under Sampling

Pada Dataset perbandingan setiap labelnya tidak seimbang yaitu sekitar 70% untuk data netral dan 15% untuk masing positive dan negative. Karena dataset yang tidak seimbang dibutuhkan penyeimbangan kelas dengan menggunakan sampling dari datasetnya. Random undersampling adalah metode non-heuristik yang bertujuan untuk menyeimbangkan distribusi kelas melalui penghapusan acak contoh kelas mayoritas. Alasan di baliknya adalah untuk mencoba menyeimbangkan dataset dalam upaya untuk mengatasi kekhasan dari algoritma machine learning.

3.3. Representasi text berbasis graph

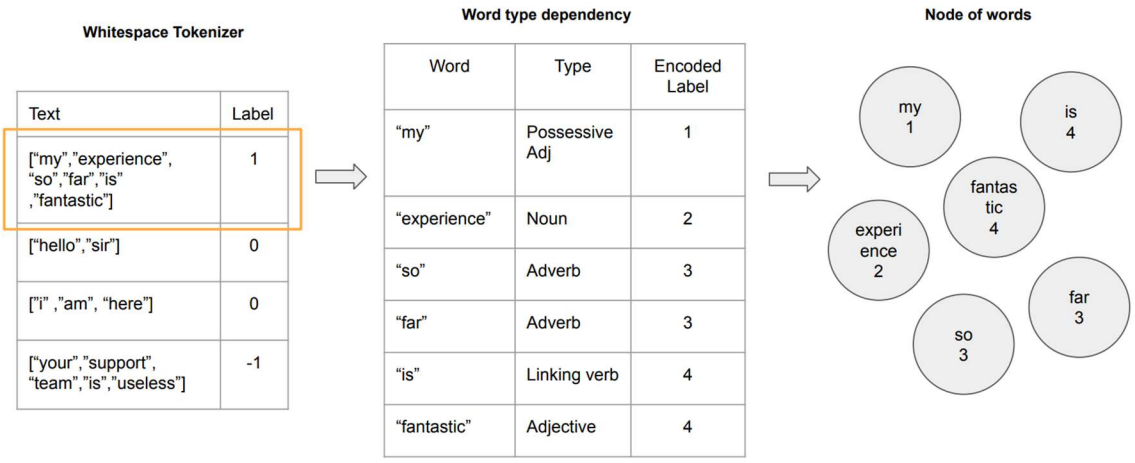
Setelah melakukan preprocessing terhadap dataset yang ada selanjutnya dilakukan perubahan data tersebut ke graph. Pada kalimat akan dilakukan tokenizer untuk memisahkan kalimat menjadi kata antar kata. Dari kata-kata tersebut akan dilakukan metode depedency parsing yang berfungsi untuk mengetahui relasi antar kata pada kalimat tersebut sehingga dapat dihubungkan menjadi edge antar node. Selanjutnya dilakukan word embedding untuk mengubah text di node menjadi vector karena model tidak dapat di train dengan data yg berformat text.

1. TF-IDF Vectorizer

TF-IDF merupakan singkatan dari Term Frequency - Inverse Document Frequency. TF-IDF merupakan gabungan dari 2 proses yaitu Term Frequency (TF) dan Inverse Document Frequency (IDF). TF-IDF biasa digunakan ketika kita ingin mengubah data teks menjadi vektor namun dengan memperhatikan apakah sebuah kata tersebut cukup informatif atau tidak. Mudahnya, TF-IDF membuat kata yang sering muncul memiliki nilai yang cenderung kecil, sedangkan untuk kata yang semakin jarang muncul akan memiliki nilai yang cenderung besar. Ini berfungsi untuk mengurangi vektor 0 dari vectorizer dibandingkan menggunakan Bag of Word yang akan memunculkan banyak 0 pada representasi katanya karena disesuaikan dengan jumlah kata unik di dataset.

2.Word Type Encode

Menggunakan Stanza Stanford untuk mengubah tiap kata pada kalimat menjadi sesuai dengan POS (part of speech) pada grammar bahasa inggris. Kemudian tipe kata tersebut di encode menjadi numerical.



Gambar 8. Part of Speech

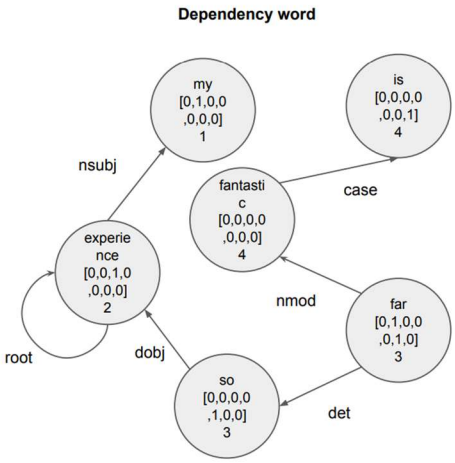
3. Dependency Parser

Pada tahap ini menggunakan Stanza Stanford untuk mengetahui dependency parsing antara dua kata. Relasi antara dua kata ini akan menjadi edge pada graph yang akan dibuat.

Clausal Argument Relations	Description
NSUBJ	Nominal Subject
DOBJ	Direct Object
IOBJ	Indirect Object
CCOMP	Clausal Complement
XCOMP	Open Clausal Complement

Nominal Argument Relations	Description
NMOD	Nominal Modifier
AMOD	Adjectival Modifier
NUMMOD	Numeric Modifier
APPOS	Appositional Modifier
DET	Determiner
CASE	Prepositions, Postpositions, and other Case Markets

Other Notable Relations	Description
CONJ	Conjuct
CC	Coordinating Conjunctional



Gambar 9. Dependency Parser

3.4. Data Split

Pada tahap akan dilakukan pemisahan terhadap data pelatihan dan data pengujian. Pembagian data yang ideal adalah 70% berbanding 30% atau 80% berbanding 20% terhadap data pelatihan dan data pengujian. Pada penelitian ini akan digunakan perbandingan 80:20 yaitu 80% untuk data pelatihan dan 20% untuk data pengujian untuk model berbasis pohon maupun model berbasis graf. [15].

3.5. Pemodelan dengan Graph Convolutional Network

Pada proses training model, graph yang telah dibuat dan di split dimasukkan ke model yang berbasis graph seperti Graph Convolutional Network(GCN) dengan menggunakan parameter default lalu dilakukan tuning terhadap parameter tersebut. Model ini berfungsi untuk klasifikasi teks pada sentimen analisisnya. Model GCN yang akan dipakai pada penelitian ini adalah GCN, Transformer-Graph, dan Chebynev-GCN.

1. Message Passing, proses ini merupakan proses dimana pengiriman nilai feature berupa node properties dan edge properties dilakukan sehingga terdapat pembobotan di edge tergantung dari node-node yang berdekatan. Terdapat 3 tahapan dalam proses message passing yaitu: Message, Aggregate, dan Combine.
    - Message, pada fungsi ini memilih node source dari semua edge, dan menghitung value message dengan membagi status node source dengan derajat sourcenya.
    - Aggregate, akan mengambil message dari node target.
    - Combine, mengembalikan node dan edge yang telah di update berdasarkan node-node tetangganya.
2. Readout Layer, mengambil representasi semua node untuk mendapatkan representasi dari keseluruhan graph.
3. Graph embedding, merepresentasikan graf menjadi vektor dengan panjang yang tetap untuk setiap entitas/node dalam graf. Embedding ini merepresentasikan graf dengan dimensi yang lebih rendah dari graf dan mempertahankan topologi grafnya. Setelah pengubahan semua graf menjadi vector selesai dilakukan,

langkah selanjutnya adalah pemodelan dengan menggunakan pendekatan GCN, Chebycev-Graph, dan Transformer-Graph. Pemodelan ini bertujuan untuk melakukan klasifikasi sentimen pada dataset.

3.6. Evaluasi Model

Setelah melakukan klasifikasi sentimen, langkah selanjutnya yaitu evaluasi dan hasil analisis. Pada pembuatan sistem ini menggunakan 2 metode evaluasi yaitu:

1. Akurasi

Accuracy adalah persentase data hasil klasifikasi yang benar dibandingkan dengan seluruh data aktual yang ada. Persamaan (1) merupakan persamaan untuk menghitung Accuracy. [16]

$$Accuracy = \frac{TP + TN}{TP+FP+TN+FN} \times 100\% \quad (1)$$

Penjelasan:

- True Positive (TP) : Nilai yang didapat ketika hasil klasifikasi dan validasi yang didapat berlabel positif dan data asli nya juga berlabel positif.
- True Negative (TN) : Nilai yang didapat ketika hasil klasifikasi dan validasi yang didapat berlabel negatif dan data asli nya juga berlabel Positif.
- False Positive (FP) : Nilai yang didapat ketika hasil klasifikasi dan validasi yang didapat berlabel positif sedangkan data asli nya berlabel Negatif.
- False Negative (FN) : Nilai yang didapat ketika hasil klasifikasi dan validasi yang didapat berlabel negatif sedangkan data asli nya berlabel positif.

2.F1-Score

F1-Score merupakan perhitungan evaluasi yang menggabungkan nilai recall dan Precision. Rumus F1-Score dituliskan pada persamaan (2) dibawah. [17]

$$F1 - score = 2 \frac{precision \times recall}{precision + recall} \quad (2)$$

Penjelasan:

- Recall adalah persentase data yang berlabel positif dan data aktual juga berlabel positif, dibandingkan dengan seluruh data. Persamaan (3) merupakan persamaan untuk menghitung Recall:

$$recall = \frac{TP}{TP+} \times 100\% \quad (3)$$

- Precision adalah perhitungan yang digunakan untuk mengetahui seberapa banyak persentase data hasil klasifikasi yang memiliki nilai TP (True Positive). Persamaan (4) persamaan untuk menghitung Precision:

$$precision = \frac{TP}{TP+F} \times 100\% \quad (4)$$

4. Evaluasi

Bagian ini berisi dua sub-bagian, yaitu Hasil Pengujian dan Analisis Hasil Pengujian. Pengujian dan analisis yang dilakukan selaras dengan tujuan TA sebagaimana dinyatakan dalam Pendahuluan.

4.1 Rancangan Eksperimen

Mesin yang akan digunakan untuk eksperimen merupakan instance yang disediakan oleh BRI melalui Google Cloud Project. Instance ini disediakan oleh BRI sebagai sarana un- tuk melakukan eksperimen. Spesifikasi yang disediakan adalah 4vCPUs dengan memori sebesar 32 gigabytes (GB). Satu vCPU akan setara dengan sebuah hyperthread core.

Tabel 1. Pengaturan eksperimen model berbasis pohon

No	Model	Parameter yang digunakan		Dataset	
		Learning Rate	*Min data in leaf*	*Train*	*Test*
1	*Random Forest*	-	20	80%	20%
2	LightGBM	0.1	20	80%	20%
3	XGBoost	0.1	-	80%	20%

Tabel 2. Pengaturan eksperimen model berbasis graf

No	Model	Parameter yang digunakan			Dataset	
		Optimizer	*Learning Rate*	*Dropout*	*Train*	*Test*
1	GCN	Adam	0.01	0.3	80%	20%
2	Chebycev-Graph	Adam	0.01	0.3	80%	20%
3	Transformer-Graph	Adam	0.01	0.3	80%	20%

Pengaturan parameter akan membahas parameter yang akan digunakan untuk model berbasis graf maupun model berbasis pohon. Model berbasis pohon akan ditentukan parameter dengan performa terbaik atau hyperparameter tuning dengan menggunakan GridSearchCV yang diperoleh dari library sklearn yang dapat dilihat pada tabel 3.2. Tidak akan dilakukan hyperparameter tuning menggunakan GridSearchCV untuk model berbasis graf disebabkan tidak dapat menerima model berbentuk graf. Tabel 3.2 akan menjelaskan nilai yang dipakai dan juga nilai yang diuji pada model berbasis pohon. Agar eksperimen yang dilakukan sama terhadap semua model, harus ditentukan parameter yang bernilai tetap untuk keseluruhan model. Pada model berbasis pohon, kita dapat menggunakan min_data_in_leaf dan learning_rate untuk dijadikan parameter yang bernilai tetap untuk ketiga model berbasis pohon.

4.2 Skenario Eksperimen

Dalam eksperimen ini, akan dibuat dan diimplementasikan tiga model berbasis graf dan tiga model berbasis pohon. Terdapat beberapa parameter yang divariasikan untuk melihat performa terbaik dari model. Variasi dari yang dipilih merupakan nilai yang diambil dise- babkan sumber daya dari mesin yang terbatas. Percobaan max_depth, n_estimators, Epoch dan Layer akan dilakukan sebanyak 10 kali. Percobaan ini dilakukan untuk melihat kestabilan dari metode yang digunakan.

Tabel 3. Eksperimen variasi max depth pada model berbasis pohon

Model	Variasi Max Depth
Random Forest	5,10,15,20
LightGBM	5,10,15,20
XGBoost	5,10,15,20

Tabel 4. Eksperimen variasi n estimators pada model berbasis pohon

Model	Variasi N Estimators
Random Forest	25, 50, 75, 100
LightGBM	25, 50, 75, 100
XGBoost	25, 50, 75, 100

Tabel 5. Eksperimen waktu inferensi pada model berbasis pohon

Model	N Estimator	Variasi Max Depth
Random Forest	50	5,10,15,20
LightGBM	50	5,10,15,20
XGBoost	50	5,10,15,20

Terdapat tiga percobaan yang akan dilakukan dalam model berbasis pohon. Percobaan ini akan melakukan variasi terhadap suatu parameter. Parameter yang digunakan pada eksperimen berbasis pohon adalah max_depth (lihat tabel 3) dan n_estimators (lihat tabel 4). Untuk eksperimen variasi max_depth, nilai dari n_estimators yang digunakan adalah 100. Untuk eksperimen variasi n_estimators, nilai max_depth yang digunakan adalah 10. Percobaan yang terakhir merupakan perhitungan terhadap waktu inferensi dari model berbasis pohon (lihat tabel 5).

Tabel 6. Eksperimen variasi *epoch* pada model berbasis graf

Model	Variasi Epoch
GCN	25, 50, 75, 100
Chebycev-Graph	25, 50, 75, 100
Transformer-Graph	25, 50, 75, 100

Tabel 7. Eksperimen variasi *layer* pada model berbasis graf

Model	Variasi Hidden Layer
GCN	20, 40, 60, 80
Chebycev-Graph	20, 40, 60, 80
Transformer-Graph	20, 40, 60, 80

Tabel 8: Eksperimen waktu inferensi pada model berbasis graf

Model	Hidden Layer	Variasi Epoch
GCN	40	25, 50, 75, 100
Chebycev-Graph	40	25, 50, 75, 100
Transformer-Graph	40	25, 50, 75, 100

Terdapat tiga percobaan yang akan dilakukan dalam model berbasis graf. Parameter yang digunakan pada eksperimen berbasis graf adalah epoch (lihat tabel 6) dan layer (lihat tabel 7). Epoch dan Layer dipilih sebagai parameter yang divariasikan untuk melihat apakah penambahan pada kedua parameter tersebut akan memengaruhi performa dari model atau tidak. Untuk eksperimen variasi epoch, nilai dari layer yang digunakan adalah 2. Untuk eksperimen variasi layer, nilai epoch yang digunakan adalah 40. Percobaan yang terakhir merupakan perhitungan terhadap waktu inferensi dari model (lihat tabel 8).

4.3 Hasil Pengujian

Pada Bagian ini, akan dijelaskan hasil eksperimen yang telah dilakukan. Terdapat enam eksperimen yang telah dijalankan. Evaluasi performa terhadap masing-masing eksperimen dilakukan dengan menggunakan nilai F1-Score. Evaluasi waktu inferensi akan menggunakan satuan milisekon. Evaluasi akan dilakukan terhadap data pengujian.

3.1 Eksperimen max_depth pada Model Berbasis Pohon

Pada eksperimen ini didapatkan hasil bahwa XGBoost memiliki performa tertinggi dengan nilai F1-Score mencapai 0.7232 saat max_depth bernilai 10 yang dapat dilihat pada tabel 9. Terlihat pada tabel bahwa meningkatkan max_depth pada model Random Forest tidak akan memengaruhi performa model. Dari tabel juga didapatkan bahwa XGBoost dan LightGBM merupakan model yang memiliki variasi performa yang lebih rendah dibandingkan Random Forest Ini menunjukan bahwa XGBoost dan LightGBM lebih stabil dibandingkan Random Forest pada eksperimen max_depth. Dari eksperimen tersebut didapatkan bahwa meningkatkan max_depth di LightGBM akan meningkatkan hasil performa dari model.

Tabel 9. Eksperimen max_depth pada Model Berbasis Pohon

Max Depth	Random Forest		LightGBM		XGBoost	
	Akurasi	F1-Score	Akurasi	F1-Score	Akurasi	F1-Score
5	0.6381	0.4971	**0.6886**	**0.5741**	0.6063	0.5143
10	0.6381	0.4971	**0.6707**	**0.5894**	0.6356	0.5213
15	0.6381	0.4971	0.6669	**0.5730**	**0.6822**	0.5505
20	0.6381	0.4971	0.6628	0.6012	**0.7232**	**0.6132**

3.2 Eksperimen n_estimator pada Model Berbasis Pohon

Secara keseluruhan didapatkan bahwa LightGBM mendapatkan performa tertinggi dengan nilai AUC mencapai 0.722 disaat n_estiamator bernilai 75 pada Tabel 10. Pada Tabel 10 juga terlihat bahwa XGBoost mendapatkan performa yang paling baik pada hampir semua variasi n_estimator yaitu 25, 50 dan 200. Hasil eksperimen menunjukan bahwa Random Forest mendapatkan performa yang paling buruk kecuali pada variasi n_estimator 25 yaitu nilai F1-Score maksimal yang didapat adalah 0,631. Pada

eksperimen ini terlihat bahwa LightGBM dan XGBoost lebih stabil dibandingkan dengan Random Forest. Dari eksperimen ini juga didapatkan bahwa meningkatkan nilai n_estimator pada LightGBM dan XGboost akan menaikan performa model.

Tabel 10. Eksperimen n-estimator pada Model Berbasis Pohon

Estimator	Random Forest		LightGBM		XGBoost	
	Akurasi	F1-Score	Akurasi	F1-Score	Akurasi	F1-Score
25	0.6381	0.4971	0.6657	**0.5935**	**0.6063**	0.5133
50	0.6381	0.4971	0.6711	**0.6066**	**0.6356**	0.5541
75	0.6381	0.4971	0.6791	**0.6101**	**0.6822**	0.6092
100	0.6381	0.4971	0.7122	0.6125	**0.7271**	**0.6245**

3.3 Eksperimen epoch pada Model Berbasis Graf

Pada eksperimen ini terlihat pada Tabel 11 bahwa GCN memiliki performa yang paling baik dibandingkan Chebynec-GCN dan Tansformer-GCN pada seluruh epoch. Terlihat juga bahwa GCN membutuhkan epoch yang lebih tinggi untuk mendapatkan model dengan performa yang lebih baik. Semakin banyak epoch, maka GCN dapat mempelajari data lebih baik sehingga performanya lebih baik. Hasil eksperimen didapatkan bahwa Transformer-GCN memiliki performa paling buruk untuk keseluruhan epoch. Dari ketiga model didapatkan juga bahwa GraphSAGE memiliki performa yang lebih stabil dibandingkan Chebynec-GCN dan Tansformer-GCN. Secara keseluruhan, pada eksperimen ini dapatkan bahwa semakin banyak epoch yang digunakan maka semakin baik performa dari model yang dibuat.

Tabel 11. Eksperimen epoch pada Model Berbasis Graf

Epoch	Chebycev-GCN		Transformer Graph		GCN	
	Akurasi	F1-Score	Akurasi	F1-Score	Akurasi	F1-Score
25	0.6183	0.5517	0.6218	**0.5750**	**0.6219**	0.3042
50	0.6190	0.5828	0.6495	0.5619	**0.6895**	**0.6151**
75	0.6276	**0.6183**	0.6397	0.5604	**0.7173**	0.6163
100	0.6651	**0.6090**	0.6389	0.5712	**0.6789**	0.6021

3.4 Eksperimen hidden_layer pada Model Berbasis Graf

Pada eksperimen ini terlihat bahwa sebagian model akan mendapat nilai F1-Score yang semakin menurun. Terlihat pada Tabel 12 bahwa model Transformer-Graph mendapatkan performa yang paling buruk dibandingkan dua metode lainnya pada saat hidden_layer bernilai 80. Pada hasil eksperimen juga terlihat bahwa performa dari Chebynev-GCN selalu menurun disetiap layernya. Nilai F1-Score tertinggi terjadi pada model GraphSAGE saat menggunakan 80 hidden_layer dengan nilai 0.7173. Oleh sebab itu, GCN adalah metode graf yang paling efisien dibandingkan dua metode lainnya. Secara berurutan didapatkan bahwa performa paling baik pada model berbasis graf pada eksperimen hidden_layer adalah GCN, Chebynev-GCN dan Transformer-GCN. Pada eksperimen ini didapatkan bahwa GCN merupakan model yang paling stabil dibandingkan Chebynev-GCN dan Transformer-GCN. Pada eksperimen ini juga didapatkan bahwa meningkatkan hidden_layer pada model berbasis graf akan menurunkan sebagian performa prediksi model.

Tabel 12. Eksperimen hidden_layer pada Model Berbasis Graf

Hidden Layer	Chebycev-GCN		Transformer Graph		GCN	
	Akurasi	F1-Score	Akurasi	F1-Score	Akurasi	F1-Score
20	0.6120	0.5517	0.6179	**0.5957**	**0.6188**	0.5925
40	0.6632	0.5800	0.6727	*0.5808*	**0.7173**	***0.6016***
60	0.6775	0.5857	0.6521	*0.5892*	**0.7103**	***0.6005***
80	0.6651	0.5779	0.6028	*0.5470*	**0.6699**	***0.5816***

3.4 Eksperimen Waktu Inferensi

Pada percobaan ini didapatkan bahwa GCN memiliki waktu inferensi paling cepat (lihat Tabel 13). Transformer-GCN menggunakan fungsi Transformer yang menjalankan input secara berurutan sehingga membutuhkan waktu yang lebih lama dibandingkan GCN dan Chbynev-GCN yang dapat dilihat dari generator yang digunakan yang berupa graph. Dari eksperimen ini didapatkan bahwa meningkatkan epoch pada seluruh model berbasis graf tidak secara signifikan memengaruhi waktu inferensi. Grafik hubungan epoch terhadap waktu inferensi dapat dilihat pada gambar. Pada percobaan model berbasis pohon, didapatkan bahwa waktu inferensi lebih lama dibandingkan dengan model berbasis graf (dapat dilihat tabel 13). LightGBM mendapatkan waktu inferensi terendah untuk hampir semua n-estimator kemudian diikuti oleh Random Forest dan XGBoost. Dari eksperimen ini didapatkan bahwa meningkatkan max_depth pada model berbasis pohon dapat membuat waktu inferensi menjadi lebih lama. Grafik peningkatan max_depth terhadap waktu inferensi dapat dilihat pada gambar

Tabel 13. Eksperimen Waktu Inferensi (milisecond)

Hidden Layer	Chebycev-GCN	Transformer Graph	GCN
20	48.82	***46.23***	51.46
40	58.96	***52.87***	62.3
60	83.39	***76.48***	95.59
80	84.85	***79.21***	97.86

4.1.1 Eksperimen dengan menggunakan Under-Sampling

Pada eksperimen ini didapatkan bahwa GCN dan XGBoost memiliki hasil evaluasi terbaik dari masing-masing mode graf dan tradisional. Perbedaan diantara keduanya adalah sekitar 0.015% dimana XGBoost lebih tinggi dari GCN. Hasil dari eksperimen under-sampling dapat dilihat terjadi penurunan evaluasi model dikarenakan adanya pengurangan dataset agar terbentuknya label yang seimbang.

Tabel 15. Hasil Eksperimen Under-Sampling

	Transformer-Graph	Chebycev-GCN	GCN	Random Forest	LightGBM	XGBoost
Akurasi	0.6522	0.6452	**0.6665**	0.5912	0.6632	**0.6798**
F1-Score	0.5394	0.5341	**0.5814**	0.4598	0.5921	**0.5892**

4.2 Analisis Hasil Pengujian

Dari tabel 14 (hasil pengujian), dapat kita lihat bahwa perbedaan dari GCN dan XGBoost sekitar 0.01% dari segi evaluasi modelnya yaitu Akurasi dan F1-Score. Dan dari sisi inferesi waktu terdapat perbedaan sebesar 100ms yang membuat GCN lebih cepat dibandingkan dengan XGBoost. Kecepatan ini dipengaruhi oleh model GCN yang dapat mengambil node yang tertentu sehingga membuat modelling lebih cepat dilakukan. dibandingkan XGBoost yang merupakan model *tree-based* yang akan memakan banyak *resources* untuk training datanya.

Tabel 14. Hasil Pengujian

	Transformer-Graph	Chebycev-GCN	GCN	Random Forest	LightGBM	XGBoost
Akurasi	0.6727	0.6775	**0.7173**	0.6381	0.7122	**0.7232**
F1-Score	0.5808	0.5857	**0.6016**	0.4971	0.6125	**0.6163**
Inference Time	51.46 ms	48.82 ms	**46.23 ms**	59.14ms	71.18 ms	**155.6 ms**

5. Kesimpulan

Pada penelitian ini, telah dilakukan berbagai percobaan untuk membangun model rekomendasi perbankan berbasis graf dan model rekomendasi perbankan berbasis pohon. Model berbasis graf yang digunakan adalah GCN, Chebycev-GCN dan Transformer-GCN. Model berbasis pohon yang digunakan adalah Random Forest, LightGBM dan XGBoost.

Pada model berbasis graf, GCN merupakan model yang memberikan performa yang paling baik pada semua percobaan kemudian diikuti oleh Chebycev-GCN dan Transformer-GCN secara berurutan. Adapun Chebycev-GCN dan Transformer-GCN memberikan performa yang kurang baik pada semua percobaan. GraphSAGE memiliki kekurangan yaitu F1-Score yang sedikit kurang dibandingkan XGBoost. GCN memberikan waktu inferensi yang paling cepat dibandingkan XGBoost disebabkan GCN merupakan model yang melakukan pemotongan node jika label ditemukan. Hasil eksperimen pada penelitian menunjukan bahwa penggunaan metode pemelajaran mesin berbasis graf memiliki performa yang lebih cepat dengan F1-Score yang berbeda sedikit dibandingkan pemelajaran mesin berbasis pohon.

Daftar Pustaka

[1] Abidah Elcholiqi, A. M. (2020). Chatbot in Bahasa Indonesia Using NLP to Provide Banking Information. IJCCS (Indonesian Journal of Computing and Cybernetics Systems). Vol.14, No.1, January 2020, pp. 91~102
[2] C.S. Kulkarni, A.U. Bhavsar, S.R. Pingale, and S.S. Kumbhar, "BANK CHAT BOT – An Intelligent Assistant System Using NLP and Machine Learning," IRJET (International Research Journal of Engineering and Technology, vol: 04 Issue: 05 | May -2017 [Online]. Available:https://www.irjet.net/archives/V4/i5/IRJET-V4I5611.pdf [Accessed: 9-Jan-2018]
[3] S.H. Aparaj, C. Shashank, and R. Bhagat, "Smart Bot - A Virtual Help Desk Chat Bot," ICAICTE2013 (International Conference on Advanced Information Communication Technology in Engineering, 2013.
[4] Brave, S., Nass, C.: Emotion in human-computer interaction. In: Jacko, J.A., Sears, A. (eds.) The human-computer interaction handbook, pp. 81–96. L. Erlbaum Associates Inc, NJ, USA (2002)
[5] Skowron, M., Rank, S., Theunis, M., Sienkiewicz, J.: The Good, the Bad and the Neutral: Affective Profile in Dialog System-User Communication. In: D'Mello, S., Graesser, A., Schuller, B., Martin, J.-C. (eds.) Affective Computing and Intelligent Interaction, pp. 337– 346. Springer, Berlin, Heidelberg (2011)
[6] Feldman, R & Sanger, J. The Text Mining Handbook : Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press, New York, 2007.
[7] Aviananda Dwirahma (2020). Berkenalan dengan Sabrina, "Robot" Customer Service dari BRI. BRITECH. 24 Mei 2022. https://digital.bri.co.id/article/berkenalan-dengan-sabrina-robot-customer-service-8dfm
[8] Lee, Vivian Lay Shan, et al. "Semi-supervised learning for sentiment classification using small number of labeled data." Procedia Computer Science 161 (2019): 577-584.
[9] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).
[10] Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." Advances in neural information processing systems 29 (2016).
[11] Chen, Jie, Tengfei Ma, and Cao Xiao. "Fastgcn: fast learning with graph convolutional networks via importance sampling." arXiv preprint arXiv:1801.10247 (2018).
[12] Yao, Liang, Chengsheng Mao, and Yuan Luo. "Graph convolutional networks for text classification." Proceedings of the AAAI conference on artificial intelligence. Vol. 33. No. 01. 2019.
[13] Bijari, Kayvan, et al. "Leveraging deep graph-based text representation for sentiment polarity applications." Expert Systems with Applications 144 (2020): 113090
[14] Ferdiana, Ridi, et al. "Dataset Indonesia untuk analisis sentimen." Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI) 8.4 (2019): 334-339.
[15] Gholamy, Afshin, Vladik Kreinovich, and Olga Kosheleva. "Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation." (2018).
[16] Reich, Y., & Barai, S. V. (1999). Evaluating machine learning models for engineering problems. Artificial Intelligence in Engineering, 13(3), 257-272.
[17] Doshi-Velez, Finale, and Roy H. Perlis. "Evaluating machine learning articles." Jama 322.18 (2019): 1777-1779.
[18] Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., & Sun, Y. (2020). Masked label prediction: Unified message passing model for semi-supervised classification. arXiv preprint arXiv:2009.03509.
[19] Rigatti, S. J. (2017). Random forest. Journal of Insurance Medicine, 47(1), 31-39.
[20] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30.
[21] Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., ... & Zhou, T. (2015). Xgboost: extreme gradient boosting. R package version 0.4-2, 1(4), 1-4.
[22] Chen, M., Wei, Z., Huang, Z., Ding, B., & Li, Y. (2020, November). Simple and deep graph convolutional networks. In International conference on machine learning (pp. 1725-1735). PMLR.
[23] Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29.