PENERAPAN ALGORITMA *BI-DIRECTIONAL LONG SHORT-TERM MEMORY* DALAM SISTEM REKOMENDASI LOKASI BERDASARKAN DATA PENGGUNA

TUGAS AKHIR
Diajukan untuk Memenuhi Salah Satu Syarat Kelulusan
Program Pendidikan Sarjana

Oleh:
Maichel Yunarto Budiman
2016130038

JURUSAN INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER – LIKMI
BANDUNG
2022

PENERAPAN ALGORITMA *BI-DIRECTIONAL LONG SHORT-TERM MEMORY* DALAM SISTEM REKOMENDASI LOKASI BERDASARKAN DATA PENGGUNA

Oleh:
Maichel Yunarto Budiman
2016130038

Bandung, 10 Januari 2022
Menyetujui

Dr. Hery Heryanto, S. Kom., M. Kom.	Dhanny Setiawan, S.T., M.T.
Pembimbing	Ketua Jurusan

JURUSAN INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER – LIKMI
BANDUNG
2022

ABSTRAK

Penanganan sistem rekomendasi pengguna sudah banyak dilakukan dengan menangkap relasi antara pengguna dan *item* yang dibeli atau diinginkan. Tetapi, dalam beberapa penelitian, dalam pengambilan keputusan hanya menggunakan kesamaan antara setiap nilai *item* dengan pengguna. Hal ini kurang bisa diterapkan mengingat pengguna memiliki pola tersendiri, sehingga diusulkan metode *Bidirectional LSTM*.

Pembuatan sistem rekomendasi menggunakan metode *Bidirectional LSTM* memerlukan beberapa proses. Proses yang diperlukan bertujuan untuk mendapatkan hasil yang maksimal dari model yang akan dibuat. Proses yang digunakan meliputi akuisisi data, *cleaning* data, *augmentasi* data, dan *splitting* data. Sementara untuk tampilan, akan digunakan *google maps* api untuk mendapat lokasi pengguna.

Model dibuat menggunakan *google colabs.* Pengujian ini dilakukan dengan cara mengubah jumlah *input layer* dan nilai *learning rate.* Dari hasil *training* maka akan didapatkan nilai RMSE. Dari nilai akurasi dan RMSE, maka dapat disimpulkan model mana yang memiliki akurasi tertinggi. Dari pengujian yang dilakukan, maka didapatkan beberapa hasil dan diambil satu hasil tertinggi. Hasil tertinggi yang didapatkan adalah model yang memiliki akurasi sebesar 93% dan RMSE sebesar 0.263. Model terbaik yang didapatkan dari hasil pengujian akan digunakan sebagai model di tampilan.

Dari hasil yang didapatkan maka dapat diambil sebuah kesimpulan yaitu penggunaan *Bidirectional LSTM* untuk merekomendasikan lokasi pengguna dapat menghasilkan hasil akurasi yang tinggi, sehingga dapat disimpulkan bahwa *Bidirectional LSTM* dapat digunakan untuk rekomendasi lokasi. *Bidirectional LSTM* dapat digunakan juga dengan beberapa dataset yang berbeda. Untuk pengujian selanjutnya, dapat diuji coba kan dengan mengubah nilai *dropout* atau menambahkan jumlah *layer*.

Kata kunci: Rekomendasi Lokasi, pola perjalanan, *Bidirectional LSTM*

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yesus Kristus atas segala berkat, kasih karunia, penyertaannya dan kebaikannya sehingga penulis dapat menyelesaikan penyusunan tugas akhir yang berjudul **PENERAPAN ALGORITMA BI-DIRECTIONAL LONG SHORT-TERM MEMORY DALAM SISTEM REKOMENDASI LOKASI BERDASARKAN DATA PENGGUNA** dengan tepat waktu yang dapat diajukan untuk memenuhi salah satu syarat kelulusan pada program sarjana (S1) dengan jurusan Informatika dalam bidang keahlian Rekayasa Perangkat Lunak di Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI Bandung.

Dalam menyelesaikan Tugas Akhir ini, penulis sadar laporan ini masih jauh dari kesempurnaan, masih banyak terdapat kekurangan, kesalahan maupun ketidaklayakan yang terdapat di dalam laporan ini, kiranya terjadi karena kelemahan, keterbatasan ilmu pengetahuan dan pengalaman penulis. Dengan demikian, penulis meminta kebijaksanaan dan pengertian dari para pembaca untuk mamukluminya. Pada kesempatan ini, penulis ingin mengucapkan terima kasih sebesar besarnya kepada pihak-pihak yang telah membantu dalam mengerjakan tugas akhir ini:

1. Bapak Dr. Hery Heryanto, S. Kom., M. Kom. selaku dosen pembimbing dalam memberikan arahan, memberikan masukan, bimbingan, dan meluangkan waktu kepada penulis yang berguna bagi penulis untuk menyelesaikan tugas akhir ini.
2. Bapak Dhanny Setiawan, S.T., M.T selaku Ketua Jurusan Teknik Informatika STMIK LIKMI.
3. Seluruh dosen STMIK LIKMI yang selama ini telah memberikan pelajaran dan ilmu kepada penulis selama masa perkuliahan.
4. Keluarga dan saudara-saudara penulis yang telah banyak membantu dan memberikan dukungan dalam doa dan lainnya selama menyelesaikan tugas akhir ini.
5. Seluruh teman Angkatan 2016 yang telah memberi semangat kepada penulis.
6. Pihak-pihak lain yang tidak dapat disebutkan satu per satu yang selalu mendukung dan memberi semangat kepada penulis.

Akhir kata, penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah berperan penting dalam membantu penulis dalam menyelesaikan tugas akhir ini.

Bandung, 10 Januari 2022

Penulis

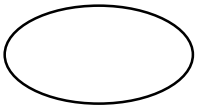
DAFTAR ISI

DAFTAR GAMBAR

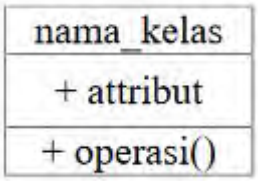
DAFTAR TABEL

DAFTAR SIMBOL

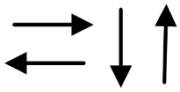
1. *Use Case Diagram*

Nama Simbol	Simbol	Keterangan
Use Case		Sebuah kebiasaan atau fungsi utama (key behaviour) yang dilakukan oleh sistem perangkat lunak
Asosiasi		Hubungan antara aktor dengan use case yang menunjukan adanya interaksi aktor dengan sistem atau campur tangan aktor dalam suatu use case
Aktor		Entitas eksternal yang berhubungan dengan sistem

2. *Class Diagram*

Nama Simbol	Simbol	Keterangan
Class		Sebuah kelas pada struktur sistem. Simbol ini memiliki tiga susunan, yaitu nama kelas, atribut dan operasi
Asosiasi		Relasi antar kelas dengan makna umum.

3. *Activity Diagram*

Nama Simbol	Simbol	Keterangan
Kondisi Awal		Menunjukan kondisi awal kegiatan sistem
Kondisi Akhir		Menunjukan kondisi berakhirnya kegiatan sistem
Action State		Kegiatan yang dilakukan oleh sistem atau pengguna
Desicion		Percabangan dimana akan dilakukan salah satu dari dua pilihan yang kondisinya terpenuhi
Arah Aktivitas		Menunjukan arah sebuah *action state* ke *action state* atau *decision* selanjutnya

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Sistem rekomendasi lokasi adalah sebuah sistem yang merekomendasikan tempat-tempat yang mungkin ingin dikunjungi oleh pengguna berdasarkan pola pengguna (*Next Point-Of-Interest*). Selain untuk meberikan rekomendasi tempat kepada pengguna, hasil dari sistem rekomendasi ini juga sangat penting dalam perkembangan para pebisnis untuk mendapatkan pelanggan yang berpotensi menguntungkan usaha mereka lebih banyak (He, Li dan Liao, 2017).Tetapi walau memiliki banyak keuntungan, penelitian yang mengambil topik tentang *next point-of*-interest masih menemui kesulitan yaitu untuk mengumpulkan data *check-in* perorangan (He, Li dan Liao, 2017).

Banyak metode yang sudah dilakukan untuk memenuhi kekurangan ini, seperti *Matrix Factorization* (Gambs dan Killijian, 2012), dan *neural networks*. Dalam jurnal yang menggunakan metode *matrix factorization,* digunakan algoritma *naive bayes,* atau algoritma *markov chain* untuk menentukan rekomendasi menggunakan metode kemungkinan atau persentase. Kedua algoritma tersebut memerlukan jumlah data yang mencukupi untuk membuat persentase, jika tidak, maka nilai persentase dihasilkan akan bernilai kecil (Gambs dan Killijian, 2012). Maka dipilihlah *neural network* sebagai salah satu alternatif pemecahan masalah ini. *Neural Network*, akan mempelajari data input sebagai sebuah kesatuan. Pola data akan disimpan dalam setiap *node*, dan akan dipindahkan ke *node* berikutnya jika memenuahi nilai *treshold*. Dalam penulisan ini, dipilihlah sebuah metode bernama *Bidirectional Long Short-Term Memory*. Metode ini diplih karena data yang akan diproses bersifat sekuensial. Hal ini dibuktikan dari dataset yang digunakan memiliki *utcTimeStamp*. Sehingga dapat disimpulkan bahwa data yang diperoleh dapat diolah menjadi data yang bersifat sekuensial.

Dari pemaparan di atas maka penelitian ini memiliki tujuan untuk meneliti tentang pengaruh penggunaan metode *Bidirectional LSTM* dengan menggunakan dataset *Foursquare* NYC terhadap akurasi dari model yang dihasilkan. Maka penelitian ini akan

mengangkat judul “Penerapan Algoritma *Bi-Directional Long Short-Term Memory* Dalam Sistem Rekomendasi Lokasi Berdasarkan Data Pengguna”

1.2 Rumusan Masalah

Masalah yang hendak diangkat dalam penlitian ini yaitu:

1. Berapa akurasi penerapan algoritma *bidirectional Long Short-Term Memory* dalam merekomendasikan tempat kepada pengguna dengan data yang bersifat dinamis?
2. Parameter apa saja yang berpengaruh dalam penerapan algoritma *bidirectional Long Short-Term Memory* untuk sistem rekomendasi?

1.3 Tujuan

Tujuan dari penelitian ini adalah:

1. Mendapatkan akurasi yang dihasilkan dari penerapan algoritma *bidirectional long short-term memory* dalam merekomendasikan tempat kepada pengguna dengan data yang bersifat dinamis
2. Mendapatkan parameter yang memiliki pengaruh dalam meberikan rekomendasi tempat kepada pengguna menggunakan algoritma *bidirectional long short-term memory*.

1.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Data yang diambil untuk dataset dari kaggle.com (*foursquare*)
2. Untuk menghasilkan prediksi dalam sistem yang akan dibuat, pengguna harus memasukan 4 lokasi.

1.5 Kegunaan Hasil

Adapun kegunaan hasil yang diharapkan dari pembuatan sistem ini bagi masyarakat adalah mendapatkan lokasi rekomendasi dari *trajectory* pengguna. Hasilnya akan ditampilkan dalam bentuk lokasi *latitude* dan *longitude* dan digambarkan dalam GPS. Sehingga kegunaan protipe yang dihasilkan adalah untuk menampilkan hasil rekomendasi lokasi yang nantinya bisa dikembangkan menjadi sebuah aplikasi untuk memberikan rekomendasi lokasi lanjutan bagi pengguna,

1.6 Metodologi Penelitian

Tahapan yang dilakukan dalam penelitian ini adalah sebagai berikut:

1. Studi Pustaka

   Penelitian ini mengumpulkan jurnal-jurnal yang menjadi landasan teori dari penelitian ini. Jurnal-jurnal diambil dari *google scholar* dan digunakan sebagai studi pustaka dalam penelitian ini.

2. Pengumpulan Data

   Setelah mempelajari literatur, langkah selanjutnya adalah mengumpulkan data. Data yang digunakan dalam penelitian ini didapatkan dari data *check-in* pengguna di foursquare yang dapat diakses dari kaggle.com.

3. Pemodelan Data

   Setelah mendapatkan data, kemudian data tersebut dimodelkan agar bisa digunakan sebagai data *training*.

4. Training Data

   Dataset yang sudah ada akan di *training* dengan pemodelan yang sudah disiapkan

5. Testing Data

   Dataset yang sudah disediakan sebagai data *testing* akan di test pada tahap ini sehingga menghasilkan tingkat akurasi dalam menentukan lokasi

6. Evaluasi

   Setelah melalui fase data *testing*, hasil akan dievaluasi untuk menilai kekurangan dan kelebihan dari pemodelan yang sudah dibuat. Hasil akan ditaruh pada bab 4, sementara evaluasi akan ditaruh pada bab 5.

1.7 Sistematika Penulisan

Penulisan ini dibuat dengan susunan sistematika penulisan sebagai berikut:

BAB I LATAR BELAKANG

Menjelaskan tentang latar belakang penelitian, rumusan masalah, tujuan penelitian dan batasan masalah

BAB II LANDASAN TEORI

Menjelaskan tentang teknik pembangunan sistem, perangkat lunak yang digunakan, set data yang digunakan, dan algoritma yang digunakan.

BAB III ANALISIS SISTEM

Menjelaskan tentang penjelasan singkat sistem, dan diagram jalan nya sistem tersebut serta algoritma yang digunakan.

BAB IV PERANCANGAN SISTEM

Menjelaskan tentang implementasi sistem, hasil dari pengujian, dan evaluasi dari sistem yang dibuat

BAB V KESIMPULAN DAN SARAN

Menjelaskan tentang kesimpulan dan saran yang untuk penelitian lanjutan yang akan datang.

BAB II

LANDASAN TEORI

2.1 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah suatu kumpulan kata yang terdiri dari: reakayasa, dan perangkat lunak. Rekayasa menurut kamus *Cambridge* disebut dengan *engineer* adalah:

"*A person whose job is to design or build machines, engines, or electrical equipment, or things such as roads, railways, or bridges, using scientific principles"* (*Cambridge*)
Sangat sering pengartian dari rekayasa atau *engineer* diartikan sebagai teknisi, namun menurut kamus *Cambridge* perngertian dari rekayasa sendiri atau *engineer* adalah penerapan sains yang digunakan untuk merancang sesuatu. Sehingga dapat disimpulkan bahwa rekayasa atau *engineer* adalah orang yang menerapkan sains dan matematika dalam pekerjaan mereka merancang dan menciptakan serta merawat sesuatu yang berkaitan dengan penerapan sains.

Perangkat lunak menurut Sommerville dalam bukunya yang berjudul "*Software Engineer-Tenth Edition*" adalah:

"*Software systems are abstract and intangible. They are not constrained by the properties of materials, nor are they governed by physical laws or by manufacturing processes. This simplifies software engineering, as there are no natural limits to the potential of software.*" (Sommerville, 2016:18)
Selain itu, dalam buku yang sama, Sommerville menjelaskan kembali tentang perangkat lunak sebagai berikut:

"However, when we are talking about software engineering, software is not just the programs themselves but also all associated documentation, libraries, support websites, and configuration data that are needed to make these programs useful." (Sommerville, 2016:19)
Sementara perangkat lunak menurut Pressman dalam bukunya yang berjudul *Software Engineer a Practitioner's Approach* mengartikan *software* adalah sesuatu yang lebih bersifat logika daripada elemen yang bersifat fisik (Pressman dan Maxim, 2020:).

Sehingga dapat disimpulkan bahwa perangkat lunak adalah suatu kumpulan kode yang dijalankan dalam suatu perangkat keras. Perangkat lunak tidak memiliki bentuk fisik, atau material pabrik sehingga memungkinkan pertumbuhan dan perkembangan yang pesat.

Rekayasa perangkat lunak menurut Ian Sommervile dalam bukunya yang berjudul "*Software Engineering-Ninth Edition*" adalah:

"Software engineering is an engineering discipline that is concerned with all aspects of software production from initial conception to operation and maintenance." (Sommerville, 2016:20)

Dari pernyataan di atas dapat disimpulkan bahwa rekayasa perangkat lunak adalah sebuah ilmu yang memperhatikan semua aspek perangkat lunak yang bermula dari perancangan konsep atau analisis, implementasi, dan perawatan. Rekayasa perangkat lunak bukan hanya membuat suatu perangkat lunak, tetapi juga harus menyelesaikan sampai tahap perawatan perangkat lunak itu sendiri.

2.2 Metodologi Penelitian

Metodologi penelitian yang dipakai dalam penulisan ini adalah metode penelitian yang berorientasi objek. Menurut Wessfeld dalam bukunya yang berjudul *The Object-Oriented Thought Process* mengatakan bahwa objek adalah sebagai berikut:

"For example, when you look at a person, you see the person as an object. And an object is defined by two components: attributes and behaviors. A person has attributes, such as eye color, age, height, and so on. A person also has behaviors, such as walking, talking, breathing, and so on. In its basic definition, an object is an entity that contains both data and behavior." (Weisfeld, 2019:14)

Dari pernyataan di atas, maka dapat disimpulkan bahwa objek adalah suatu unit yang memiliki tugas khusus yang menjadi perwakilan dari entitas permasalahan yang ingin dipecahkan. Dengan kata lain, objek adalah entitas dalam metodologi penelitian berorientasi objek. Pemograman berorientasi objek pertama kali diperkenalkan pada tahun1990, namun belum sangat populer karena para praktisi di bidang komputer pada saat itu lebih terbiasa menggunakan pemograman secara struktural.

Pemograman berorientasi objek menurut Dan Clark dalam bukunya yang berjudul *"Beginning C# Object-Oriented Programming, 2nd Edition"* adalah:

"Object-oriented programming is an approach to software development in which the structure of the software is based on objects interacting with each other to accomplish a task. This interaction takes the form of messages passing back and forth between the objects. In response to a message, an object can perform an action."(Clark, 2013:1)

Sementara menurut Wessfeld, pemograman berorientasi objek adalah:

"*The fundamental advantage of OO programming is that the data and the operations that manipulate the data (the code) are both encapsulated in the object. For example, when an object is transported across a network, the entire object, including the data and behavior, goes with it.*" (Weisfeld, 2019)

Sehingga dapat disimpulkan bahwa pemograman berorientasi objek adalah pengembangan perangkat lunak yang strukturnya berdasarkan interaksi antara objek yang dibuat untuk menyelesaikan suatu tujuan.

2.3 Model Pengembangan Rational Unified Process (RUP)

Menurut Krutchen, dalam bukunya yang berjudul *the rational unified process made easy: a practitioner's guide to the RUP* mengatakan bahwa dalam pengartiannya, RUP memiliki banyak definisi tergantung kepada siapa pertanyaan itu ditujukan. Tetapi, pada intinya, menurut Krutchen, model pengembangan RUP memiliki 3 poin penting yaitu:

1. *The RUP is a software development approach that is iterative, architecture-centric, and use-case-driven. It is described in a variety of whitepapers and books. The most comprehensive information can be found in the RUP product itself, which contains detailed guidelines, examples, and templates covering the full software lifecycle.*
2. *The RUP is a well-defined and well-structured software engineering process. It clearly defines who is responsible for what, how things are done, and when to do them. The RUP also provides a well-defined structure for the lifecycle of a RUP project, clearly articulating essential milestones and decision points.*
3. *The RUP is also a process product that provides you with a customizable process framework for software engineering. The RUP product supports process customization and authoring, and a wide variety of processes, or Process Configurations, can be assembled from it. These RUP configurations can be made to support small or large teams and disciplined or less-formal approaches to development. The RUP product contains several out-of-the-box Process Configurations and Process Views that guide analysts, developers, testers, project managers, configuration managers, data analysts, and other team members in how to develop software. The RUP is used by a wide variety of companies in different industry sectors*. (Kroll dan Kruchten, 2003)

Dari tiga poin di atas, maka dapat disimpulkan bahwa model pengembangan RUP adalah model pengembangan yang memiliki iterasi dalam implementasinya, kejelasan dalam pelaksanaan dan *checkpoint* dan juga metode ini dapat digunakan secara fleksibel dalam pembangunan projek yang berskala besar atau kecil. Hal ini dimungkinkan karena dalam setiap iterasi atau *step* yang ada dari flow (gambar 2.1) memiliki langkah atau proses yang melanjutkan dari proses sebelumnya. Dalam model pengembangan RUP, ada beberapa prinsip yang harus diikuti oleh setiap pengguna yang menggunakan model ini. Beberapa prinsip yang harus diikuti menurut Krutchen adalah:

1. *Attack major risks early and continuously…or they will attack you.*
2. *Ensure that you deliver value to your customer.*
3. *Stay focused on executable software.*
4. *Accommodate change early in the project.*
5. *Baseline an executable architecture early on.*
6. *Build your system with components.*
7. *Work together as one team.*
8. *Make quality a way of life, not an afterthought.* (Kroll dan Kruchten, 2003)

Menurut Krutchen, ada beberapa kesalahan yang sering dilakukan oleh pengguna RUP. Menurut Krutchen, kesalahan yang paling fatal adalah menyamakan RUP dengan metode *waterfall*. Kedua metode ini sangat berbeda, RUP pada setiap fase nya didorong oleh kegagalan dari fase sebelumnya, sehingga setiap fase memiliki masa perbaikan sebelum masuk ke fase selanjutnya. Pada *waterfall* terkadang hal ini dihilangkan, sehingga tidak ada perbaikan dalam setiap fase(Kroll dan Kruchten, 2003). Dalam penerapannya RUP memiliki 4 langkah yang harus diikuti dalam setiap pengembangan perangkat lunak. Langkah-langkah tersebut adalah:

1. *Inception Phase*

   Fase ini adalah fase awal dalam penggunaan pemodelan RUP. Pada fase ini, pembuat sistem harus benar-benar mengerti tentang sistem yang akan dibuat. Pembuat sistem harus memahami cakupan, jangkauan dan kegunaan dari sistem yang hendak dirancang. Ada beberapa step yang harus diperhatikan dalam fase ini, yaitu (Kroll dan Kruchten, 2003:106):

   1. Understand what to build. Determine the vision, the scope of the system, and its boundaries, that is, what is inside the system and what is outside. Identify who wants this system and what it is worth to them.

   2. Identify key system functionality. Decide which use cases (which ways of using the system) are most critical.

   3. Determine at least one possible solution. Identify at least one candidate architecture.

   4. Understand the costs, schedule, and risks associated with the project.

   5. Decide what process to follow and what tools to use.

   Dari poin yang ditulis di atas maka dapat disimpulkan bahwa tahap *inception* adalah tahapan yang berguna untuk memahami sistem yang akan dibuat, bagaimana cara membuat sistem tersebut, mengerti pembayaran dan penjadwalan, dan mengetahui *tools* yang dibutuhkan untuk membangun perangkat lunak tersebut.

2. *Elaboration Phase*

   Pada fase ini, menurut Krutchen, pembuat sistem dapat mengolah hal-hal yang lebih terperinci yang harus dipersiapkan oleh pembuat sistem sehingga dapat membuat perencanaan yang lebih mendetail. Selain itu, pada fase ini, pembuat sistem dapat

mengimplementasi sistem secara luar saja, seperti pembuatan blok diagram atau *activity* diagram. Hal ini diperlukan untuk memastikan setiap kebutuhan dari *tools* yang telah direncanakan sudah terpenuhi. Lalu untuk tahap akhir, memastikan semua kebutuhan terpenuhi sebelum masuk ke dalam tahap *construction*. (Kroll dan Kruchten, 2003:122)

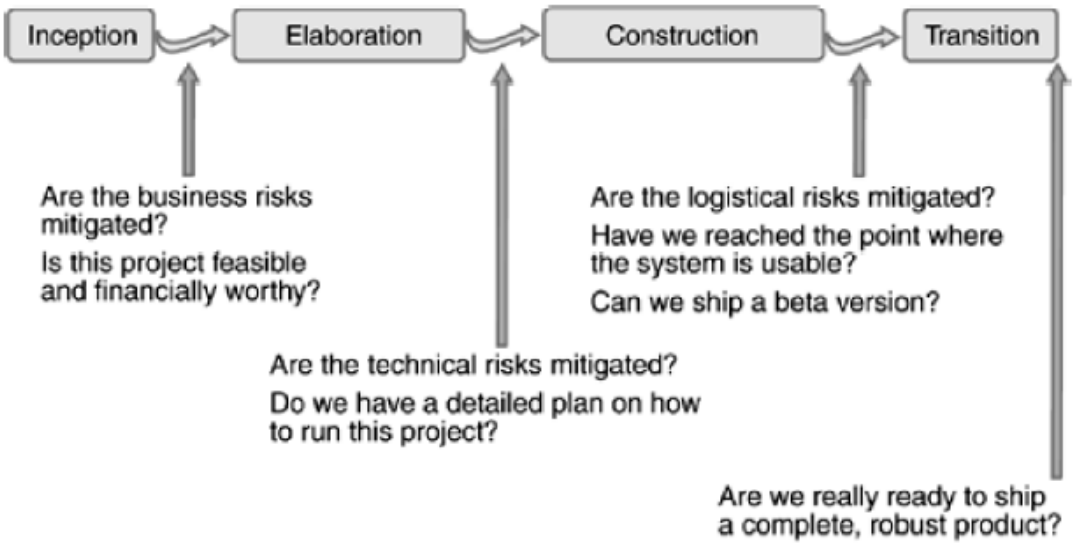
3. *Construction Phase*

   Pada fase ini, menurut Krutchen, pembuat sistem dapat memulai untuk menciptakan sistem yang siap untuk diluncurkan kepada pengguna. Pada tahap ini terdiri dari beberapa iterasi yang setiap iterasi nya memiliki fungsi sebagai peng efektifan setiap komponen seperti kodingan atau pembiayaan dan perancangan waktu. Kegunaan dari fase ini yang paling penting adalah mulai nya pengembangan sistem yang siap diluncurkan ke pengguna. (Kroll dan Kruchten, 2003:145)

4. *Transition Phase*

   Fase ini adalah fase terakhir dari proses RUP. Pada fase ini, menurut Krutchen, aplikasi atau sistem sudah siap untuk di test oleh *beta-tester*, untuk mengetest apakah sistem yang dibuat sudah sesuai dengan permintaan klien. Setelah tahap test selesai, dilanjutkan dengan tahap pelatihan pengguna. Tahap ini adalah tahap kedua ketika test sudah disetujui oleh klien. Tahap selanjutnya adalah mempersiapkan untuk peluncuran perangkat lunak. Pada tahap ini, harus mempersiapkan dokumentasi lengkap untuk penggunaan perangkat luank tersebut. Dan terakhir, improvisasi untuk proyek ke depan yang diambil dari hasil evaluasi proyejk yang sudah selesai. (Kroll dan Kruchten, 2003:162)

Dari keempat fase di atas, terdapat beberapa *checkpoint/milestone*. *Checkpoint* ini adalah indikator keberhasilan dari suatu *step* yang telah dilewati. Indikator tersebut dapat dilihat pada gambar 2.1



Gambar 2.1
Pengembangan Model RUP

(Sumber:Kroll and Kruchten, 2003)

Dari seluruh penjabaran diatas, dapat disimpulkan bahwa metode RUP adalah metode yang digunakan bukan hanya untuk membagun perangkat lunak namun untuk memastikan bahwa perangkat lunak yang dibuat sudah sesuai dengan kebutuhan pengguna. Selain untuk model pengembangan, metode ini juga digunakan untuk membantu pembuat sistem untuk mengerjakan sistem secara sistematis dan terstruktur sehingga meminimalisir kesalahan dari sistem atau perangkat lunak yang dibuat.

2.4 UML (Unified Model Language)

Definisi UML menurut Steve Tockey:

… provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes.

Definisi UML menurut Benhard Rumpe:

UML is used as notation for a range of activities, such as modeling business cases, analyzing the current and re quired form of a system as well as architecture and preliminary and detailed design at various levels of granularity. The artifacts of UML are therefore an important foundation for planning and controlling software development projects that are based on milestones. (Rumpe, 2017)

Diagram UML terdiri dari beberapa jenis yaitu (Unhelkar, 2018:14-37):

1. Diagram Struktur

Diagram struktur menggambarkan hal-hal yang akan dirancang didalam sistem. Sebagai contoh diagram kelas (salah satu jenis diagram struktur) menunjukan hubungan antar kelas-kelas mewakili objek-objek dalam sistem.

a. Class diagram

Class diagram menunjukkan sekumpulan kelas, antarmuka, dan kolaborasi serta hubungannya. Diagram ini adalah diagram yang paling umum ditemukan dalam pemodelan sistem berorientasi objek. Diagram kelas membahas tampilan desain statis dari suatu sistem.

b. Object diagram

Object diagram memperlihatkan sekumpulan objek dan hubungannya. *Object diagram* merepresentasikan snapshot statis dari berbagai hal yang ditemukan di diagram kelas. Diagram ini membahas tampilan desain statis atau tampilan proses statis sistem seperti halnya diagram kelas, tetapi dari perspektif kasus nyata atau prototipe.

c. Component diagram

Component diagram menunjukkan organisasi dan ketergantungan di antara sekumpulan komponen. *Component diagram* membahas tampilan implementasi statis dari suatu sistem dan terkait dengan diagram kelas di mana sebuah komponen biasanya dipetakan ke satu atau lebih kelas, antarmuka, atau kolaborasi.

d. *Composite Structure Diagram*

Composite Structure menunjukan struktur dalam suatu kelas dan kolaborasi yang dapat diizinkan terjadi di dalam kelas tersebut.

e. *Package Diagram*

Menjelaskan hubungan antar paket yang membentuk sistem. Misalnya, jika satu paket dalam sistem menggunakan fitur yang disediakan oleh paket lain, maka diagram akan menunjukkan "mengimpor" terlebih dahulu.

f. *Deployment Diagram*

Merupakan diagram yang menjelaskan penyebaran dari artefak pada sebuah *node* (perangkat keras atau linkungan tempat artefak tersebut berada)

2. Diagram Perilaku

   a. U*se case diagram*

      Use case diagram menunjukkan sekumpulan kasus penggunaan dan aktor (jenis kelas khusus) dan hubungannya. *Use case diagram* membahas tampilan use case statis dari suatu sistem. Diagram ini sangat penting dalam mengatur dan memodelkan perilaku sistem.

   b. *Activity diagram*

      Activity diagram adalah jenis diagram statechart khusus yang menunjukkan aliran dari aktivitas ke aktivitas dalam suatu sistem. *Activity diagram* membahas tampilan dinamis dari suatu sistem dan dapat digunakan dalam memodelkan fungsi sistem dan menekankan aliran kontrol di antara objek

   c. *State Machine diagram*

      Statechart diagram menunjukkan *state machine*, yang terdiri dari status, transisi, peristiwa, dan aktivitas. *Statechart diagram* berguna dalam pemodelan perilaku antarmuka, kelas, atau kolaborasi

3. *Interaction Diagrams*

Interaction Diagram adalah cabang dari activity diagram. Memiliki kegunaan yang hampir mirip dengan activity diagram yaitu menggambarkan proses yang terjadi dalam sistem secara berurutan.

   a. *Sequence diagram*

      Sequence diagram adalah jenis diagram interaksi yang terdiri dari sekumpulan objek dan hubungannya, termasuk pesan yang mungkin dikirim di antara mereka. Diagram interaksi membahas tampilan dinamis dari suatu sistem.

   b. *Communication Diagram*

      Diagram ini memiliki kemiripan dengan *sequence*. Diagram ini menggambarkan relasi komunikasi antar objek dalam suatu aktivitas atau kolaborasi. Perbedaan dari kedua diagram ini adalah, *sequence diagram* lebih menggambarkan

pesan yang disampaikan sementara *communication diagram* lebih menggambarkan hubungan atau relasi antar objek.

c. *Timing Diagram*

Diagram waktu menunjukkan satu atau beberapa perubahan status objek dari waktu ke waktu. Diagram waktu sangat mirip dengan diagram urutan yang diputar ke samping, sehingga waktu bertambah dari kiri ke kanan. Diagram ini dapat berguna untuk memberikan gambaran tentang berapa lama waktu yang dibutuhkan untuk bagian-bagian yang berbeda dari sebuah skenario.

Dari pernyataan diatas, maka dapat disimpulkan bahwa UML adalah sebuah alat/*tools* yang digunakan oleh pembuat sistem untuk memaparkan sistem yang akan dibuat atau dikembangkan.

2.5 Sistem Rekomendasi

Sistem rekomendasi adalah perangkat lunak dan suatu metode yang menyediakan rekomendasi untuk *item* yang menarik bagi pengguna tertentu. Rekomedasi berhubungan dengan berbagai proses membua keputusan, seperti barang untuk dibeli, musik untuk didengar atau berita untuk dibaca (Aggarwal, 2016). Dalam buku yang berjudul *Recommender System Handbook* yang ditulis oleh Francesco Ricci, Lior Rokach, dan Bracha Sphira mengatakan bahwa sdalam prosesnya, sistem rekomendasi memerlukan suatu parameter yang disebut dengan *item*:

“Item” is the general term used to denote what the system recommends to users. An RS normally focuses on a specific type of item (e.g., CDs or news) and, accordingly its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item.(Ricci et al., 2015:19)

Dalam kalimat lain, Francesco Ricci, Lior Rokach, dan Bracha Sphira juga menyatakan bahwa “*items*” yang digunakan bisa apa saja tergantung dari kebutuhan sistem tersebut. Dalam paragraf yang lain, mereka menyebutkan beberapa contoh:

A prime example is a book recommender system that assists users in selecting a book to read. On the popular website, Amazon.com, the site employs an RS to personalize the online store for each customer (Ricci et al., 2015:19)

Dalam buku yang berjudul *Recommender System: The TextBook* karangan Charu C Aggrawal, diberikan beberapa contoh studi kasus yang terjadi di dalam *internet* atau website yang menerima masukan berupa *ratings* atau *feedback* dari pengguna:

For example, consider a scenario of a content provider such as Netflix. In such cases, users are able to easily provide feedback with a simple click of a mouse. A typical methodology to provide feedback is in the form of ratings, in which users select numerical values from a specific evaluation system (e.g., five-star rating system) that specify their likes and dislikes of various items. (Aggarwal, 2016:1)

Other forms of feedback are not quite as explicit but are even easier to collect in the Web-centric paradigm. For example, the simple act of a user buying or browsing an item may be viewed as an endorsement for that item. Such forms of feedback are commonly used by online merchants such as Amazon.com, and the collection of this type of data is completely effortless in terms of the work required of a customer. (Aggarwal, 2016:1)

Sehingga menurut Aggrawal, sistem rekomendasi adalah suatu sistem yang memiliki ide dasar yaitu untuk memanfaatkan berbagai sumber data pengguna untuk menyimpulkan minat dari pengguna tersebut. Lebih lanjut lagi, menurut Aggrawal, sistem rekomendasi ini memanfaatkan data produk sebagai "*items*" terhadap pengguna. Sistem rekomendasi juga menggunakan data yang bersumber dari pelanggan tersebut pada transakso sebelumnya sehingga kebutuhan sistem untuk menyimpan *history* pengguna sangat diperlukan untuk sistem rekomendasi(Aggarwal, 2016). Sementara menurut Falk dalam bukunya yang berjudul *Practical Recommender System* memberikan kesimpulan tentang sistem rekomendasi yaitu:

A recommender system calculates and provides relevant content to the user based on knowledge of the user, content, and interactions between the user and the item. (Falk, 2019:13)

Sistem rekomendasi dapat dibagi menjadi 2 jenis yaitu *content based* dan *collaborative*:

1. *Content Based Recommender System*

   Menurut Pramod Singh dalam bukunya yang berjudul *Machine Learning with PySpark: With Natural Language Processing and Recommender Systems* mengatakan bahwa *content based recommedation* adalah rekomendasi yang merekomendasikan sesuatu (barang, film, lokasi, dll) kepada pengguna berdasarkan kesamaan *item* yang dibeli atau diberi rating oleh pengguna tersebut di masa lalu. Selain berdasarkan *item* yang disukai, *content-based* juga dapat mengambil nilai dari *item* yang paling sering dilihat atau dikunjungi (dalam toko online), film yang paling sering ditonton (dalam aplikasi

film) atau barang yang paling sering dibeli. Biasanya, sistem ini menggunakan kategori dari *item* tersebut untuk menghasilkan rekomendasi kepada pengguna.

2. *Collaborative Filtering Base*

   Menurut Pramod Singh, *collaborative filtering* adalah contoh dari sistem rekomedasi yang menggunakan kesamaan barang kesukaan antara pengguna satu dengan pengguna lain nya. Sehingga barang atau *item* yang direkomendasikan kepada pengguna bisa jadi adalah barang atau *item* yang disukai oleh pengguna lain yang memiliki kesamaan pada barang lain yang disukai oleh pengguna tersebut. Sebagai contoh, *youtube* memberi rekomedasi video untuk pengguna berdasarkan kategori video yang sering ditonton pengguna dan juga berdasarkan kategori video yang ditonton oleh pengguna lain yang memiliki kemiripan kategori rekomedasi dengan pengguna pertama.

Dari beberapa pernyataan di atas, maka sistem rekomendasi dapat disimpulkan sebagai sebuah sistem yang merekomendasikan item kepada pengguna berdasarkan relasi antara pengguna tersebut dan item yang bersangkutan. Relasi tersebut dapat terbentuk dari perilaku pengguna yang memberi penilaian kepada item tersebut atau memberi *feedback* kepada item yang bersangkutan.

2.6 Recurrent Neural Network

Dalam jurnal yang ditulis oleh Kiperwasser dan Goldberg, menyatakan bahwa pengertian RNN adalah sebagai berikut:

Recurrent neural networks (RNNs) are statistical learners for modeling sequential data. An RNN allows one to model the ith element in the sequence based on the past – the elements x1: i up to and including it. (Kiperwasser dan Goldberg, 2016)

Dalam sebuah buku yang berjudul *Recurrent Neural Networks for Short-Term Load Forecasting* yang ditulis oleh Bianchi mengatakan bahwa *neural network* adalah

RNNs are learning machines that recursively compute new states by applying transfer functions to previous states and inputs. Typical transfer functions are composed by an affine transformation followed by a nonlinear function, which are chosen depending on the nature of the particular problem at hand. (Bianchi *et al.*, 2017)

RNN disebut sebagai alat efektif yang dapat memproses serta menyimpan urutan data temporal, RNN juga dapat menangkap dan juga memproses data non-linear. Dalam pengertian lain nya, RNN juga dapa disebut sebagai abstraksi dari sistem biologi dalam

bentuk matematika, yang bisa me-*mappings* dari *input sequence* ke *output sequence*. Oleh sebab itu kebanyakan dari RNN menggantungkan hasil prediksi pada *error gradient*. *Error gradient* inilah yang menjadi penentu apakah RNN yang digunakan adalah RNN yang statik atau dinamis. *Error gradient* tidak hanya berpengaruh pada *input, output,* dan target namun lebih kepada susunan *history* data yang diberikan ke RNN sehingga dalam proses *training*, RNN menjadi algoritma yang cukup sulit dalam menentukan keefektifan nya.

Dalam jurnal yang ditulis oleh Chung, et al. diperlukan suatu algoritma baru yang mengatasi masalah *long-term* data (Chung *et al.*, 2014). Menurut Chung et al. dalam jurnal yang berjudul *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling* mengatakan bahwa untuk mengatasi data yang bersifat *long-term* dapat menggunakan turunan dari *recurrent neural network* yaitu *gated recurrent neural network.*

2.6.1 *Bidirectional Layer*

Dalam sebuah jurnal yang ditulis oleh Kiperwasser dan Goldberg, *Bidirectional* layer adalah sebuah layer yang ditambahkan ke dalam *Recurrent layer* (LSTM, GRU, RNN, ConvLSTM) yang akan memecah jalannya sebuah neuron menjadi dua yaitu *foward states* dan *backward states* (Kiperwasser and Goldberg, 2016). Hasil yang dihasilkan dari antara kedua output (*backward* dan *foward*) akan dikalkulasi untuk menghasilkan *output* akhir yang nanti akan dihasilkan oleh hasil prediksi. Penggunaan *bidirectional layer* akan memberikan *output* terbaik pada penulisan ini dikarenakan penulisan ini sangat bergantung pada pola perjalanan pengguna.

2.6.2 Gated Recurrent Neural Network

Menurut Chung et al. *gated neural network* dibagi menjadi dua jenis yaitu LSTM (*Long Short-Term Memory*) dan GRU (*Gated Recurrent Unit*).

1. LSTM (*Long Short-Term Memory*)

Long Short-Term Memory diusulkan oleh Hochreiter dan Schmidhuber (1997). Menurut Hochreiter and Schmidhuber dalam jurnal nya yang berjudul *Long Short-Term Memory* mengatakan bahwa LSTM memiliki *cell* yang tidak dimiliki oleh jenis *neural network* yang lain. *Memory Cell* adalah *cell* khusus yang dimiliki oleh LSTM. *Cell* ini berfungsi untuk menyimpan nilai *error* secara konstan sehingga tidak akan terjadi

pengurangan atau peledakan nilai gradien seiring bertambahnya waktu. Hal ini membuat LSTM dapat diberi data yang bersifat *long-term* dan mengeluarkan hasil yang sesuai.

Tidak seperti *recurrent unit* yang hanya menjumlahkan nilai *weight* dari *input* dan mengaplikasikan pada fungsi non-linear, setiap sel dalam LSTM memiliki *memory* dalam satu waktu, sehingga output h atau *activation unit* untuk setiap unit LSTM adalah:

$$h_t^j = a_t^j \tanh\left(c_t^j\right)$$

Dimana a_t^j adalah *output gate* yang mengatur jumlah *memory*. Dalam perhitungannya, output gate dapat dihitung menggunakan:

$$a_t^j = \sigma(W_0 X_t + U_c h_{t-1})^j$$

Dimana σ adalah fungsi sigmoid. Persamaan untuk mengupdate nilai sel *memory* adalah:

$$c_t^{-j} = tanh(W_c X_t + U_c h_{t-1})^j$$

Nilai dari sel *memory* yang baru dihitung dengan menggunakan nilai dari *forget gate* dan *input gate* dalam satu satuan waktu t. Sehingga dapat dirumuskan menjadi berikut:

$$f_t^{\ j} = \sigma(W_f X_t + U_f h_{t-1} + V_f C_{t-1})^j$$

$$i_t^j = \sigma(W_i X_t + U_i h_{t-1} + V_i C_{t-1})^j$$

Tidak seperti RNN tradisional yang selalu mengganti nilai konten dalam setiap satuan waktu, LSTM dapat menentukan apakah nilai yang sudah ada tetap disimpan atau diganti. Sehingga jika dari data pertama, LSTM unit mendeteksi bahwa info tersebut adalah penting, maka info tersebut akan dibawa dari awal sampai akhir sehingga berpotensi untuk mendapatkan data yang bersifat *long-term*.

2. GRU (*Gated Recurrent Unit*)

GRU diusulkan oleh Chung et al. (2014) untuk membuat setiap unit *recurrent* secara adaptif mendapatkan nilai dependensi dalam setiap satuan waktu yuang berbeda. Sama seperti LSTM unit, GRU juga memiliki unit *gate* yang berfungsi untuk menentukan nilai informasi di dalam unit tersebut hanya saja tidak memiliki sel *memory*.

Untuk fungsi aktivasi h_t^j dari GRU dalam satuan waktu t adalah interpolasi linear dari aktivasi sebelumnya dan aktivasi kandidat.

$$h_t^j = (1 - z_t^j)h_{t-1}^j + \ z_t^j h_t^j$$

Update gate dibuat untuk menentukan berapa banyak unit yang meng-*update* nilai aktivasi atau konten. *Update gate* dihitung dengan menggunakan persamaan:

$$z_t^j = \sigma(W_z X_t + U_z h_{t-1})^j$$

Prosedur ini menjumlahkan nilai linear antara *existing* state dan *state* yang baru dihitung, dalam hal ini sama dengan LSTM. Kandidat aktivasi dapat dihitung dengan menggunakan:

$$h_t^j = \tanh(W_z X_t + U(r_t \circ h_{t-1}))^j$$

Dimana r adalah *reset gate*. Dimana jika r mendekati angka 0, maka *reset gate* akan dengan mudah membca symbol pertama dari *input sequence,* sehingga hal tersebut dapat membuat GRU mengganti nilai dari *state* sebelumnya. *Reset gate* dapat dihitung menggunakan:

$$r_t^j = \sigma(W_r X_t + U_r h_{t-1})^j$$

2.6.3 Hasil *Training Recurrent Neural Network*

Ketika model sudah selesai di *training*. Maka akan menghasilkan beberapa yaitu akurasi, *loss*, RMSE, MSE, *precission, recall, f1_score*, dan MAPE. Dari hasil *output* tersebut, maka dipilihlah beberapa *output* yang sesuai untuk penelitian berdasarkan data yang digunakan. Dalam penelitian ini, peneliti mengambil nilai akurasi dan RMSE dari model. Pemilihan akurasi dilakukan karena hasil akhir dari model ini adalah untuk memprediksi lokasi akhir pengguna dan pemilihan RMSE digunakan untuk menentukan berapa perbedaan dari hasil prediksi dan hasil sebenarnya.

1. Akurasi

Akurasi adalah salah satu dari matrik performa. Angka akurasi didapatkan dari berapa jumlah data yang diprediksi benar dibagi dengan data sebenarnya. Semakin besar akurasi, mengartikan bahwa model ini dapat memprediksi semakin akurat. Penggunaan akurasi umumnya digunakan pada klasifikasi. Akurasi dapat dibedakan menjadi tiga jenis tergantung dari perbedaan akurasi antara data*training* dan data validasi (Patel 2019).

1. *Underfitting*

Underfitting adalah sebuah kondisi dimana *neural network* kekurangan jumlah *epoch* dan jumlah *node* sehingga keakuratan yang dihasilkan memiliki tingkat akurasi yang

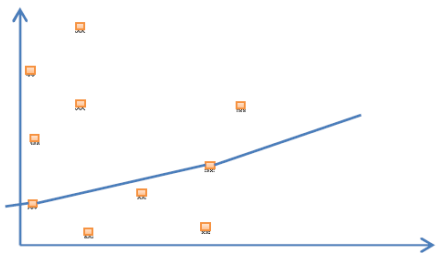
sangat kecil. Hal ini dapat berdampak buruk pada kualitas hasil data *testing* karena hasil yang dihasilkan dari *output neural network* tidak memberikan akurasi yang terbaik. Menurut Patel dalam artikelnya yang berjudul *Underfitting vs Overfitting* memberikan pemisalan untuk *underfitting* adalah:

Underfitting, it's like not studying enough and failing. (Patel, 2019)

Menurut Khan, dalam jurnal nya yang berjudul *Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning* menyatakan bahwa

Underfitting is occurs when the model is incapable of capturing the variability of the data. For example, suppose one is training a linear (y= ax+b not polynomial a and b are constant) classifier on a data set that is a parabola Tom Dietterich. This is the result of understanding or attempting to use a model which is too simple to describe a given set of data. (Jabbar dan Khan, 2015)

Sehingga dapat diambil kesimpulan, *underfitting* adalah sebuah kodisi dimana *neural network* tidak memiliki parameter yang memenuhi kesuksesan hasil *training*, sehingga hasil yang dihasilkan oleh *neural network* memiliki tingkat akurasi yang kecil. Hal ini tentu saja akan mempengaruhi hasil dari *training neural network*.



Gambar *2.2*
Contoh *Underfitting*

(Sumber: Jabbar dan Khan, 2015)

2. *Optimum*

Optimum adalah suatu kondisi dimana data hasil *training* memiliki tingkat akurasi yang stabil walaupun dengan data yang diganti-ganti. Hal ini dapat muncul karena jumlah *epoch* yang terbaik, atau nilai *weight* setiap node yang optimal. Menurut Patel dalam artikel nya yang berjudul *Overfitting vs Underfitting* menyebutkan bahwa kondisi *optimum* tidak selalu memiliki tingkat akurasi yang paling tinggi tetapi akurasi tersebut dapat dikatakan akurasi yang stabil.

3. *Overfitting*

Overfitting adalah sebuah hasil yang dihasilkan dari *training neural network* yang memiliki akurasi *training* tinggi namun akurasi tiba-tiba turun ketika di *testing*. Dalam beberapa literatur, jurnal dan artikel disebutkan bahwa *overfitting* adalah suatu kondisi yang umum terjadi dan terkadang sangat sulit untuk diidentifikasi sebelum data *testing* dimasukan untuk dilakukan pengetesan (Jabbar dan Khan, 2015) sehingga kondisi ini terkadang akan mempersulit untuk memastikan apakah hasil *training* sudah akurat atau masih *overfitting*.

Menurut Patel dalam artikel nya yang berjudul *Overfitting vs Underfitting* menyebutkan bahwa *underfitting* adalah kejadian di mana hasil yang dihasilkan oleh *neural network* memiliki tingkat keakuratan yang tinggi, namun jika diisi oleh data lain, maka tingkat ke akuratannya akan sangat menurun drastis. Dalam artikelnya, Patel memberikan gambaran tentang *overfitting* dengan sebuah pemisalan:

Overfitting is like instead of studying, we memorize the entire textbook word by word. We may be able to regurgitate any questions in the textbook but we won't be able to generalize properly and answer the questions in the test. (Patel, 2019)
Dalam jurnal yang ditulis oleh Jabbar dan Khan menyebutkan *overfitting* adalah:

Overfitting Is the one of biggest problem in training neural networks is the over-fitting of training data. That means that the neural network at the certain time during the training period does not improve its ability to solve problem anymore. But just starts to learn some random regularity contained in the set of training patterns. (Jabbar dan Khan, 2015)

Sementara menurut Ying, *overfitting* adalah:

Because of existence of overfitting, the model performs perfectly on training set, while fitting poorly on testing set. This is due to that over-fitted model has difficulty coping with pieces of the information in the testing set, which may be different from those in the training set. memorize all the data, including unavoidable noise on the training set, instead of learning the discipline hidden behind the data. (Ying, 2019)

Sehingga dapat disimpukan bahwa *overfitting* adalah sebuah kondisi dimana *neural network* terlalu mempelajari data training sehingga pemodelan yang dihasilkan tidak akan bisa fleksibel kepada berbagai data *testing*. Hal ini dapat dimisalkan seperti seseorang yang hanya menghafal dan seseorang yang paham, Seseorang yang paham, akan dengan mudah menyelesaikan berbagai permasalahan mengenai apa yang dipelajarinya, semetara orang yang menghafal, akan sangat mudah jatuh ketika mendapatkan persoalan yang berbeda dari yang dia hafalkan.

Dalam menyelesaikan masalah ini, maka disediakan beberapa cara(Ying, 2019):

1. *Early-Stopping*

Strategi ini digunakan untuk menghindari fenomena "*learning speed slow-down*". Masalah ini berarti akurasi algoritma berhenti meningkat setelah beberapa titik, atau bahkan semakin buruk karena *noise-learning*. Secara praktikal, metode ini akan terus men training setiap epoch sampai suatu titik dimana hasil akurasi yang didapatkan oleh epoch tersebut sudah tidak naik atau malah menurun. Menurut Ying, hal ini dapat berarti jika dilanjutkan, data hasil yang dihasilkan akan mengalami *overfitting.*

2. *Network Reduction*

Noise learning merupakan salah satu penyebab penting terjadinya *overfitting*. Jadi, pengurangan *noise* menjadi salah satu arah penelitian untuk menanggulangi *overfitting*. Berdasarkan pemikiran ini, *Network Reduction* diusulkan untuk mengurangi ukuran pengklasifikasi finial dalam pembelajaran relasional, terutama dalam pembelajaran keputusan. Dalam *reduction*, terdapat 2 jenis metode: *pre-prunning* dan *post-prunning*.

Pre-prunning adalah metode yang dilakukan pada saat *training* dilakukan. Biasanya, algortma ini akan memberikan atau mengurangi informasi yang dimiliki dalam proses *training*. Sementara itu, untuk *post-prunning* adalah algortima yang mengecualikan *overfitting*. *Post-prunning* akan membagi data *training* menjadi 2 jenis data, *growing set* dan *pruning set*. Metode ini akan menghapus kondisi atau peraturan dari model yang didapatkan dalam proses *training*.

3. *Expansion of Training Data*

Metode yang satu ini, bukan algoritma namun lebih ke arah penyelesaian menggunakan non-algoritma. Menurut Ying, jumlah data yang besar dan kompleks akan membuat hasil *training* menjadi lebih baik. Namun metode ini memiliki kekurangan yaitu dalam penambahan data, berarti akan ada penambahan waktu *training*. Pendekatan metode ini adalah:

1. Menambahkan jumlah data
2. Menambahkan *noise* secara acak
3. Mendapatkan kembail data dari *data set* yang dijalankan dalam training
4. Membuat data baru berdasarkan distribusi data yang ada

4. *Regularization*

Menurut Ying, dalam beberapa keadaan, kondisi *overfitting* terjadi karena *neural network* masih memperhitungkan data yang terdapat dalam model walaupun data tersebut berjumlah sangat sedikit dan tidak memiliki hasil akhir yang baik. Oleh sebab itu, data-data tersebut akan dihilangkan dari model. Cara lain adalah dengan meminimalsi *weight* dari sebuah node yang tidak memiliki pengaruh cukup besar untuk hasil training. Dalam prakteknya, metode ini dibagi menjadi 3 jenis, yaitu:

3. L1 *Regularization*
4. L2 *Regularization*
5. *Dropout*

Pada penulisan ini, cara yang akan digunakan yaitu *early stopping*. *Early-stopping* dipilih karena metode ini memilih titik tengah antara *underfitting* dan *overfitting*. Sehingga titik tengah ini nantinya akan dapat mempengaruhi *hyper-parameters* untuk algoritma RNN dalam proses *training* data.

2. RMSE

Menurut sebuah artikel yang ditulis oleh Khan, dikatakan bahwa RMSE adalah sebagai berikut

The Root Mean Square Error (RMSE) is calculated using the square root of the residuals. It depicts the absolute fit of the model to the data telling how close the observed data points presently are to the model's predicted values. The R-squared is a relative measure of the fit while RMSE is an absolute measure of the fit. RMSE can also be interpreted as the unexplained variance's standard deviation, and possesses the property of being in the same units as the response variable. The lower values of RMSE depict a better fit. RMSE is a good measure of how accurately the model predicts the response.(Khan dan Noor, 2019)

Dari penjelasan diatas, maka dapat disimpulkan bahwa RMSE adalah salah satu cara untuk menentukan peforma dari sebuah model. RMSE sendiri dihasilkan dari pengurangan nilai sebenarnya dari nilai prediksi yang dikuadratkan lalu dijumlahkan dengan nilai lain dan dibagi dengan jumlah data.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x}_i)^2}{N}}$$

RMSE akan menunjukan berapa besar perbedaan antara hasil prediksi yang dibuat dengan hasil sebenarnya.

2.7 Bahasa Pemograman Python

Milliken dalam bukunya yang berjudul *Python Project for Beginner* mengatakan bahwa *python* merupakan salah satu bahasa yang lebih mudah dipelajari. Lebih mudah dipelajari memiliki arti yaitu membuat orang awam yang belum pernah menyentuh bahasa pemograman apapun dapat menggunakan dan mempelajari bahasa pemograman python. Hal ini dikarenakan bahasa pemograman python memiliki struktur bahasa yang dekat dan hampir sama dengan bahasa Inggris dibandingkan bahasa pemograman lain nya.

Lebih lanjut lagi, Milliken menyatakan bahwa *python* juga salah satu bahasa pemograman yang paling dicari dalam industri teknologi saat ini, digunakan oleh perusahaan seperti *Google, Facebook*, IBM, dll. Itu telah digunakan untuk membangun aplikasi seperti *Instagram, Pinterest, Dropbox*, dan banyak lagi. Dan salah satu alasan mengapa python banyak digunakan adalah kemampuan bahsa pemograman ini untuk membuat berbagai aplikasi mulai dari *front end, back end, data science* dan masih banyak lagi(Milliken, 2020).

Selain itu, menurut Chan, dalam bukunya yang berjudul *Learn Python In One Day And Learn It Well: Python For Beginners With Hands-On Project: The Only Book You Need To Start Coding In Python Immediately* mengatakan bahwa *python* adalah bahasa pemrograman tingkat tinggi yang banyak digunakan yang dibuat oleh Guido van Rossum pada akhir 1980-an. Bahasa ini sangat menekankan pada keterbacaan dan kesederhanaan kode, sehingga memungkinkan pemrogram untuk mengembangkan aplikasi dengan cepat. *Python* dapat digunakan untuk berbagai macam tugas, seperti untuk aplikasi desktop, aplikasi database, pemrograman jaringan, pemrograman game, dan bahkan pengembangan seluler (Chan, 2014).

Dari beberapa pernyataan diatas maka dapat disimpulkan bahwa *python* adalah salah satu bahasa tingkat tinggi yang bisa dioperasikan dalam berbagai platform. Salah satu keunggulan *python* adalah *syntax* yang mudah yang mudah dipahami oleh orang

awam sehingga bahasa pemograman ini cukup mudah untuk dipelajari. Selain itu, bahasa pemograman ini juga bisa membuat berbagai jenis aplikasi *desktop* dan *website*.

2.8 *Django*

Dari *website* resmi *django* mengatakan bahwa *django* adalah *high-level website framework* yang bisa membuat *web development* semakin cepat. Dibuat oleh *developer* berpengalaman, *django* dapat mempersingkat sesuatu hal yang rumit menjadi mudah, sehingga pengguna bisa fokus untuk membangun *web* yang diinginkan. *Django* adalah sebuah *framework* yang lengkap dengan *design principal* yang terstruktur. *Django* juga bisa dibuat menjadi berbagai jenis website, selain itu, *django* sudah memiliki modul keamanan sendiri yang mempermudah pengguna sehingga tidak perlu terlalu memikirkan tentang keamanan.

Django adalah *web framework* yang dibangun menggunakan *python* dimana memungkinkan pengguna untuk menjalankan koding *python* di dalam sebuah *website*. Oleh sebab itu *django* dipilih sebagai *framework* untuk membuat perangkat lunak dalam penulisan ini.

2.9 *Library*

Penelitian ini umumnya menggunakan 3 buah *library* utama. *Library* ini digunakan untuk membantu mulai dari pemrosesan data sampai uji coba hasil. Adapun beberapa *library* yang digunakan pada penelitian ini adalah sebagai berikut:

1. Pandas

Pandas adalah salah satu *library* yang terkenal dikalangan *data scientist*. *Library* ini adalah *library* yang digunakan untuk analisis data, memanipulasi data, dan konversi data. Dalam penelitian ini, *pandas* digunakan untuk membaca dokumen CSV, dan konversi nilai tanggal menjadi *timestamp*.

2. Numpy

Numpy adalah salah satu *library* yang digunakan dalam pemrosesan data terutama data yang berbentuk *array*. Dalam penelitian ini, *numpy* digunakan untuk mengelompokan data ke dalam *array*, konversi tipe data dari *array*, dan mengubah bentuk dari *array*.

3. Statistic

Statistic adalah *library* yang memiliki kegunaan untuk menghitung jumlah *mean, median* dan *modus* dari kumpulan data yang memiliki nilai total yang bertipe angka. Dalam penelitian ini, *statistic* digunakan untuk mencari nilai median dari lokasi yang dikunjungi dalam dataset.

4. Sklearn

Sklearn adalah salah satu *library* yang memiliki peran penting dalam penulisan ini. *Sklearn* sendiri banyak digunakan untuk data analis dan membantu proses pengolahan data untuk *machine learning.* Dalam penulisan ini, *sklearn* digunakan untuk *splitting* data, *minmaxscaler*, menghitung nilai akurasi dan RMSE.

5. Tensorflow

Tensorflow adalah *library* utama yang digunakan dalam penulisan ini. Dalam kegunaannya, banyak orang menggunakan *tensorflow* untuk membuat model di *machine learning*. Dalam penelitian ini, *tensorflow* digunakan untuk pembuatan model, *tokenizer, TimeSeriesGenerator*, *padding* data sekuensial.

2.10 *Google Colab*

Berdasarkan dari website resmi *google colab*, *google colab* adalah produk dari Google *Research* yang memungkinkan siapa saja untuk menulis, dan melakukan eksekusi kode *python* melalui *browser*. Sangat cocok untuk *machine learning* dan data analisis. Secara lebih teknis, *google colab* adalah layanan *jupyter notebook* yang dihosting sehingga tidak perlu persiapan pada lokal sekaligus memberikan submer daya gratis yaitu GPU.

2.11 *Anaconda*

Dari *website* resmi *anaconda* mengatakan bahwa *anaconda* adalah suatu *framework* yang bisa mempermudah pekerjaan *programmer* dalam pembuatan dan pengolahan data. Menurut *website anaconda*, *anaconda* dapat membantu *developer* untuk membuat pemodelan data dan *machine learning* termasuk juga dokumentasi yang lengkap dengan dukungan 20 juta orang diseluruh dunia.

Dari keterangan diatas, maka dapat disimpulkan, *anaconda* merupakan *framework* yang dapat digunakan untuk proses *training*, *modelling* dari data serta untuk membuat *machine learning* sehingga *framework* ini dipilih untuk digunakan dalam penulisan ini. Salah satu *tools* yang digunakan dalam penelitian ini adalah *jupyter notebook*.

Dari *website* resmi *jupyter notebook* mengatakan bahwa *jupyter notebook* adalah aplikasi web *open-source* yang memungkinkan untuk membuat dan berbagi dokumen yang berisi kode langsung, persamaan, visualisasi, dan teks naratif. Penggunaannya mencakup pembersihan dan transformasi data, simulasi numerik, pemodelan statistik, visualisasi data, *machine learning*.

Menurut Milliken, *jupyter notebook* adalah:

It is an open-source integrated development environment (IDE) that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It's a powerful tool that is used in the data science community and makes it easier for us to learn Python because we can solely focus on writing code. (Milliken, 2020:32)

Dari pernyataan di atas, maka dapat disimpulkan penggunaan *jupyter notebook* dalam penulisan ini dikarenakan kemampuan dari *jupyter notebook* yang bisa melakukan visualisi data dan melakukan penggambaran hasil visualisasi serta untuk pembuata *training, modelling* dan *testing* data secara *case by case*.

2.12 Studi Pendahulu

Pada tabel 2.1, disajikan studi pendahulu yang digunakan dalam penulisan ini.

Tabel 2.1
Studi Pendahulu

Judul	Pengarang	Rumusan Masalah	Metode	Hasil
Personalized Next Point-of-Interest	Jing He, Xin Li, Lejian Liao, and William K.Cheung (Member IEEE)	Cara mempelajari distribusi pola untuk memprediksi lokasi pengguna selanjutnya.	Model dipersonalisasikan menggunakan probabilitas per lokasi berdasar distribusi pola pengguna.	*Tensor Based Latent* digunakan untuk mencatat perilaku *checkin* yang secara berurutan dengan menjelajahi tingkat pola persetiap pengguna.

Judul	Pengarang	Rumusan Masalah	Metode	Hasil
An Interactive Multi-Task Learning Framework for Next POI Recommendation with Uncertain Check-ins	Lu Zhang, Zhu sun, Jie Zhang, Yu Lei, Chen Li, Ziqing Wu, Horst Kloeden and Felix Klanner	Cara menentukan POI untuk *check-in* yang tidak pasti.	Menggunakan metode *interactive multi-task learning* untuk mencatat *temporal aware* dan *spatial aware location*. Model dibuat secara sekuensial.	IMTL yang digunakan terbukti memiliki tingkat akurasi lebih tinggi daripada matrix base atau markov chain.
Mining User Check-In Behavior with a Random Walk for Urban Point-Of-Interest Recommendation	Josh Jia-Ching Ying Eric Hsueh-Chan Lu Wen-Ning Kuo Vinvent S. Tseng	Bagaimana memprediksi lokasi selanjutnya untuk pejalan kaki.	Metode yang digunakan adalah *Relevance Learning* dimana metode ini terbati menjadi beberapa proses: *Feature Extraction, User-POI Graph, Dynamic HITS-Based Random Walk*.	Paper ini menyajikan hasil berupa perbandingan beberapa metode dan ditemukan bahwa *User-POI* memiliki fitur yang lebih baik yaitu Prei.
PDPNN: Modeling User Personal Dynamic Preference For Next Point-Of INterest Recommendation	Jihwen Zhing, Can Ma, JIang Zhou, dan Wiping Wang	Cara membuat prediksi lokasi secara *short-term* dan *long-term*	Metode yang digunakan dengan menggunakan algoritma *neural network* . Data dimodelkan menggunakan pemodelan PDPNN dengan metode *inner epoch* sehingga proses *training* data lebih cepat.	Paper ini menyajikan hasil berupa bentuk grafik tingkat kecepatan *training* data dan akurasi, didapatkan bahwa PDPNN adalah pemodelan terbaik untuk *next POI.*
Personalized Recommendation Algorithm Based on Preference Features	Liang Hu, Guohang Song, Zhenzhen Xie, dan Kuo Zhao	Data yang sedikit dan kompleks membuat prediksi lokasi semakin sulit.	*Similiarity* adalah standar untuk menghitung relasi antar sesama pengguna atau sesama item. Metode yang dilakukan adalah dengan menggabungkan kedua fitur item yang nantinya akan dimasukan dalam korelasi *Pearson*.	Metode yang ditemukan akan dibandingkan dengan metorde lama yang digunakan di *MovieLens*. *Movielens* adalah rekomendasi film yang ditemukan pada tahun 1997. Dalam hasilnya, menurut perbandingan, ditemukan hasil yang menyatakan bahwa UPIF (*User Preference-Item Feature*) lebih unggul daripada algoritma BI, IF CF.

BAB III
METODOLOGI PENELITIAN

3.1 Gambaran Umum

Sistem rekomendasi lokasi adalah sistem rekomendasi yang merekomendasikan lokasi kepada pengguna berdasarkan pola pengguna tersebut. Sistem rekomendasi lokasi yang akan dibuat menggunakan algoritma *bidIrectional* LSTM (*Long Short-Term Memory*). *bidirectional* LSTM dipilih karena berdasarkan riset pendahulu, *bidirectional* LSTM adalah salah satu metode dari algoritma sistem rekomendasi (selain *matrix factoirization, tensor factorization, markov chain*) yang bisa untuk memprediksi preferensi lokasi pengguna berdasarkan pola perjalanan.

Dalam pembuatan sistem ini, dibagi menjadi beberapa proses. Beberapa proses tersebut terdiri dari:

1. Pemodelan data

   Data yang digunakan diambil dari kaggle.com. Data yang diambil adalah data pengguna yang melakukan checkpoint di NYC. Data akan dimasukan ke dalam *google colab* dalam bentuk file CSV. Setelah itu, data akan dibersihkan dalam proses *cleaning* dan akan di *splitting* antara data *training* dan data *testing*. Model terbaik yang dihasilkan dari tahap pemodelan data akan digunakan menjadi model pada prediksi lokasi. Hasil dari pengujian model akan disertakan dalam bab 4.

2. Prediksi lokasi

   Proses ini bertujuan untuk memberikan prediksi dari lokasi yang diinput oleh pengguna. Model yang digunakan berasal dari proses pemodelan data. Data yang dimasukan oleh pengguna merupakan data *latitude* dan *longitude* yang nanti akan dikonversi menjadi alamat menggunakan layanan *google map*. Setelah pengguna memasukan lokasi, maka sistem akan mengeluarkan prediksi lokasi selanjutnya. Pengujian fungsi akan disertakan dalam bab 4.

3.2 Spesifikasi Kebutuhan

Spesifikasi kebutuhan dalam penulisan ini dibagi menjadi dua yaitu kebutuhan perangkat keras dan perangkat lunak.

3.2.1 Perangkat Keras

Spesifikasi kebutuhan perangkat keras dapat dilihat pada tabel 3.1:

Tabel 3.1
Spesifikasi kebutuhan perangkat keras

No	Deskripsi Kebutuhan	Keterangan
1	*Processor*	Intel core 5 atau AMD 9 atau MacOS M1
2	Memori	8192 MB DDR 3
3	*Hardisk*	Minimal 1 TB

3.2.2 Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak, dapat dilihat pada tabel 3.2

Tabel 3.2
Spesifikasi kebutuhan perangkat lunak

No	Deskripsi Kebutuhan	Keterangan
1	Sistem operasi	Windows 7/ 10 atau BigSur 14
2	Bahasa pemograman	Python versi 3.8
3	*Editor*	*Jupyter Notebook* / *Google Colab*

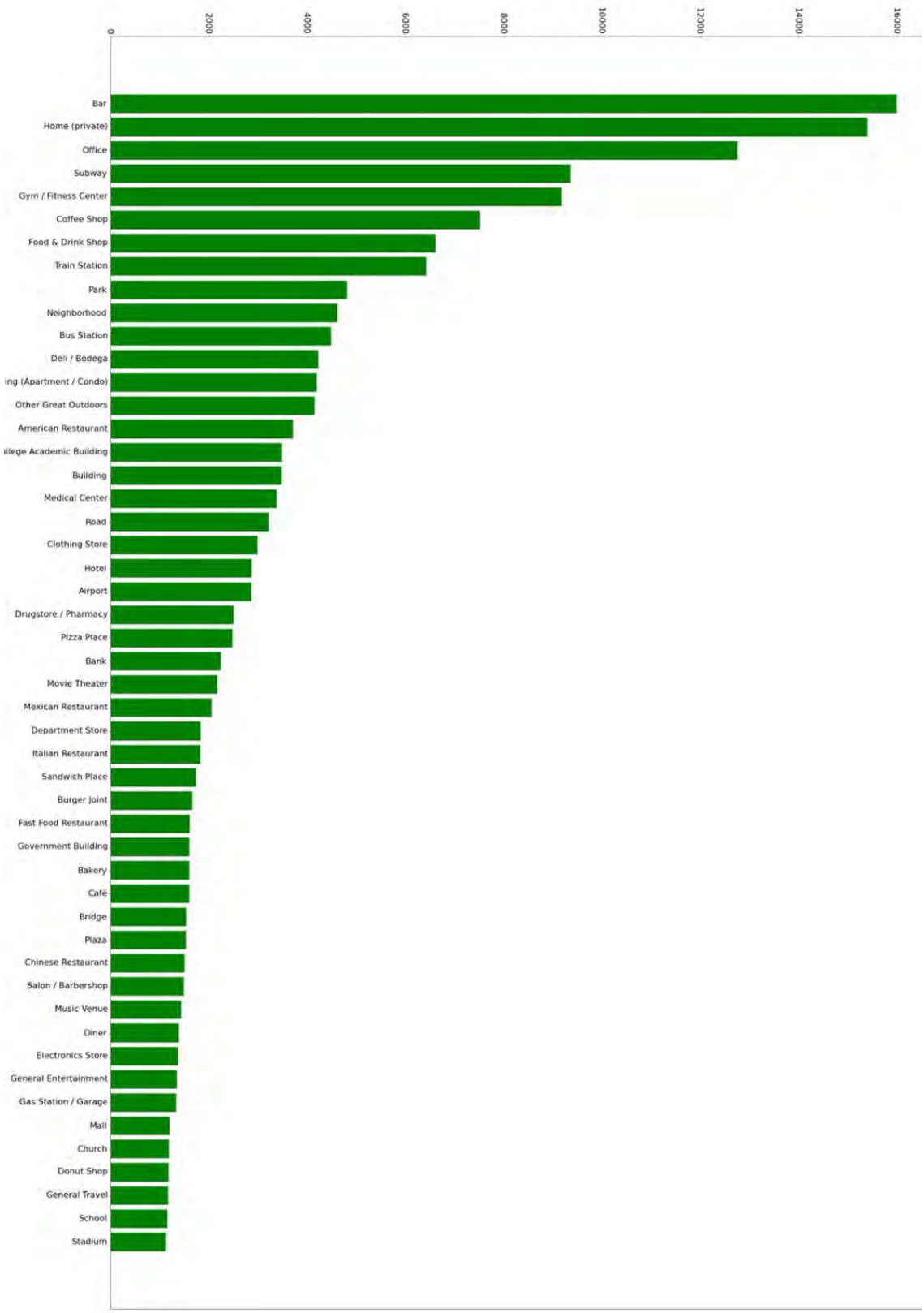
3.3 Visualisasi Data

Penulisan ini menggunakan data yang didapatkan dari *foursquare*. Data *foursquare* ini diambil dari kaggle.com. Dalam kaggle.com disediakan beberapa pilihan data, dalam penulisan ini, data yang dipilh adalah data foursquare dari New York City. Data *foursquare* terdiri dari beberapa *header* yang akan digunakan dalam penulisan ini. *Header* yang dimaksud adalah:

1. *userId*
2. *vanueId*
3. *venueId*
4. *venueCategory*
5. *latitude*
6. *longitude*
7. *utcTimestamp*

Visualisasi data yang dilakukan dari data *foursquare* dibagi dalam 3 kategori:

1. Kategori lokasi dan seberapa sering lokasi tersebut dikunjungi. Visualisasi data dari kategori lokasi ini dapat dilihat pada gambar 3.1



Gambar 3.1
Jumlah kunjungan per lokasi

Data pengguna adalah data yang dikumpulkan dari kota New York selama 2 tahun (2012-2013) dengan data yang berjumlah 227.428 data. Gambar 3.2 menunjukan gambar dari data yang telah diambil.

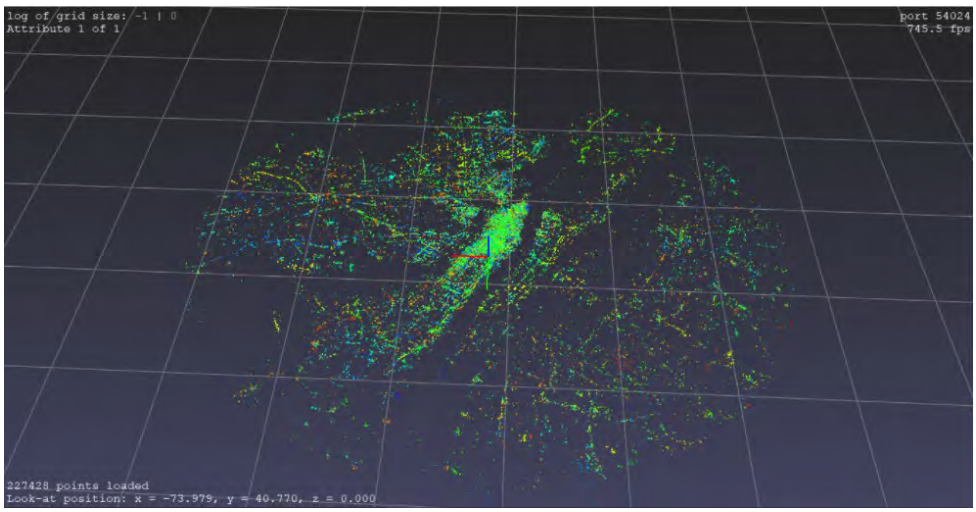
	userId	venueId	venueCategoryId	venueCategory	latitude	longitude	timezoneOffset	utcTimestamp
0	470	49bbd6c0f964a520f4531fe3	4bf58dd8d48988d127951735	Arts & Crafts Store	40.719810	-74.002581	-240	2012-04-03 18:00:09+00:00
1	979	4a43c0aef964a520c6a61fe3	4bf58dd8d48988d1df941735	Bridge	40.606800	-74.044170	-240	2012-04-03 18:00:25+00:00
2	69	4c5cc7b485a1e21e00d35711	4bf58dd8d48988d103941735	Home (private)	40.716162	-73.883070	-240	2012-04-03 18:02:24+00:00
3	395	4bc7086715a7ef3bef9878da	4bf58dd8d48988d104941735	Medical Center	40.745164	-73.982519	-240	2012-04-03 18:02:41+00:00
4	87	4cf2c5321d18a143951b5cec	4bf58dd8d48988d1cb941735	Food Truck	40.740104	-73.989658	-240	2012-04-03 18:03:00+00:00
...	...	...	...	...	...	...	...	...
227423	688	3fd66200f964a52000e71ee3	4bf58dd8d48988d1e7931735	Music Venue	40.733596	-74.003139	-300	2013-02-16 02:29:11+00:00
227424	560	4bca32ff0687ef3be789dbcc	4bf58dd8d48988d16c941735	Burger Joint	40.745719	-73.993720	-300	2013-02-16 02:31:35+00:00
227425	945	50a77716e4b0b5a9492f6f56	4bf58dd8d48988d103941735	Home (private)	40.854365	-73.883070	-300	2013-02-16 02:33:16+00:00
227426	671	4514efe0f964a520e7391fe3	4bf58dd8d48988d11d941735	Bar	40.735981	-74.029309	-300	2013-02-16 02:34:31+00:00
227427	942	4a1e0ca0f964a520bf7b1fe3	4bf58dd8d48988d116941735	Bar	40.726805	-73.957422	-300	2013-02-16 02:35:36+00:00

227428 rows × 8 columns

Gambar 3.2
Sampel Data dari NYC Foursquare

Data dari kaggle ini, diketahui memiliki 8 kolom yang terdiri dari: *userId, vanueId, venueId, venueCategory, latitude, langtitude, timezoneOffset, utcTimeStamp*. Penelitian ini menggunakan nilai dari kolom *userId, venueId, latitude, longtitude,* dan *utcTimeStamp*.

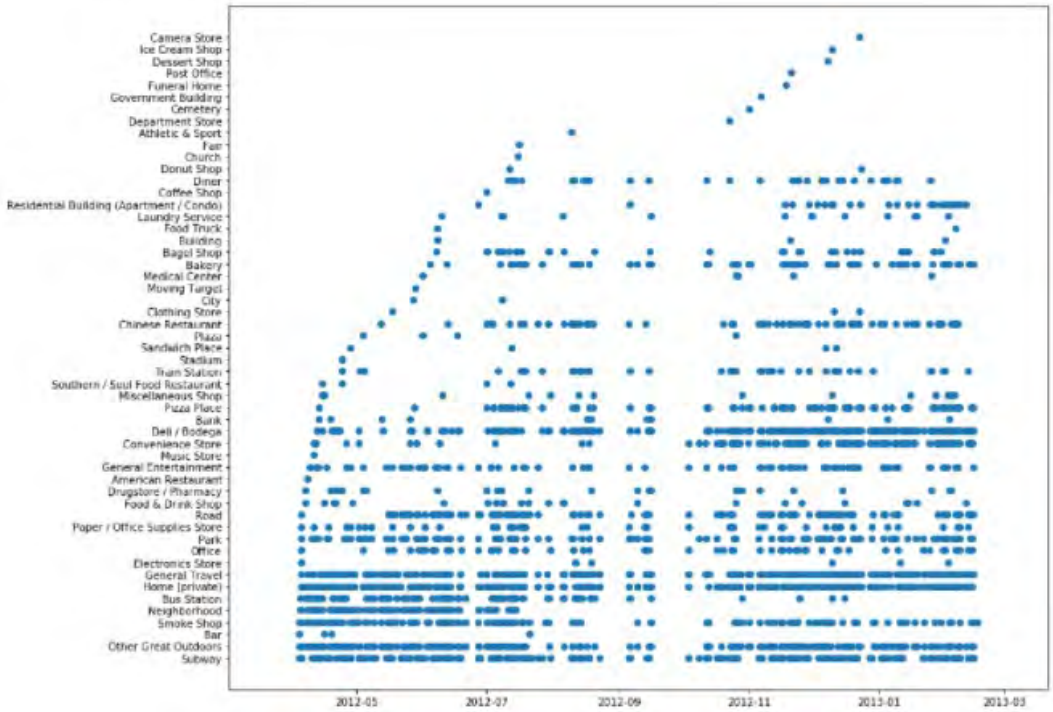
Setelah data selesai di akuisisi, dimulai tahap visualisasi data. Visualisasi data ini berguna untuk mempelajari data yang akan digunakan dalam sistem sehingga beberapa nilai yang tidak valid bisa diketahui. Selain untuk menghilangkan nilai tidak valid, visualisasi data juga dapat membantu untuk mengetahui *sparsity* dari data tersebut. Gambar 3.3 menunjukan *mapping* dari setiap lokasi lat ln yang tersedia dalam data.



Gambar 3.3
Mapping Data dari NYC Foursquare

Pada gambar 3.3 terlihat banyak lokasi yang berjauhan dan tidak tertumpu pada satu area sehingga dapat disimpulkan bahwa data yang tersedia memiliki cangkupan wilayah yang sangat luas. Hal ini memungkinkan untuk terjadinya pola data *trajectory* yang hilang atau pola data *trajectory* yang unik. Hal ini dikarenakan kemungkinan adanya lokasi yang berjauhan dalam satu *trajectory* yang mungkin tidak dikunjungi oleh pengguna lain. Maka dari gambar 3.3 didapatkan informasi bahwa ada kategori lokasi yang harus dihilangkan.

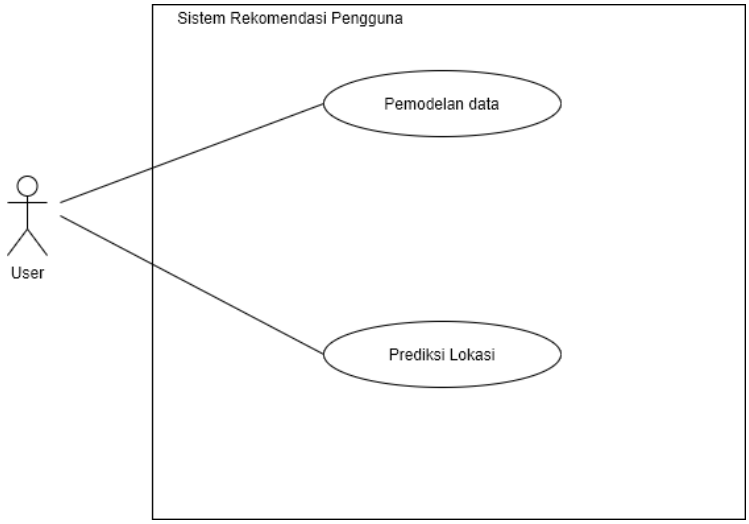
Untuk mengetahui jumlah kunjungan per *venueCategory*, maka dilakukan mapping yang dapat dilihat pada gambar 3.4.



Gambar 3.4
Mapping Data dari NYC Foursquare

3.4 Diagram *Usecase*

Diagram usecase adalah diagram yang menggambarkan interaksi pengguna dengan sistem. *Usecase* sistem ini dapat dilihat pada gambar 3.5,



Gambar 3.5
Usecase sistem rekomendasi lokasi pengguna

Usecase memiliki beberapa scenario yang akan dijelaskan pada scenario *usecase.* Skenario *usecase* adalah keterangan dari deskripsi alur kerja yang dikembangan menggunakan *usecase* secara bertahap dari sistem yang sedang dikembangkan. Dibawah ini dilampirkan beberapa skenario dari *usecase* sistem yang sedang dikembangkan.

3.4.1 *Usecase* Skenario Pemodelan Data

Tabel 3.3 dan 3.4 akan menunjukan skenario utama dan alternatif dari case pemodelan data berdasarkan sistem yang sedang dikembangkan.

Nama *Usecase* : Pemodelan Data
Aktor : User
Pre-condition : File CSV sudah disiapkan

Tabel 3.3
Skenario utama pemodelan data

Aksi Aktor:	Reaksi Sistem
Skenario utama: Pemodelan Data	
1. Upload file CSV *kaggle*	
	2. Membaca file yang diupload
	3. Pengelompokan data berdasarkan id pengguna dan tanggal
	4. Memberi label *false* untuk lokasi yang hanya muncul 1x dalam data
	5. Mengelompokan lokasi setiap 4 jam menjadi array *trajectories*
	6. Membuang *trajectories* yang memiliki jumlah elemen dibawah 3
	7. Melakukan augmentasi data, dan memisahkan antara *trajectories* dan *destination*

Aksi Aktor:	Reaksi Sistem
Skenario utama: Pemodelan Data	
	8. Memulai data training dengan menggunakan learning rate dan epoch.
	9. Data training berhasil, menampilan tampilan berhasil beserta akurasi dari data training
	10. Menghitung RMSE
	11. Simpan Model, dan nilai RMSE

Tabel 3.4
Skenario alternatif pemodelan data

Aksi Aktor:	Reaksi Sistem
Skenario alternatif: Terjadi error file CSV	
1. Upload file CSV *kaggle*	
	2. Membaca file yang diupload
	3. File error. Menampilkan pesan error

3.4.2 *Usecase* Skenario Prediksi Lokasi

Tabel 3.5, 3.6, dan 3.7 akan menunjukan skenario dari case prediksi lokasi berdasarkan sistem yang sedang dikembangkan.

Nama *Usecase* : Prediksi Lokasi
Aktor : User
Pre-condition : Model yang dihasilkan dianggap sudah memiliki nilai optimum,

Tabel 3.5
Skenario utama Prediksi Lokasi

Aksi Aktor:	Reaksi Sistem
Skenario utama: Prediksi Lokasi dengan 4 input lokasi	
1. Membuka halaman prediksi lokasi	
	2. Menampilkan tampilan untuk memasukan *lat long* dari 4 lokasi terakhir
3. Memasukan nilai *lat long*	
4. Menekan tombol "*Get Recommendation*"	
	5. Validasi model
	6. Load data
	7. Validasi data
	8. Validasi data berhasil
	9. Memulai data testing
	10. Menampilkan output berupa 4 lokasi terakhir dan lokasi rekomendasi.

Tabel 3.6
Skenario alternatif nilai lokasi yang dimasukan di bawah 4

Aksi Aktor:	Reaksi Sistem
Skenario Alternatif: Prediksi Lokasi dengan input dibawah 4 lokasi	
1. Membuka halaman prediksi lokasi	
	2. Menampilkan tampilan untuk memasukan *lat long* dari 4 lokasi terakhir
3. Memasukan nilai *lat long*	
4. Menekan tombol "*Get Recommendation*"	
	5. Validasi model
	6. Load data
	7. Validasi data
	8. Validasi data tidak berhasil karena jumlah nilai *lat long* dibawah 4
	9. Menampilkan pesan *error*, dan kembali ke halaman prediksi data

Tabel 3.7
Skenario alternatif model *error*

Aksi Aktor:	Reaksi Sistem
Skenario alternatif model error	
1. Membuka halaman prediksi lokasi	
	2. Menampilkan tampilan untuk memasukan *lat long* dari 4 lokasi terakhir
3. Memasukan nilai *lat long*	
4. Menekan tombol "*Get Recommendation*"	
	5. Validasi model
	6. Validasi model tidak berhasil, menampilkan halaman utama dan pesan error

3.5 Diagram Aktivitas

Diagram *activity* adalah diagram yang menggambarkan aktivitas ke aktivitas dari alur yang terjadi dalam suatu sistem. Diagram *activity* akan menjelaskan tentang bagaimana alur sistem ini berjalan.

3.5.1 Diagram Aktivitas Pemodelan Data

Gambar 3.6 akan menunjukan tentang proses pemodelan data. Dalam proses nya, pemodelan data dibagi menjadi beberapa bagian yaitu:

1. Sistem memvalidasi data yang digunakan

2. Jika data valid, maka akan dimulai proses data *cleaning*. Dalan data *cleaning,* akan dibuang semua data lokasi yang hanya dikunjungi sekali dari jumlah keseluruhan data. Hal ini dilakukan untuk mengurangi data sparsity.
3. Data akan dikelompokan berdasar id pengguna dandiurutkan nilai timestamp. Setelah diurutkan berdasarkan *timestamp*, data akan dimasukan ke dalam *array trajectories* untuk setiap interval 4 jam. Setelah dimasukan ke dalam array *trajectories*, untuk setiap *array trajectories* yang memiliki elemen di bawah 3 akan dibuang.
4. Setelah *array trajectory* sudah bersih dilanjutkan dengan *tokenizer*. *Tokenizer* berfungsi untuk melabeli setiap elemen yang terdiri dari tipe data *string* menjadi tipe data *float32*.
5. Selanjutnya akan dilakukan proses *augmentasi*. Proses ini dilakukan untuk meningkatkan nilai akurasi. Data augmentasi dalam penulisan ini dibagi menjadi 2 jenis. Yang pertama, setiap *array trajectories* yang memiliki panjang lebih dari 4, maka akan dibuat array tambahan baru dengan pola sebagai berikut:

   array_traj = [loc_a, loc_b, loc_c, loc_d, loc_e]

   extra_array_traj_1 = [loc_a, loc_b, loc_c, loc_d]

   extra_array_traj_2 = [loc_b, loc_c, loc_d, loc_e]

   extra_array_traj_3 = [loc_c, loc_d, loc_e]

   Selanjutnya, untuk setiap array yang lebih dari 5 akan dibagi ke dalam beberapa array dengan pola seperti berikut:

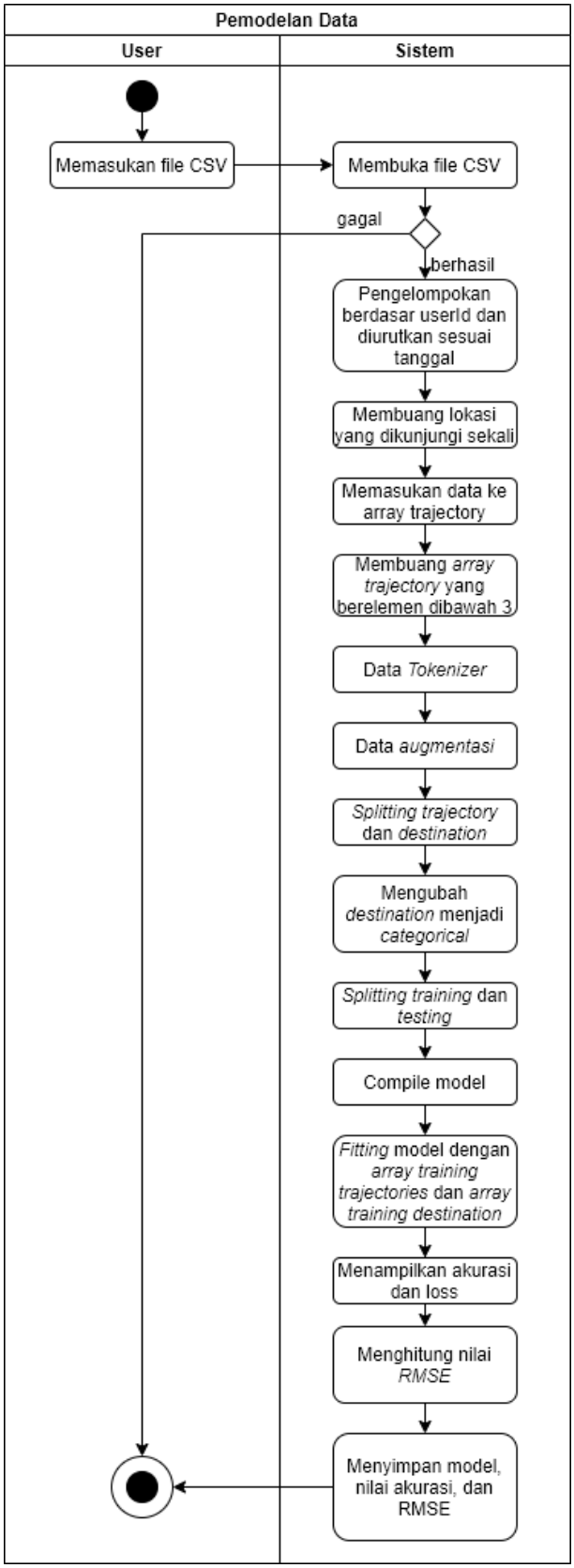
   array_traj = [loc_a, loc_b, loc_c, loc_d, loc_e]

   new_array_traj_1 = [loc_a, loc_b, loc_c, loc_d]

   new_array_traj_2 = [loc_b, loc_c, loc_d, loc_e]

   Pembagian pertama adalah pembagian yang digunakan untuk memecah *array trajectory* yang memiliki panjang lebih dari 4 dan ditambahkan ke *array* utama. Pembagian kedua adalah pembagian yang digunakan untuk memecah *array* berdasarkan urutan pola.

6. Setelah *augmentasi*, dilanjutkan dengan *split array trajectory* dengan *destination*. Nilai *destination* adalah nilai terakhir dari setiap *array trajectory*.
7. Setelah *split array trajectory*, maka dilajutkan dengan *padding*. *Padding* berfungsi untuk menyamakan panjang dari *array trajectory*. Disini, setiap *array trajectory* yang memiliki elemen kurang dari 4, maka akan ditambahkan nilai 0 disetiap depan *array*.
8. *Array destination* dijadikan data *categorical*.
9. Setelah proses *split array,* maka dilanjutkan dengan *splitting* data untuk data *training* dan *testing* dengan menggunakan rasio 80% training, 20% testing.
10. Setelah selesai, *array trajectory* untuk *training* dan *array destination training* dimasukan ke dalam model untuk di training.
11. Setelah *training* selesai, akan dilakukan perhitungan untuk RMSE dan akurasi
12. Setelah perhitungan selesai, maka hasil perhitungan dan model akan disimpan.

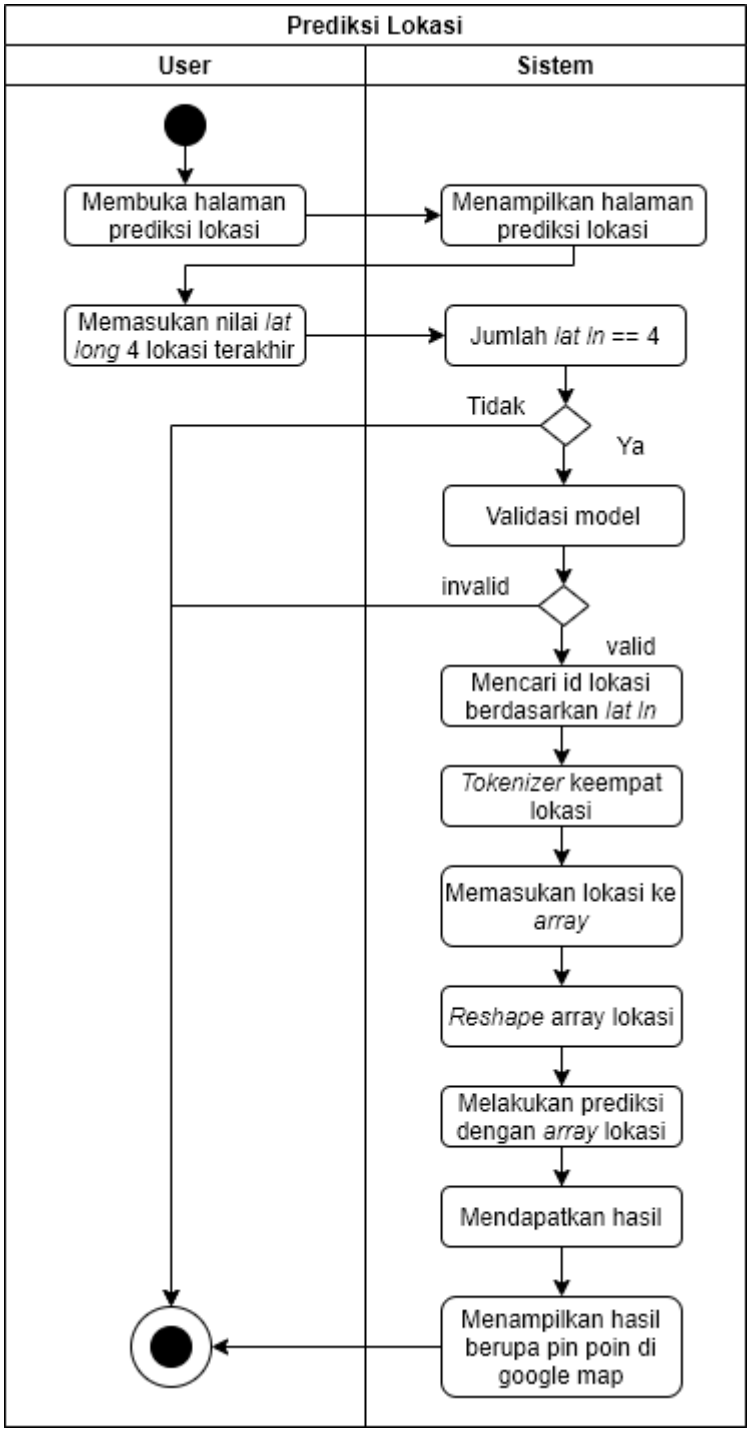


Gambar 3.6
Diagram Aktivitas Pemodelan Data

3.5.2 Diagram Aktivitas Prediksi Lokasi

Gambar 3.7 akan menunjukan tentang proses prediksi lokasi*.* Setelah hasil dari pemodelan data dinggap optimum, maka akan dilanjutkan dengan prediksi lokasi yang akan dibagi dalam beberapa proses yaitu;

2. Pengguna memilih nilai *lat ln* dari *dropdown* yang disediakan, lalu menekan tombol *submit*.
3. Setelah nilai *lat ln* disubmit, sistem akan memvalidasi apakah nilai yang disubmit berjumlah lebih dari 4. Jika tidak, maka proses akan selesai dan kembali ke halaman *testing*.
4. Jika data yang disubmit berjumlah lebih dari 4, maka akan dilanjutkan dengan pencarian id lokasi berdasarkan nilai *lat ln* yang dimasukan.
5. Setelah lokasi ketemu, maka akan dilanjutkan dengan *tokenizer* untuk mengubah id lokasi menjadi *float32.*
6. Setelah selesai, akan dilanjutkan dengan proses *reshape array*.
7. Setelah *array* berhasil di *reshape*, maka akan dilanjutkan dengan proses prediksi. Array yang tersedia akan di prediksi lokasi selanjutnya menggunakan model yang disediakan.
8. Hasil prediksi akan ditampilkan dalam bentuk pin poin di google maps.



Gambar 3.7
Diagram Aktivitas Prediksi Lokasi

3.6 Diagram Kelas

Gambar 3.8 akan menunjukan kelas-kelas, metode, serta operator yang ada dalam perancangan sistem rekomendasi yang akan dibuat. Sistem memiliki beberapa kelas seperti berikut:

1. Pembuatan Model

   Kelas ini merupakan main *class* dari data *training*. Kelas ini memiliki satu input yaitu file CSV dan satu *method* yaitu dataPreprocessing.

2. Data Training

   Kelas ini berfungsi untuk melakukan proses training. Kelas ini terdiri dari beberapa *method* yaitu:

   1. dataPreprocessing: Berfungsi untuk melakukan proses data *cleaning*. Data *cleaning* akan menghilangkan nilai lokasi yang hanya dikunjungi sekali dari total keseluruhan data, Mengurutkan data sesuai userId dan utcTimeStamp. Lalu memasukan data tersebut dalam array, dilanjutkan dengan proses *augmentasi* data dan diakhiri dengan memisahkan antara *array trajectories* dan *array destination*.
   2. dataSplitting: Berfungsi untuk memisahkan antara data *training* dan data *testing*.
   3. dataTrainingLSTM: Berfungsi untuk melakukan data *training* yang telah di split dari *method* dataSplitting.
   4. saveResult: Berfungsi untuk menyimpan hasil berupa model, akurasi, presisi, recall, lalu hasil *tokenizer* lokasi dan data lokasi (*venueId, lat,ln*).

3. Model

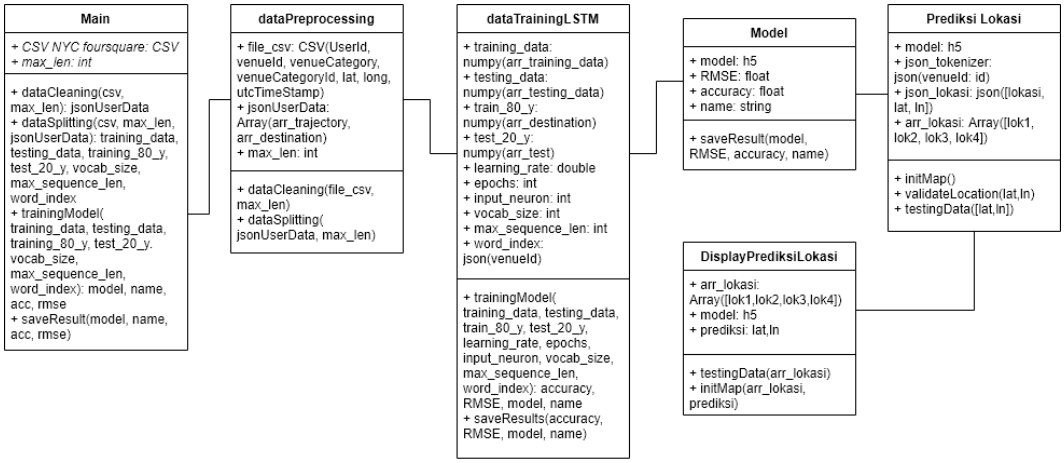
   Kelas ini berfungsi untuk melakukan menyimpan model yang akan digunakan dalam prediksi lokasi.

4. Prediksi Lokasi

   Kelas ini berfungsi untuk menampilkan halaman input *user* berupa *dropdown* yang berisikan *list* lokasi. Pengguna akan memilih lokasi di halaman ini sebelum akhirnya di prediksi menggunakan model yang sudah ada.

5. Display Prediksi Lokasi

   Kelas ini berfungsi untuk menampilkan hasil dari prediksi yang didapatkan dari *input* pengguna. Sebelum ditampilkan, *input* pengguna akan di prediksi menggunakan model yang ada. Sebelum di prediksi, input pengguna akan dijadikan *numpy array* dengan *shape* (1,4,1) untuk dimasukan ke dalam model dan diprediksi. Setelah hasil prediksi keluar, akan ditampilkan dalam *google maps*.



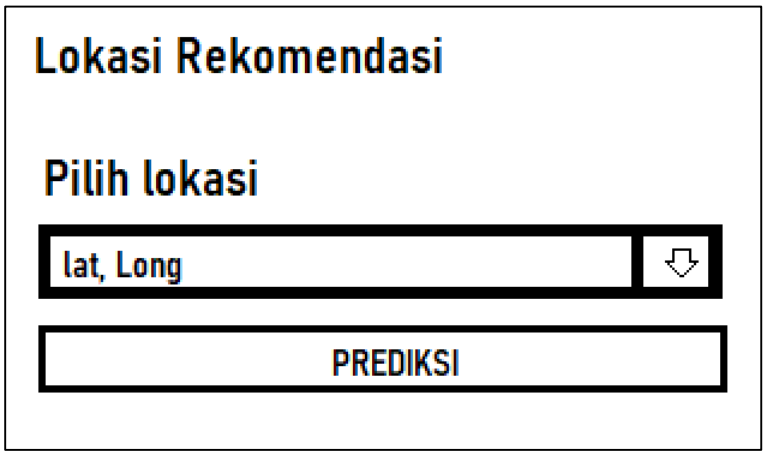
Gambar 3.8
Diagram Kelas Sistem Rekomendasi

3.7 Rancangan Antar Muka

Rancangan antar muka ini dibuat untuk menggambarkan secara umum gambaran dari sistem rekomendasi lokasi yang akan disusun dalam penulisan ini. Berikut rancangan antar muka dari sistem yang akan dibuat.

3.7.1 Tampilan Halaman *Display Testing*

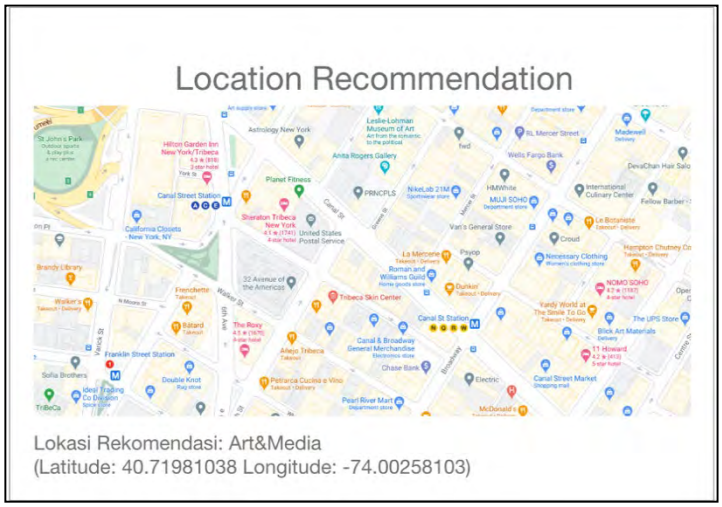
Gambar 3.6 menunjukan tampilan data *testing*. Tampilan ini berfungsi untuk mendapatkan masukan dari pengguna berupa data baru yang berupa 4 lokasi yang dipilih dari nilai *lat long*. Setelah diinput, maka pengguna bisa menekan tombol submit untuk mendapat prediksi.



Gambar 3.9
Tampilan Display Data Testing

3.7.2 Tampilan Data *Testing* Selesai

Gambar 3.7 menunjukan tampilan data *testing* selesai. Jika testing data sudah selesai, maka akan muncul rekomendasi selanjutnya. Selain menampilkan lokasi rekomendasi selanjutnya, sistem juga akan menampilkan lokasi di dalam map berdasar hasil *latitude* dan *longitude* dari hasil rekomendasi.



Gambar 3.10
Tampilan Data Testing Selesai

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Spesifikasi dan Implementasi Pengujian

Pengujian dalam penulisan ini dibagi menjadi dua pengujian yaitu pengujian perangkat lunak dan pengujian model. Pengujian perangkat lunak adalah pengujian yang digunakan untuk menguji perangkat lunak yang dibuat. Pengujian model adalah pengujian yang digunakan untuk menguji model dari hasil *training* dan di validasi menggunakan data *testing*.

4.1.1 Spesifikasi dan Implementasi Pengujian Perangkat Lunak

Pengujian perangkat lunak yang dibuat akan diuji menggunakan laptop dengan spesifikasi yang dapat dilihat pada tabel 4.1

Tabel 4.1
Spesifikasi Untuk Pengujian Perangkat Lunak

NO	Konfigurasi Perangkat	Spesifikasi
1	Prosesor	M1
2	RAM	8 GB
3	Kapasitas Penyimpanan	512 GB

4.1.2 Spesifikasi dan Implementasi Pengujian Model *Bi-Directional LSTM*

Pengujian model *bidirectional LSTM* diuji menggunakan *google colab* dengan spesifikasi yang dapat dilihat pada tabel 4.2

Tabel 4.2
Spesifikasi untuk Pengujian Model

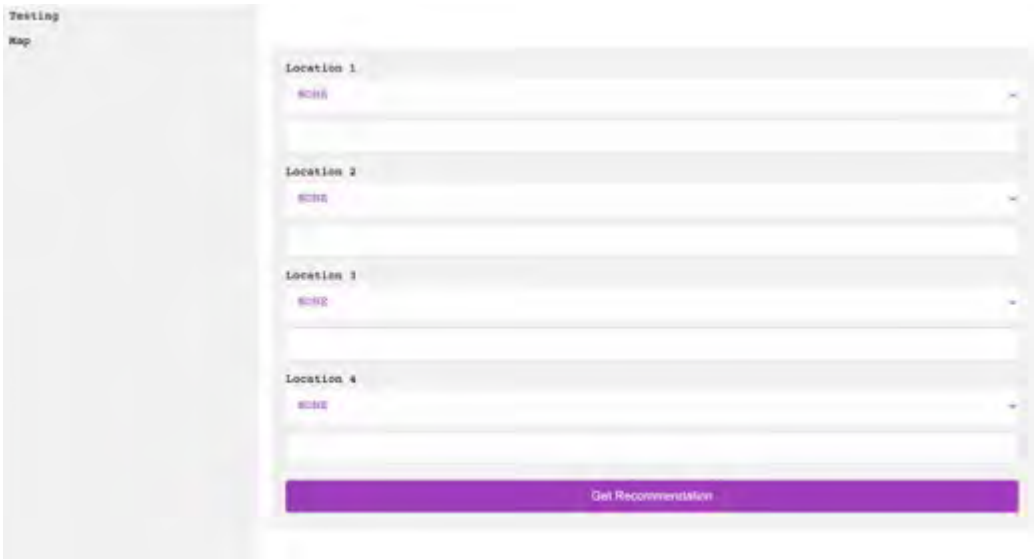
NO	Konfigurasi Perangkat	Spesifikasi
1	RAM	12 GB
2	Lingkungan Runtime	Python 3
3	Prosesor	GPU

4.2. Tampilan Antar Muka

Tampilan antar muka yang dtampilkan pada sistem yang dibuat dalam penulisan ini meliputi:tampilan halaman testing dan halaman hasil. Seluruh tampilan yang disebutkan akan dijelaskan lebih rinci pada sub bab dibawah.

4.2.1 Tampilan Display Testing

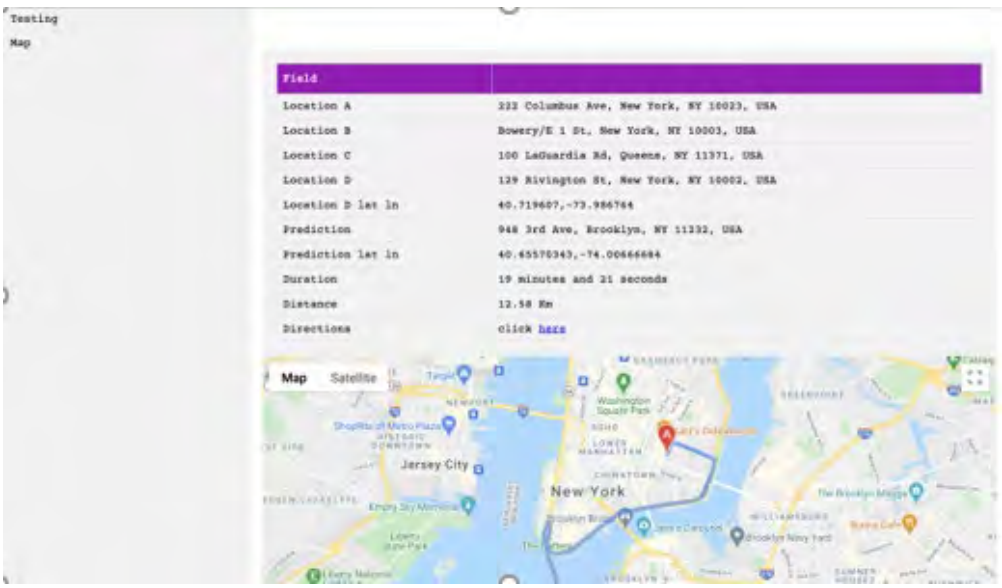
Gambar 4.1 adalah gambar dari halaman *testing*. Pada halaman *testing*, pengguna akan memilih 4 lokasi terakhir dari *dropdown* yang disediakan. Setelah memilih, akan muncul pin point pada *maps* yang akan ditampilkan.



Gambar 4.1
Tampilan Halaman Testing

4.2.2 Tampilan Data Testing Selesai

Gambar 4.2 adalah gambar dari halaman output ketika hasil prediksi sudah keluar. Pada halaman ini ditampilkan nama dari pin poin *lat ln* untuk setiap lokasi, lalu hasil prediksi dan juga akan digambarkan pada *google maps*.



Gambar 4.2
Tampilan Halaman Rekomedasi

4.3 Hasil Pengujian

Hasil pengujian adalah hasil dari pengujian yang dilakukan dalam penulisan ini. Hasil dari pengujian akan dibagi menjadi 2 yaitu pengujian fungsi dan pengujian model. Pada pengujian fungsi akan dijelaskan fungsi apa saja yang ada di dalam sistem yang dibuat pada penulisan ini. Pengujian model berisi tentang proses dan hasil dari pembuatan model mulai dari akuisisi data sampai menjadi model yang digunakan untuk testing pada sistem yang dibuat.

4.3.1 Pengujian Model

Pengujian model adalah pengujian yang dilakukan untuk menguji nilai akurasi, dan RMSE dari model yang dibuat. Pengujian ini menggunakan *bidirectional* LSTM, jumlah epoch 1000 epoch, dan learning rate yang digunakan adalah Adam. Model yang digunakan akan menggunakan beberapa layer yaitu: embedding layer, *bidirectional* LSTM *layer* (jumlah neuron 64 dan recurrent dropout 0.2), *dropout layer*, *dense layer* (jumlah neuron sesuai dengan total lokasi yaitu 12915), dan terkahir *Activation* layer dengan menggunakan softmax sebagai *activation function*. Model juga menggunakan *categorical crossentropy* sebagai *loss function*.

Pengujian ini dilakukan dengan mengubah jumlah dari layer pada embedding layer dengan 50, 100, 150 yang dapat dilihat pada tabel 4.3, 4.4, dan 4.5.

Tabel 4.3
Tabel 50 *hidden layer*

Layer	**Jumlah Neuron**
Input Layer	50
Bidirectional_LSTM	128
Dropout	128
Output Layer	12,915

Tabel 4.3 menunjukan *summary* dari model yang menggunakan 50 *hidden layer* pada *input layer*. Jumlah LSTM *layer* 64 dan karena menggunakan *bidirectional layer* maka jumlah LSTM *layer* menjadi 128.

Tabel 4.4
Tabel 100 *hidden layer*

Layer	**Jumlah Neuron**
Input Layer	100
Bidirectional_LSTM	128
Dropout	128
Output Layer	12,915

Tabel 4.4 menunjukan *summary* dari model yang menggunakan 100 *hidden layer* pada *input layer.* Jumlah LSTM *layer* 64 dan karena menggunakan *bidirectional layer* maka jumlah LSTM *layer* menjadi 128.

Tabel 4.5
Tabel 150 *hidden layer*

Layer	**Jumlah Neuron**
Input Layer	150
Bidirectional_LSTM	128
Dropout	128
Output Layer	12,915

Tabel 4.5 menunjukan *summary* dari model yang menggunakan 150 *hidden layer* pada *input layer.* Jumlah LSTM *layer* 64 dan karena menggunakan *bidirectional layer* maka jumlah LSTM *layer* menjadi 128.

Penelitian ini juga akan menggunakan 3 *learning rate* yang berbeda, yaitu: Adam (0.1), Adam (0.01) dan Adam (0.001) yang dapat dilihat pada gambar 4.6, 4.7 dan 4.8.

```
opt = Adam(learning_rate=0.1)
model3.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

Gambar 4.3
Model dengan 0.1 *learning rate*

```
opt = Adam(learning_rate=0.01)
model3.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

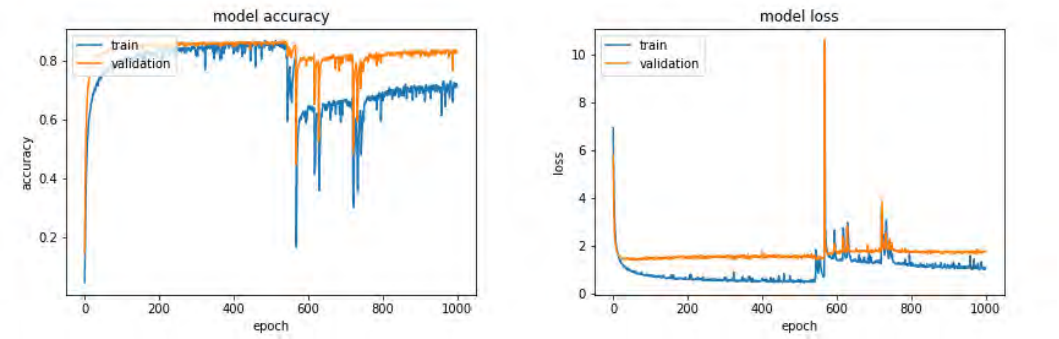
Gambar 4.4
Model dengan 0.01 *learning rate*

```
opt = Adam(learning_rate=0.001)
model3.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

Gambar 4.5
Model dengan 0.001 *learning rate*

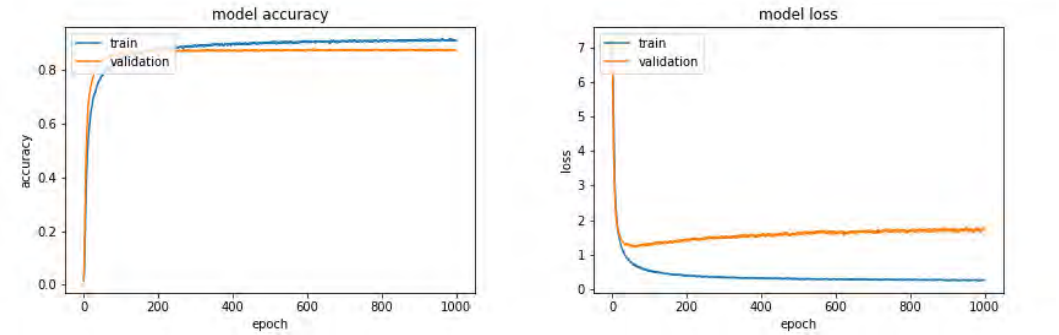
4.3.2.1 Hasil Pengujian

Hasil pengujian akan dibagi ke dalam 3 bagain besar yaitu berdasarkan nilai dari *hidden layer*. Pengujian pertama diambil dari *hidden layer* yang berjumlah 50. Hasil pengujian untuk 50 *hidden layer* dapat dilihat pada gambar 4.9, 4.10, 4.11.



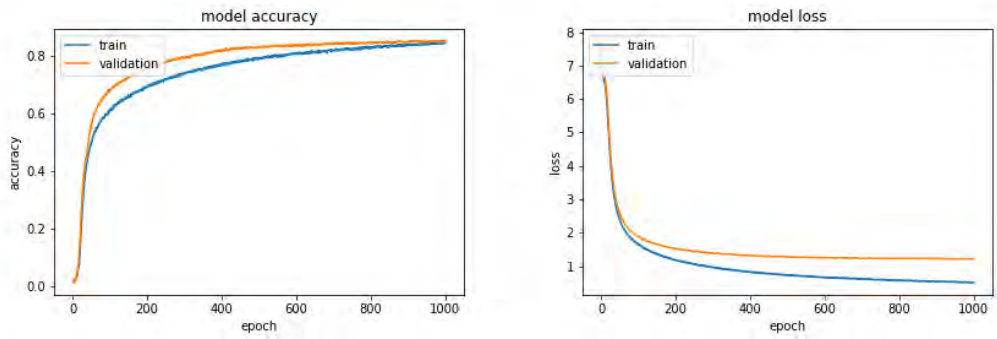
Gambar 4.6
Hasil 50 *hidden layer* dengan 0.1 *learning rate*

Gambar 4.6 menunjukan hasil dari penggunaan model dengan 50 *hidden layer* dan 0.1 *learning rate*. Dapat dilihat pada gambar, gambar tersebut menunjukan adanya fluktuasi disetiap *epoch*. Fluktuasi ini terjadi karena kecilnya jumlah *learning rate* sehingga membuat model dengan cepat mencapai titik sub optimum. Model ini menghasilkan akurasi sebesar 90.8%.



Gambar 4.7
Hasil 50 *hidden layer* dengan 0.01 *learning rate*

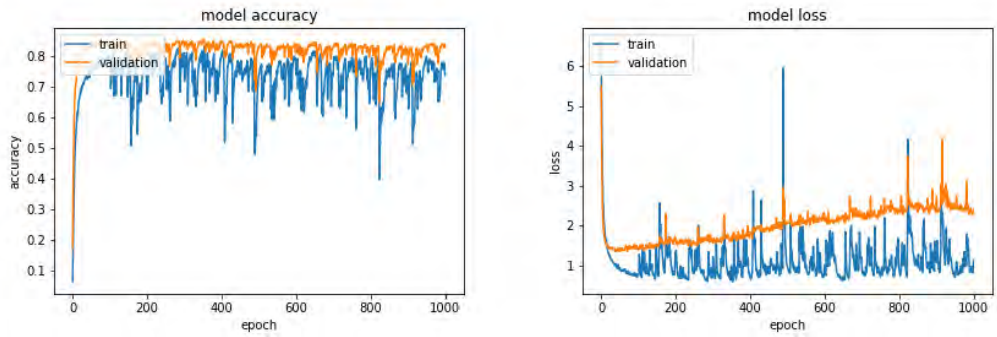
Gambar 4.7 menunjukan hasil dari penggunaan model dengan 50 *hidden layer* dan 0.01 *learning rate*. Dapat dilihat pada gambar, pada akurasi, nilai akurasi yang awalnya tertinggal, dapat bertemu di suatu titik dengan nilai validasi. Sementara pada model *loss*, terjadi peningkatan pada *validation loss*. Hal seperti ini dapat terjadi karena akurasi pada validasi tidak mengalami kenaikan dan membentuk seperti kurva kecil dengan banyak nilai fluktuasi. Hal ini dikarenakan model sudah mencapai titik optimum sebelum epoch berakhir. Model ini memberikan hasil akurasi sebesar 93.4%.



Gambar 4.8
Hasil 50 *hidden layer* dengan 0.001 *learning rate*

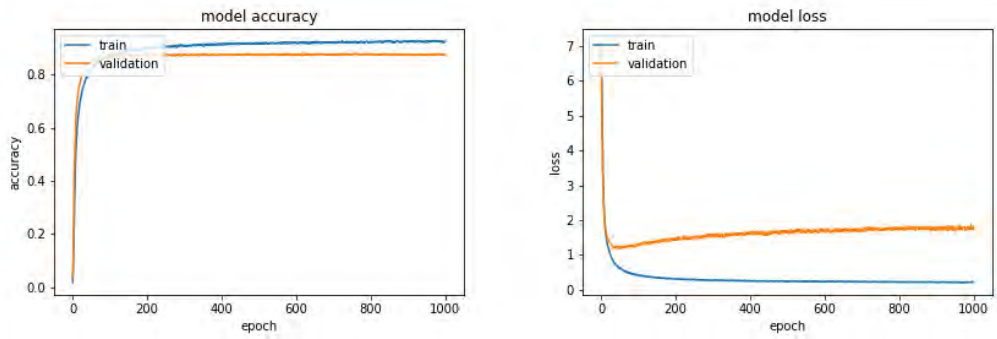
Gambar 4.8 menunjukan hasil dari penggunaan model dengan 50 *hidden layer* dan 0.001 *learning rate*. Dapat dilihat pada gambar, pada akurasi, nilai akurasi yang awalnya tertinggal jauh, akhirnya dapat bertemu di suatu titik dengan nilai validasi. Sementara pada model *loss*, terjadi penurunan antara *training loss* dan *validation loss*. Dengan mengunakan nilai *learning rate* yang makin kecil, maka model akan semakin lama untuk menuju ke nilai sub optimum. Model ini memberikan hasil akurasi sebesar 92.2%.

Pengujian kedua akan menggunakan nilai hidden layer berjumlah 100. Hasil dapat dilihat pada gambar 4.12, 4.13, 4.14



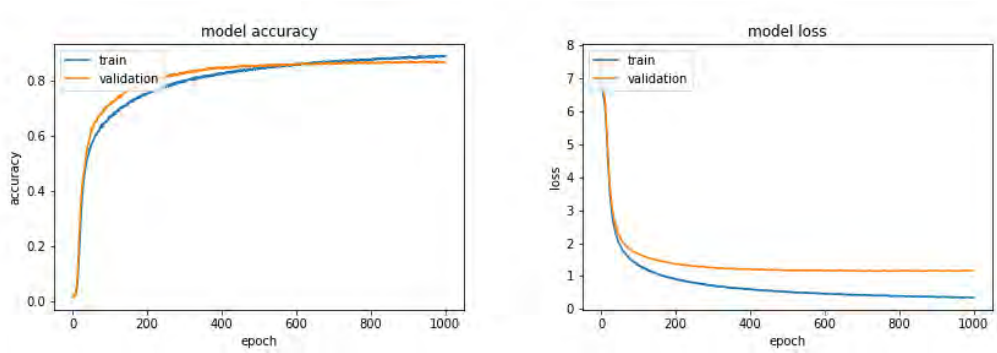
Gambar 4.9
Hasil 100 *hidden layer* dengan 0.1 *learning rate*

Gambar 4.9 menunjukan hasil dari penggunaan model dengan 100 *hidden layer* dan 0.1 *learning rate*. Dapat dilihat pada gambar, gambar tersebut menunjukan adanya fluktuasi disetiap *epoch*. Fluktuasi ini terjadi karena kecilnya jumlah *learning rate* sehingga membuat model dengan cepat mencapai titik sub optimum. Model ini menghasilkan akurasi sebesar 90.2%.



Gambar 4.10
Hasil 100 *hidden layer* dengan 0.01 *learning rate*

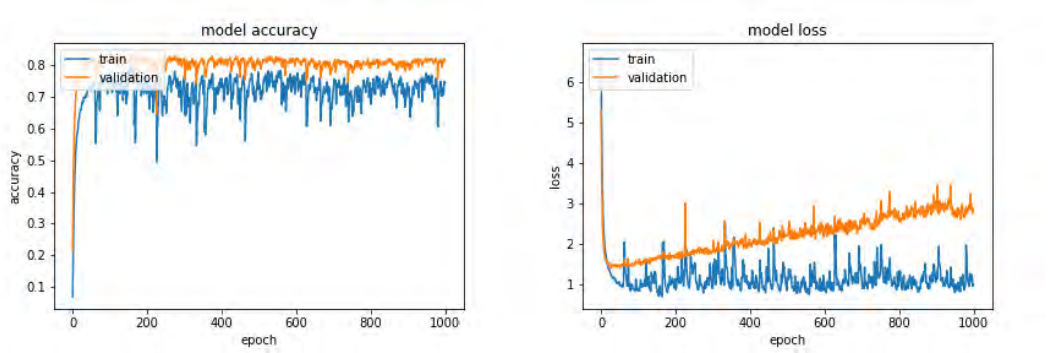
Gambar 4.10 menunjukan hasil dari penggunaan model dengan 100 *hidden layer* dan 0.01 *learning rate*. Dapat dilihat pada gambar, pada akurasi, nilai akurasi yang awalnya tertinggal, dapat bertemu di suatu titik dengan nilai validasi. Sementara pada model *loss*, terjadi peningkatan pada *validation loss*. Hal seperti ini dapat terjadi karena akurasi pada validasi tidak mengalami kenaikan dan membentuk seperti kurva kecil dengan banyak nilai fluktuasi. Model ini memberikan hasil akurasi sebesar 93.5%.



Gambar 4.11
Hasil 100 *hidden layer* dengan 0.001 *learning rate*

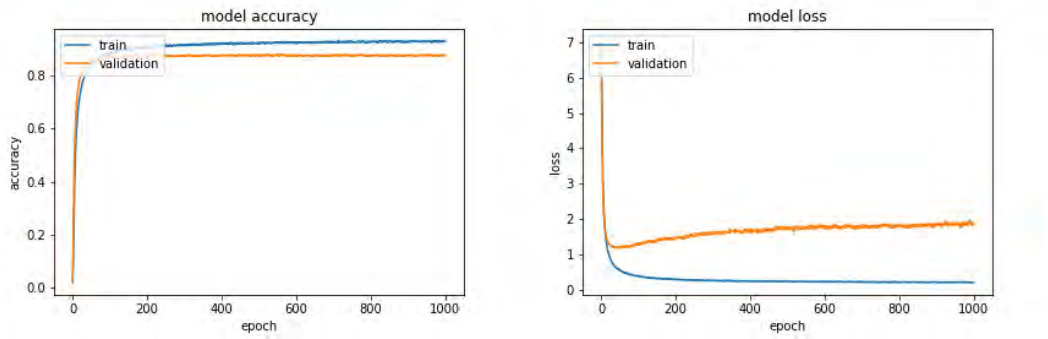
Gambar 4.11 menunjukan hasil dari penggunaan model dengan 100 *hidden layer* dan 0.001 *learning rate*. Dapat dilihat pada gambar, pada akurasi, nilai akurasi yang awalnya tertinggal jauh, akhirnya dapat bertemu di suatu titik dengan nilai validasi dan akhirnya menjadi stabil. Sementara pada model *loss*, terjadi penurunan antara *training loss* dan *validation loss* tetapi menjadi stabil mendekati *epoch* akhir. Dengan mengunakan nilai *learning rate* yang makin kecil, maka model akan semakin lama untuk menuju ke nilai sub optimum. Model ini memberikan hasil akurasi sebesar 93.3%.

Dan pengujian ketiga menggunakan *hidden layer* berjumlah 150. Hasil dapat dilihat pada gambar 4.15, 4.16, 4.17



Gambar 4.12
Hasil 150 *hidden layer* dengan 0.1 *learning rate*

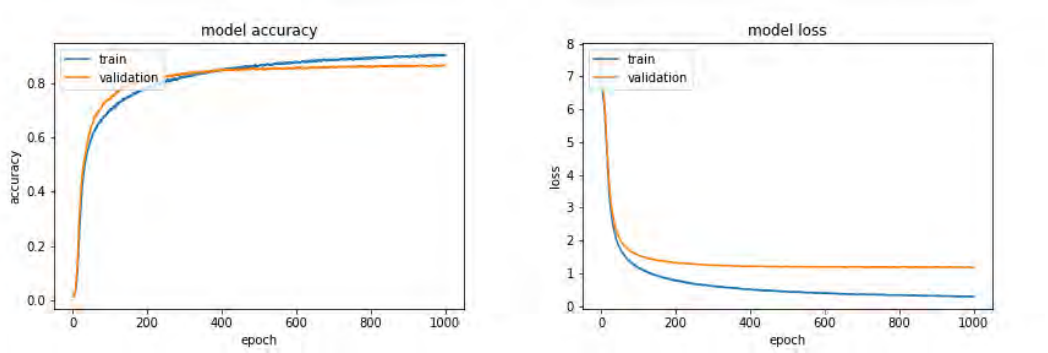
Gambar 4.12 menunjukan hasil dari penggunaan model dengan 150 *hidden layer* dan 0.1 *learning rate*. Dapat dilihat pada gambar, gambar tersebut menunjukan adanya fluktuasi disetiap *epoch*. Fluktuasi ini terjadi karena kecilnya jumlah *learning rate* sehingga membuat model dengan cepat mencapai titik sub optimum. Selain fluktuasi, terjadi juga kenaikan dari nilai *validation loss* yang membuat model ini *overfitting*. Hal ini dikarenakan jumlah neuron pada *input layer* yang besar membuat model lebih komplek. Kompleks nya suatu model akan berdampak pada akurasi, namun tidak selalu membuat akurasi menjadi lebih baik (dapat dilihat perbandingan antara gambar 4.9, 4.12, dan 4.15) Model ini menghasilkan akurasi sebesar 89.7%.



Gambar 4.13
Hasil 150 *hidden layer* dengan 0.01 *learning rate*

Gambar 4.13 menunjukan hasil dari penggunaan model dengan 150 *hidden layer* dan 0.01 *learning rate*. Dapat dilihat pada gambar, pada akurasi, nilai akurasi yang awalnya tertinggal, dapat bertemu di suatu titik dengan nilai validasi. Sementara pada model *loss*, terjadi fluktuasi pada *validation loss*. Hal seperti ini dapat terjadi karena akurasi pada

validasi tidak mengalami kenaikan dan membentuk seperti kurva kecil dengan banyak nilai fluktuasi. Model ini memberikan hasil akurasi sebesar 93.4%



Gambar 4.14
Hasil 150 *hidden layer* dengan 0.001 *learning rate*

Gambar 4.14 menunjukan hasil dari penggunaan model dengan 150 *hidden layer* dan 0.001 *learning rate*. Dapat dilihat pada gambar, pada akurasi, nilai akurasi yang awalnya tertinggal jauh, akhirnya dapat bertemu di suatu titik dengan nilai validasi dan akhirnya menjadi stabil. Sementara pada model *loss*, terjadi penurunan antara *training loss* dan *validation loss* tetapi menjadi stabil mendekati *epoch* akhir. Dengan mengunakan nilai *learning rate* yang makin kecil, maka model akan semakin lama untuk menuju ke nilai sub optimum. Model ini memberikan hasil akurasi sebesar 93.1%.

Dari hasil gambar pengujian dari 4.9 sampai 4.16 dapat ditarik sebuah kesimpulan bahwa:

1. Nilai *learning rate* sangat mempengaruhi kecepatan sebuah model untuk mencapai nilai sub optimum. Hal ini dikarenakan semakin besar nilai *learning rate*, maka semakin cepat model tersebut menuju titik sub optimum.
2. Nilai *learning rate* sangat mempengaruhi sebuah model dalam hal kecepatan untuk mencapai titik sub optimum. Hal ini juga akan mempengaruhi jumlah *epoch* yang digunakan.
3. Nilai *neuron* pada *input layer* sangat mempengaruhi kompleksitas sebuat model. Hal ini akan sangat berpengaruh pada akurasi. Dalam beberapa kasus, semakin banyak *neuron* pada *input layer* akan memberikan akurasi yang besar. Namun tidak semua kasus demikian. Ada yang tidak membutuhkan jumlah *neuron* yang besar. Jika nilai

akurasi pada model tertinggal jauh dari nilai akurasi validasi, maka ada kemungkinan bahwa model tersebut terlalu kompleks untuk data yang diberikan.

Untuk hasil dari penggunaan nilai *neuron* dan *learning rate* yang berupa nilai akurasi, dan RMSE dapat dilihat pada tabel 4.6.

Tabel 4.6
Nilai Akurasi dan RMSE

Learning_rate	**RMSE**	**Akurasi (%)**
50 *neuron*		
0.1	0.267	90.8%
0.01	0.263	93.4%
0.001	0.264	92.3%
100 *neuron*		
0.1	0.263	90.2%
0.01	0.263	93.5%
0.001	0.263	93.3%
150 *neuron*		
0.1	0.264	89.7%
0.01	0.264	93.4%
0.001	0.264	93.1%

Dari tabel 4.6 maka dapat diambil model tertinggi yaitu model dengan 100 *hidden layer* dengan *learning rate* 0.01. Nilai akurasi 93.5% menunjukan bahwa model ini berhasil memprediksi lokasi secara tepat dengan akurasi 93.5%. RMSE dari model dengan 100 *hidden layer* dan 0.001 *learning rate* berjumlah 0.263. RMSE menunjukan berapa selisih antara nilai sebenarnya dan hasil prediksi. Dengan RMSE yang kecil, menunjukan bahwa model ini banyak memprediksi lokasi yang tidak berbeda jauh dari nilai lokasi sebenarnya. Sehingga dapat disimpulkan bahwa model dengan *hidden layer* 100 dan *learning rate* 0.001 adalah model terbaik dari penelitian ini.

4.3.2 Pengujian Fungsi

Pengujian fungsi adalah pengujian yang berfokus pada segala jenis spesifikasi dan fungsi dari sistem yang dibuat. Pengujian ini dilakukan dengan memasukan *input* dan mencantumkan bagaimana sistem merespon pada input tersebut. Penguji akan meilhat hasil dari respon yang sistem keluarkan dan memvalidasi apakah hasil keluaran dari sistem itu sesuai dengan yang diinginkan atau tidak. Jika hasil yang dikeluarkan sesuai dengan

yang diharapkan, maka kesimpulan akan dinyatakan valid. Daftar pengujian fungsi dapat dilihat pada tabel 4.7.

Tabel 4.7
Pengujian Fungsi

No	Aksi	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Memulai sistem rekomendasi	Sistem menampilkan halaman testing	Sistem menampilkan halaman testing	*Valid*
2	Mengklik dropdown pada setiap lokasi	Sistem menampilkan list dari *lat ln* lokasi	Sistem menampilkan list dari *lat ln* lokasi	*Valid*
3	Memilih lokasi dari *dropdown*	Sistem menampilkan nama jalan dari lokasi tersebut dan menampilkan pin poin di google map	Sistem menampilkan nama jalan dari lokasi tersebut dan menampilkan pin poin di google map	*Valid*
4	Men *submit* lokasi dengan total dibawah 4	Sistem mengeluarkan alert isi semua data	Sistem mengeluarkan alert isi semua data	*Valid*
5	Men *submit* lokasi dengan total 4 lokasi	Memulai data testing.	Memulai data testing	*Valid*

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari penulisan Tugas Akhir ini, diambil sejumlah kesimpulan yang dijabarkan dalam beberapa poin seperti berikut:

1. Berdasarkan hasil yang didapatkan, *bidirectional* LSTM (*Long Short-Term Memory*) dapat menghasilkan hasil terbaik dengan menggunakan nilai *learning rate* 0.01 dan *hidden layer* denga jumlah 100, dengan hasil akhir akurasi adalah 93.5% dan RMSE sebesar 0.263. Sehingga dapat disimpulkan bahwa *bidirectional LSTM* dapat digunakan untuk prediksi lokasi berdasarkan data pengguna.
2. Berdasarkan hasil yang didapatkan, parameter yang mempengaruhi hasil dari *Bidirectional* LSTM adalah nilai *learning rate* dan jumlah *hidden layer*.

5.2 Saran

Dari hasil yang sudah disimpulkan, maka diberikan beberapa saran yang membantu untuk penelitian selanjutnya dari penulisan ini. Penggunaan dataset adalah hal yang paling penting dalam *deep learning*. Dataset mempengaruhi kualitas dari model yang akan dibuat. Selain dari dataset, data *cleaning* juga sangat berpengaruh dalam penelitian ini. Data *cleaning* adalah cara untuk menghilangkan *noise* dari dataset yang digunakan. Selain data *cleaning*, data *augmentasi* juga memiliki peran besar dalam menghasilkan akurasi yang tinggi.

Penggunaan metode *bidirectional LSTM* dapat diujicobakan dengan menggunakan lebih banyak LSTM *layer* atau dengan mengubah nilai *dropout*, atau dengan menambahkan tambahan *layer* seperti *Convolutional layer,* atau *Attention layer* ke dalam model yang ada. Selain dari penggunaan metode, dataset yang digunakan juga dapat disesuaikan. Metode ini dapat menggunakan dataset yang bervariasi seperti *movie, next word, sarcasm, toxic chat,* atau dataset pembelian barang. Dari dataset lokasi sendiri, dapat ditambahkan *high season* sebagai salah satu faktor tambahan tertentu apakah pengguna akan menuju ke lokasi tersebut pada musim sekarang.

DAFTAR PUSTAKA

Aggarwal, C.C. (2016) *Recommender Systems*.

Bianchi, M. *et al.* (2017) *Recurrent Neural Networks for Short-Term Load Forecasting An Overview and Comparative Analysis*. Available at: http://www.springer.com/series/10028.

Chung, J. *et al.* (2014) "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling."

Clark, D. (2013) *Beginning C# Object-Oriented Programming*.

Falk, K. (2019) *Practical Recommender Systems*.

Gambs, S.́bastien and Killijian, M.-O. (2012) "Next Place Prediction using Mobility Markov Chains," *EuroSys'12 : proceedings of the EuroSys 2012 Conference, April 10-13, 2012, Bern, Switzerland* [Preprint].

He, J., Li, X. and Liao, L. (2017) *Category-aware Next Point-of-Interest Recommendation via Listwise Bayesian Personalized Ranking*. Available at: https://developer.foursquare.com/categorytree.

Jabbar, H.K. and Khan, R.Z. (2015) *METHODS TO AVOID OVER-FITTING AND UNDER-FITTING IN SUPERVISED MACHINE LEARNING (COMPARATIVE STUDY)*.

Khan, M. and Noor, S. (2019) "Performance Analysis of Regression-Machine Learning Algorithms for Predication of Runoff Time," *Agrotechnology*, 08(01). doi:10.35248/2168-9881.19.8.187.

Kiperwasser, E. and Goldberg, Y. (2016) "Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations."

Kroll, Per. and Kruchten, Philippe. (2003) *The rational unified process made easy : a practitioner's guide to the RUP*. Addison-Wesley.

Milliken, C.P. (2020) *Python Projects for Beginners*, *Python Projects for Beginners*. Apress. doi:10.1007/978-1-4842-5355-7.

Patel, K. (2019) *Overfitting vs Underfitting in Neural Network and Comparison of Error rate with Complexity Graph*, *https://towardsdatascience.com/overfitting-vs-underfitting-ddc80c2fc00d*.

Pressman, R.S. dan Maxim, B.R. (2020) *Software Engineering*.

Ricci, F. *et al.* (2015) *Recommender Systems Handbook Second Edition*.

Rumpe, B. (2017) *Agile modeling with UML: Code generation, testing, refactoring*, *Agile Modeling with UML: Code Generation, Testing, Refactoring*. Springer International Publishing. doi:10.1007/978-3-319-58862-9.

Sommerville, I. (2016) *Software engineering*.

Unhelkar, B. (2018) *Software Engineering with UML*.

Weisfeld, M. (2019) *The Object-Oriented Thought Process Fifth Edition*.

Ying, X. (2019) “An Overview of Overfitting and its Solutions,” in *Journal of Physics: Conference Series*. Institute of Physics Publishing. doi:10.1088/1742-6596/1168/2/022022.

https://www.djangoproject.com/, diakses tanggal 29 April 2021, jam 14.40 WIB

https://docs.conda.io/en/latest/, diakses tanggal 20 April 2021, jam 14.00 WIB

https://research.google.com/colaboratory/faq.html, diakses tanggal 13 Juni 2021, jam 12.00 WIB

LAMPIRAN

1. Dataset

	userId	venueId	venueCategoryId	venueCategory	latitude	longitude	timezoneOffset	utcTimestamp
0	470	49bbd6c0f964a520f4531fe3	4bf58dd8d48988d127951735	Arts & Crafts Store	40.719810	-74.002581	-240	Tue Apr 03 18:00:09 +0000 2012
1	979	4a43c0aef964a520c6a61fe3	4bf58dd8d48988d1df941735	Bridge	40.606800	-74.044170	-240	Tue Apr 03 18:00:25 +0000 2012
2	69	4c5cc7b485a1e21e00d35711	4bf58dd8d48988d103941735	Home (private)	40.716162	-73.883070	-240	Tue Apr 03 18:02:24 +0000 2012
3	395	4bc7086715a7ef3bef9878da	4bf58dd8d48988d104941735	Medical Center	40.745164	-73.982519	-240	Tue Apr 03 18:02:41 +0000 2012
4	87	4cf2c5321d18a143951b5cec	4bf58dd8d48988d1cb941735	Food Truck	40.740104	-73.989658	-240	Tue Apr 03 18:03:00 +0000 2012
...	...	...	...	...	...	...	...	...
227423	688	3fd66200f964a52000e71ee3	4bf58dd8d48988d1e7931735	Music Venue	40.733596	-74.003139	-300	Sat Feb 16 02:29:11 +0000 2013
227424	560	4bca32ff0687ef3be789dbcc	4bf58dd8d48988d16c941735	Burger Joint	40.745719	-73.993720	-300	Sat Feb 16 02:31:35 +0000 2013
227425	945	50a77716e4b0b5a9492f6f56	4bf58dd8d48988d103941735	Home (private)	40.854365	-73.883070	-300	Sat Feb 16 02:33:16 +0000 2013
227426	671	4514efe0f964a520e7391fe3	4bf58dd8d48988d11d941735	Bar	40.735981	-74.029309	-300	Sat Feb 16 02:34:31 +0000 2013
227427	942	4a1e0ca0f964a520bf7b1fe3	4bf58dd8d48988d116941735	Bar	40.726805	-73.957422	-300	Sat Feb 16 02:35:36 +0000 2013

227428 rows × 8 columns

2. Pemodelan data

1. Function TrainingModel

```
def trainingModel(self):
    # LSTM
    print("BUILDING MODEL")
    model = Sequential()
    model.add(Embedding(self.vocab_size, self.input_neuron,
input_length=self.max_sequence_len, trainable=False))
    # model3.add(Bidirectional(LSTM(64, recurrent_dropout=0.6,
return_sequences=True)))
    model.add(Bidirectional(LSTM(64)))
    model.add(Dropout(0.7))
    model.add(Dense(self.vocab_size))
    model.add(Activation('softmax'))

    opt = Adam(learning_rate=self.learning_rate)
    model.compile(loss='categorical_crossentropy', optimizer=opt,
metrics=['accuracy'])
    model.summary()
    print("BUILDING MODEL DONE")
    print("STARTING DATA TRAINING")
    history = model.fit(self.training_data,
                        self.train_80_y,
                        validation_split=0.2,
                        epochs=self.epochs,
                        batch_size=128,
                        verbose=1,
                        shuffle=True)
    # show pyplot loss
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc='upper left')
    plt.show()
    # show pyplot acc
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc='upper left')
    plt.show()

    print("DATA TRAINING DONE")
    print("COUNT ACC AND RMSE")
    # save model
    name = "data"+str(self.input_neuron)+"_"+str(self.learning_rate).replace('.','')+".h5"
    # count accuracy
    predict = model.predict(self.testing_data)
    y_pred = np.argmax(predict, axis=-1)
    y_true = np.argmax(self.test_20_y, axis=-1)
    acc = accuracy_score(y_true, y_pred)

    # count rmse
```

```
    all_loc = []
    for i in self.word_index:
      all_loc.append(self.word_index[i])

    scaler = MinMaxScaler()
    all_loc = np.array(all_loc)
    scaler.fit(all_loc.reshape((all_loc.shape[0], 1)))
    # all_loc = np.array(all_loc)

    y_pred = np.array(y_pred)
    y_true = np.array(y_true)
    y_pred = scaler.transform(y_pred.reshape((y_pred.shape[0], 1)))
    y_true = scaler.transform(y_true.reshape((y_true.shape[0], 1)))

    rmse = mean_squared_error(y_true, y_pred, squared=False)
    self.saveResults(model, name, rmse, acc)
```

2. Function saveModel

```
def saveResults(self, model, name, rmse, acc):
    model.save(name)
    percentage = "{:.0%}".format(acc)
    file1 = open("results.txt","w")
    file1.write("RMSE: "+str(rmse))
    file1.write("ACC: "+percentage)
    file1.close()
```

3. Prediksi Lokasi

1. *Function initMap*

```
function initMap() {
    var directionsService = new google.maps.DirectionsService;
    var directionsDisplay = new google.maps.DirectionsRenderer;
    var map = new google.maps.Map(document.getElementById('map-route'), {
        zoom: 7,
        center: {lat: lat_a, lng: long_a}
    });
    directionsDisplay.setMap(map);
    calculateAndDisplayRoute(directionsService, directionsDisplay);
}
```

2. *Function validateLocation*

```
@csrf_exempt
def validateLocation(request):
      try:
                location_1 = request.GET['id-google-address-a']
                location_2 = request.GET['id-google-address-b']
                location_3 = request.GET['id-google-address-c']
                location_4 = request.GET['id-google-address-d']

                addr_1 = request.GET['text-google-address-a']
                addr_2 = request.GET['text-google-address-b']
                addr_3 = request.GET['text-google-address-c']
                addr_4 = request.GET['text-google-address-d']
```

```
            return TestingRun(request)
    except:
            # return JsonResponse({'locs':[location_1, location_2]})
            return TestingError(request)
```

3. *Function testingData*

```
@csrf_exempt
def TestingData(request):
    location_1 = request.GET['id-google-address-a']
    location_2 = request.GET['id-google-address-b']
    location_3 = request.GET['id-google-address-c']
    location_4 = request.GET['id-google-address-d']

    addr_1 = request.GET['text-google-address-a']
    addr_2 = request.GET['text-google-address-b']
    addr_3 = request.GET['text-google-address-c']
    addr_4 = request.GET['text-google-address-d']
    # define empty array
    arr_location = []
    arr_lat = []
    arr_ln = []
    arr_id = []

    # define models
    models = LocationModel.locs[1:]
    lat_model = LocationModel.lat[1:]
    ln_model = LocationModel.ln[1:]

    arr_id.append(models[int(location_1.split()[0])-2]) if location_1.split()[0] != "0" else
arr_id
    arr_id.append(models[int(location_2.split()[0])-2]) if location_2.split()[0] != "0" else
arr_id
    arr_id.append(models[int(location_3.split()[0])-2]) if location_3.split()[0] != "0" else
arr_id
    arr_id.append(models[int(location_4.split()[0])-2]) if location_4.split()[0] != "0" else
arr_id

    arr_location.append(int(location_1.split()[0])-2)  if location_1.split()[0] != "0" else
arr_location
    arr_location.append(int(location_2.split()[0])-2)  if location_2.split()[0] != "0" else
arr_location
    arr_location.append(int(location_3.split()[0])-2)  if location_3.split()[0] != "0" else
arr_location
    arr_location.append(int(location_4.split()[0])-2)  if location_4.split()[0] != "0" else
arr_location

    print(location_3)
    # return JsonResponse({'locs':[location_4, arr_ln]})
    print(models[int(location_1.split()[0])-1])
    # return JsonResponse({'locs':[location_3, int(location_3.split()[0])-1]})
    # upload the tokenizer file
    with open('data/data_tokenize.json') as json_file:
            data = json.load(json_file)

    # tokennize the input
    # return JsonResponse({'locs':[arr_id]})
    arr_lat.append(location_1.split()[1])
```

```
arr_lat.append(location_2.split()[1])
arr_lat.append(location_3.split()[1])
arr_lat.append(location_4.split()[1])

arr_ln.append(location_1.split()[2])
arr_ln.append(location_2.split()[2])
arr_ln.append(location_3.split()[2])
arr_ln.append(location_4.split()[2])

data_test = []

for i in arr_id:
        data_test.append(data[i])


np_test = np.array(data_test)
tf_model = tf.keras.models.load_model('<model_name>'')
testing = np_test.reshape(1, 4, 1)
result = tf_model.predict(testing)
tok_loc = np.argmax(result, axis=-1)

# get the venue name from result
index = 0
loc_id = ""
for key, value in data.items():
        if value == tok_loc.tolist()[0]:
                loc_id = key
                break

for i in models:
        if i == loc_id:
                # get lat ln
                lat_result = lat_model[index]
                ln_result = ln_model[index]
                break
        index += 1

# return JsonResponse({'locs':[loc_id]})
# get lat ln
# lat_result = lat_model[index]
# ln_result = ln_model[index]

# save output in array
arr_lat.append(lat_result)
arr_ln.append(ln_result)
arr_id.append(loc_id)

lat_a = arr_lat[0]
long_a = arr_ln[0]
lat_b = arr_lat[1]
long_b = arr_ln[1]
lat_c = arr_lat[2] if len(arr_lat) > 2 else None
long_c = arr_ln[2] if len(arr_lat) > 2 else None
lat_d = arr_lat[3] if len(arr_lat) > 3 else None
long_d = arr_ln[3] if len(arr_lat) > 3 else None
lat_e = arr_lat[4] if len(arr_lat) > 4 else None
long_e = arr_ln[4] if len(arr_lat) > 4 else None
```

```
# return JsonResponse({'locs':[lat_c, long_c]})

origin = []
destination = []

if lat_e is not None:
        destination.append(lat_e)
        destination.append(long_e)
        origin.append(lat_d)
        origin.append(long_d)
elif lat_d is not None:
        destination.append(lat_d)
        destination.append(long_d)
        origin.append(lat_c)
        origin.append(long_c)
elif lat_c is not None:
        destination.append(lat_c)
        destination.append(long_c)
        origin.append(lat_b)
        origin.append(long_b)
elif lat_b is not None:
        destination.append(lat_b)
        destination.append(long_b)
        origin.append(lat_a)
        origin.append(long_a)

#only call API if all 4 addresses are added
if lat_a and lat_b:
        directions = Directions(
                lat_a= lat_a,
                long_a=long_a,
                lat_b = lat_b,
                long_b=long_b,
                lat_c= lat_c,
                long_c=long_c,
                lat_d = lat_d,
                long_d=long_d,
                lat_e = lat_e,
                long_e=long_e
                )
else:
        return redirect(reverse('main:model'))

ori = []
dest = []

if lat_e is not None:
        dest.append(lat_e)
        dest.append(long_e)
        ori.append(lat_d)
        ori.append(long_d)
elif lat_d is not None:
        dest.append(lat_d)
        dest.append(long_d)
        ori.append(lat_c)
        ori.append(long_c)
elif lat_c is not None:
        dest.append(lat_c)
```

```
                dest.append(long_c)
                ori.append(lat_b)
                ori.append(long_b)
        elif lat_b is not None:
                dest.append(lat_b)
                dest.append(long_b)
                ori.append(lat_a)
                ori.append(long_a)
        # return JsonResponse({'locs':[directions, lat_d]})
        context = {
        "google_api_key": settings.GOOGLE_API_KEY,
        "base_country": settings.BASE_COUNTRY,
        "lat_a": lat_a,
        "long_a": long_a,
        "lat_b": lat_b,
        "long_b": long_b,
        "lat_c": lat_c,
        "long_c": long_c,
        "lat_d": lat_d,
        "long_d": long_d,
        "lat_e": lat_e,
        "long_e": long_e,
        "origin": f'{ori[0]}, {ori[1]}',
        "destination": f'{dest[0]}, {dest[1]}',
        "directions": directions,
        "addr_1": addr_1,
        "addr_2": addr_2,
        "addr_3": addr_3,
        "addr_4": addr_4,
        "arr_id": "-".join(arr_id),
        }
        return render(request, 'main/map.html', context)
```

4. ***Function testing.html***

```
        {% extends 'base.html' %}
        {% load static %}
        {% block extend_head %}
        {% endblock %}
        {% block content %}
        <h3>Prediksi Lokasi</h3>
        {% if messages %}
            {% for message in messages %}
                <div class="alert {% if message.tags %}alert-{{ message.tags }}{% endif %}"
role="alert">{{ message }}</div>
            {% endfor %}
        {% endif %}
        <div class="container">
          <form action="{% url 'model:results' %}">
            <label for="id-google-address-a">Location 1</label>
            <select              name="id-google-address-a"              id="id-google-address-a"
onchange="changeSelect('A')">
              <option value="NONE" selected="true">NONE</option>
              {% for location in list_loc.loc %}
                <option value="{{ location }}">{{ location }}</option>
              {% endfor%}
            </select>
            <input type="text" id="text-google-address-a" name="text-google-address-a">
            <label for="id-google-address-b">Location 2</label>
```

```
            <select name="id-google-address-b" id="id-google-address-b"
onchange="changeSelect('B')">
              <option value="NONE" selected="true" disabled>NONE</option>
              {% for location in list_loc.loc %}
                <option value="{{ location }}">{{ location }}</option>
              {% endfor%}
            </select>
            <input type="text" id="text-google-address-b" name="text-google-address-b">
            <label for="id-google-address-c">Location 3</label>
            <select name="id-google-address-c" id="id-google-address-c"
onchange="changeSelect('C')">
              <option value="NONE" selected="true" disabled>NONE</option>
              {% for location in list_loc.loc %}
                <option value="{{ location }}">{{ location }}</option>
              {% endfor%}
            </select>
            <input type="text" id="text-google-address-c" name="text-google-address-c">
            <label for="id-google-address-d">Location 4</label>
            <select name="id-google-address-d" id="id-google-address-d"
onchange="changeSelect('D')">
              <option value="NONE" selected="true" disabled>NONE</option>
              {% for location in list_loc.loc %}
                <option value="{{ location }}">{{ location }}</option>
              {% endfor%}
            </select>
            <input type="text" id="text-google-address-d" name="text-google-address-d">
            <div class="form-button mt-3">
              <button id="submit" type="submit" class="btn btn-primary">
                Get Recommendation
              </button>
            </div>
          </form>
        </div>
        <div class="map-container">
          <div id="map-route"></div>
        </div>
        {% endblock %}
        {% block extend_footer %}
          <script type="text/javascript">
            var google_api_key = "{{google_api_key|safe}}";
            var base_country = "{{base_country|safe}}";
            var arr_loc = [];
            function changeSelect(part) {
              switch (part) {
                case 'A':
                  if(arr_loc.length > 0){
                    arr_loc[0] = document.getElementById('id-google-address-a').value;
                  }
                  else{
                    arr_loc.push(document.getElementById('id-google-address-a').value);
                  }
                  showRoadName(document.getElementById('id-google-address-a').value,
"text-google-address-a")
                  break;
                case 'B':
                  //                document.getElementById("id-lat-"+"b").value                =
document.getElementsById('id-google-address-b').value;
                  if(arr_loc.length > 1){
```

```
                    arr_loc[1] = document.getElementById('id-google-address-b').value;
                  }
                  else{
                    arr_loc.push(document.getElementById('id-google-address-b').value);
                  }
                  showRoadName(document.getElementById('id-google-address-b').value,
"text-google-address-b")
                    break;
                  case 'C':
                    //                  document.getElementById("id-lat-"+"c").value                  =
document.getElementsById('id-google-address-c').value;
                    if(arr_loc.length > 2){
                      arr_loc[2] = document.getElementById('id-google-address-c').value;
                    }
                    else{
                      arr_loc.push(document.getElementById('id-google-address-c').value);
                    }
                    showRoadName(document.getElementById('id-google-address-c').value,
"text-google-address-c")
                    break;
                  case 'D':
                    //                  document.getElementById("id-lat-"+"d").value                  =
document.getElementsById('id-google-address-d').value;
                    if(arr_loc.length > 3){
                      arr_loc[3] = document.getElementById('id-google-address-d').value;
                    }
                    else{
                      arr_loc.push(document.getElementById('id-google-address-d').value);
                    }
                    showRoadName(document.getElementById('id-google-address-d').value,
"text-google-address-d")
                    break;
                  default:
                    break;
                }
                if(arr_loc.length > 0){
                  mapTraining(arr_loc)
                }
              }
              function setGetParameter(){
                var location = []
                var road_name = []

                if     (document.getElementById('id-google-address-a').value     !=     ''     &&
document.getElementById('id-google-address-a').value != 'CHOOSE DATA') {
                  location.push(document.getElementById('id-google-address-a').value);
                  road_name.push(document.getElementById('text-google-address-
a').value);
                }
                if(document.getElementById('id-google-address-b').value       !=       ''       &&
document.getElementById('id-google-address-b').value != 'CHOOSE DATA') {
                  location.push(document.getElementById('id-google-address-b').value);
                  road_name.push(document.getElementById('text-google-address-
b').value);
                }
                if(document.getElementById('id-google-address-c').value       !=       ''       &&
document.getElementById('id-google-address-c').value != 'CHOOSE DATA') {
                  location.push(document.getElementById('id-google-address-c').value);
```

```
                road_name.push(document.getElementById('text-google-address-c').value);
              }
              if(document.getElementById('id-google-address-d').value != '' &&
document.getElementById('id-google-address-d').value != 'CHOOSE DATA') {
                location.push(document.getElementById('id-google-address-d').value);
                road_name.push(document.getElementById('text-google-address-
d').value);
              }
              document.getElementById("testing").innerHTML = location;
              if(location.length < 4){
                window.alert('Please choose at least 4 locations')
              }
              else{
                passParams(location)
              }
             }
            </script>

            <script src="{% static 'preprocess_testing.js' %}"></script>
            <script src="{% static 'google_maps_training_page.js' %}"></script>
          {% endblock %}
```

5. ***Function map.html***

```
          {% extends 'base.html' %}
          {% load static %}
          {% block extend_head %}
          {% endblock %}
          {% block content %}
          <h3 id="map">Next Location Recommendation</h3>
          <div class="container">
            <table>
              <thead>
                <tr>
                  <th>Field</th>
                  <th></th>
                </tr>
              </thead>
              <tbody>
                <tr>
                  <td>Location A</td>
                  <td>{{addr_1}}</td>
                </tr>
                <tr>
                  <td>Location B</td>
                  <td>{{addr_2}}</td>
                </tr>
                <tr>
                  <td>Location C</td>
                  <td>{{addr_3}}</td>
                </tr>
                <tr>
                  <td>Location D</td>
                  <td>{{directions.locs_ori}}</td>
                </tr>
                <tr>
                  <td>Location D lat ln</td>
                  <td>{{directions.origin}}</td>
                </tr>
```

```
                <tr>
                  <td>Prediction</td>
                  <td>{{directions.locs_destination}}</td>
                </tr>
                <tr>
                  <td>Prediction lat ln</td>
                  <td>{{directions.destination}}</td>
                </tr>
                <tr>
                  <td>Duration</td>
                  <td>{{directions.duration}}</td>
                </tr>
                <tr>
                  <td>Distance</td>
                  <td>{{directions.distance}}</td>
                </tr>
                <tr>
                  <td>Directions</td>
                  <td id="dir-toggle">click <a href="javascript:void(0)"
onclick="DirectionsToggle()">here</a></td>
                </tr>
              </tbody>
            </table>
            <table id="dir-table" hidden>
              <thead>
                <tr>
                  <th>Directions</th>
                  <th>Distance</th>
                  <th>Duration</th>
                </tr>
              </thead>
              <tbody>
                {% for leg in directions.route %}
                    <tr>
                    <td>Leg {{ forloop.counter }}</td>
                    <td></td>
                    <td></td>
                  </tr>
                  {% for dist, dur, text in leg.steps %}
                  <tr>
                    <td>{{text|safe}}</td>
                    <td>{{dist}}</td>
                    <td>{{dur}}</td>
                  </tr>
                  {% endfor %}
                {% endfor %}
              </tbody>
            </table>
          </div>
          <div class="map-container">
            <div id="map-route"></div>
          </div>
          {% endblock %}
          {% block extend_footer %}
          <script type="text/javascript">
              var google_api_key = "{{google_api_key|safe}}";
              var lat_a = {{lat_a|safe}};
              var long_a = {{long_a|safe}};
```

```
    var lat_b = {{lat_b|safe}};
    var long_b = {{long_b|safe}};
    var lat_c = {{lat_c|safe}};
    var long_c = {{long_c|safe}};
    var lat_d = {{lat_d|safe}};
    var long_d = {{long_d|safe}};
    var lat_e = {{lat_e|safe}};
    var long_e = {{long_e|safe}};
    var origin = "{{origin|safe}}";
    var destination = "{{destination|safe}}";
    var directions = {{directions|safe}};
  </script>
  <script src="{% static 'google_maps.js' %}"></script>
{% endblock %}
```