BAB II
TINJAUAN PUSTAKA

2.1. Kajian Pustaka

Penelitian yang melakukan pengolahan data teks sudah sangatlah banyak diterapkan baik menggunakan metode *Deep Learning (DL)*. Khususnya pada studi kasus *text generation*, dengan implementasi *DL* dapat menghasilkan sebuah model yang dapat berpikir seperti manusia untuk menyelesaikan masalah, pendekatan tersebut dapat meningkatkan secara signifikan terhadap kinerja sebuah aplikasi yang dapat mendeteksi suatu teks dan menghasilkan *output generate text.*

Tahun 2018, Mrs. Dipti Pawade et al. mempublish penelitian yang berjudul *Story Scrambler - Automatic Text Generation Using Word Level RNN-LSTM*. Penelitian tersebut membangun sebuah model *text generator* menggunakan algoritme *RNN* dan *LSTM*. Dengan mengimplementasikan *one-hot encoding* untuk target *dataset* dan menggunakan optimizer *SGD.* Hasilnya adalah sebuah model yang dapat menghasilkan *generate text* sebuah cerita dengan akurasi sebesar 63% berdasarkan metrik evaluasi kemiripan manusia [3].

Penelitian yang lain pada tahun yang sama, dipublikasikan oleh Tianyu Liu et al. Membawakan judul *Table-to-Text Generation by Structure-Aware Seq2seq Learning* menggunakan data *WIKBIO.* Data tersebut dibuat oleh Lebret, Grangier dan Auli pada tahun 2016, data tersebut berisi lebih dari 700.000 data biografi bersumber dari wikipedia dengan total kata sebesar 400.000. Penelitian ini membangun sebuah *text generator* menggunakan algoritme LSTM, meneliti mengenai komparasi performa algoritme *LSTM* dan algoritme *Kneser-Ney (KN)*. Hasilnya adalah model yang dapat menghasilkan *generate seq2seq word* dengan nilai 2.21 untuk model *KN* dan nilai 40.04 menggunakan algoritme *LSTM* berdasarkan evaluasi *BLEU-4* [7].

Penelitian selanjutnya Afroz Ahamad yang berjudul, *Generating Text through Adversarial Training using Skip-Thought Vectors* pada tahun 2020. Membangun sebuah model *text generation* menggunakan data bersumber dari *BookCourpus* sebanyak 250.000 kalimat, Afroz mengimplementasikannya menggunakan metode *Skip-Thought Generative Advesarial Network (STGAN)*, dengan menggunakan preprocessing tokenizing dimana pada masing masing kalimat akan ditambahkan <*s*> sebelum *start token* dan </*s*> sebagai *end token* dalam proses *encode*. Proses *decoding* akan berhenti ketika terdapat </*s*>. Penelitian ini menghasilkan sebuah model yang dapat menghasilkan *output generate text* dengan akurasi 62% berdasarkan evaluasi persepsi kesamaan manusia dengan membandingkan 5 tulisan model dan 5 tulisan dari Afroz yang ditunjukan oleh responden *expert* dalam bidang penulisan [8].

Penelitian yang berjudul *Context Based Text-Generation Using LSTM Networks* dibuat oleh Sivasurya Santhanam, dipublikasikan pada tahun 2020. Penelitian ini membangun model *text generator* menggunakan data *The Lord of Rings Dataset.* Dengan menggunakan algoritme *LSTM,* penelitian tersebut menghasilkan model *text generator* berdasarkan beberapa *input* kata sebagai konteks dapat membuat sebuah cerita berdasarkan konteks yang ada. Penelitian ini menghasilkan performa model dengan akurasi sebesar 97% berdasarkan metrik *validation accuracy* dan *loss* [9].

Penelitian selanjutnya diteliti oleh M. M. Moila et al. yang berjudul *The Development of A Sepedi Text Generation Model using Long-Short Term Memory.* Penelitian tersebut membangun sebuah model *text generator* menggunakan *NCHLT Sepedi text corpus. Sepedi* adalah bahasa utama di Afrika Selatan, Moila mengimplementasikan model *text generator* menggunakan algoritme *LSTM.* Penelitian ini menghasilkan model text generation dengan akurasi sebesar 50% dan loss sebesar 1.6145 berdasarkan evaluasi metrik validation accuracy dan loss [10].

Dataset, corpus ini berisi koleksi percakapan fiksi yang kaya akan metadata yang diekstrak dari skrip film mentah, 220.579 percakapan antara 10.292 pasang karakter film melibatkan 9.035 karakter dari 617 film dengan total 304.713

percakapan. Penelitian ini dibangun dengan menggunakan algoritme *LSTM* dan *GRU* dan menghasilkan model seq2seq yang dengan sistem question answer [11].

Penelitian oleh H. V. Krishna Sai Buddana et al. yang berjudul *Word Level LSTM and RNN for Automatic Text Generation* pada tahun 2021. Penelitian ini menggunakan *Story dataset from Republic of Plato*, dan mengimplementasikan algoritme *LSTM* dan *GRU* untuk membuat model *text generator*. Hasil dari kombinasi algoritme tersebut adalah model yang dapat menghasilkan *generate text* dengan akurasi sebesar 80% berdasarkan metrik *validation accuracy* dan *loss* [12].

Dari penjelasan di atas, ringkasan penelitian yang relevan ditunjukan pada **Tabel 2.1** di bawah ini :

Tabel 2. 1. Kajian Pustaka

No	Penulis	Tahun	Objek	Metode Penyelesaian	Metode Pengumpulan Data	Hasil
1	Mrs. Dipti Pawade et al.	2018	*Text Generator.*	*RNN, LSTM.*	*Story dataset.*	Menghasilkan model *text generator* yang dapat mengarang cerita dengan tingkat kemiripan sebesar 63% berdasarkan evaluasi persepsi kemiripan oleh manusia.
2	Tianyu Liu et al.	2018	*Text Generator.*	*LSTM.*	*WIKBIO contains 728,321 articles from English Wikipedia.*	Menghasilkan model *text generator seq2seq* dengan nilai 40.04 berdasarkan metrik *BLEU.*
3	Afroz Ahamad.	2020	*Text Generator.*	*Skip-Thought GAN.*	*250.000 kalimat dari BookCorpus dataset.*	Menghasilkan model *text generator* dengan tingkat kemiripan *text* terhadap hasil tulisan manusia, sebesar 63%.
4	Sivasurya Santhanam.	2020	*Text Generator.*	*LSTM.*	*The Lord of the Rings dataset.*	Menghasilkan model *text generator* dengan akurasi sebesar 97% berdasarkan metrik *validation accuracy* dan *loss.*

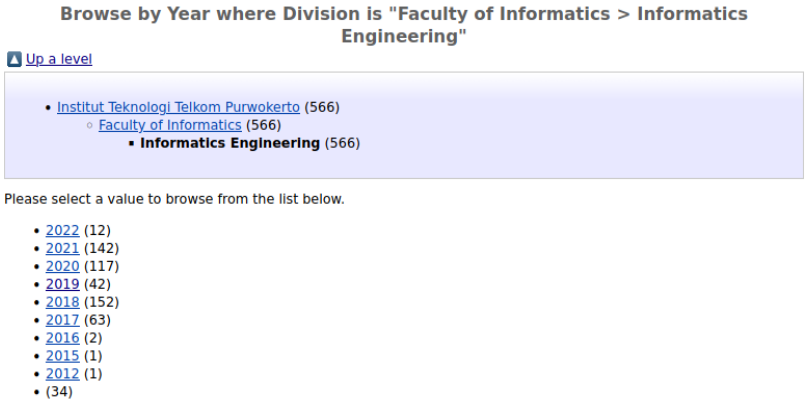
No	Penulis	Tahun	Objek	Metode Penyelesaian	Metode Pengumpulan Data	Hasil
5	M. M. Moila et al.	2020	*Text Generator.*	*LSTM, GRU*	*NCHLT Sepedi text corpus.*	Menghasilkan model *text generation* dengan akurasi sebesar 50% dan loss sebesar 1.6145 berdasarkan evaluasi metrik *validation accuracy* dan *loss.*
6	Dr. Manoj Kumar et al.	2021	*Text Generator.*	*LSTM, GRU.*	*Cornell Movies Dataset.*	Menghasilkan model *seq2seq* yang dengan sistem *question answer.*
7	H. V. Krishna Sai Buddana et al.	2021	*Text Generator.*	*LSTM.*	*Story dataset from Republic of Plato.*	Menghasilkan model *text generator* dengan akurasi sebesar 80% berdasarkan metrik *validation accuracy* dan *loss*.

Berdasarkan Tabel 2.1. dapat disimpulkan bahwa data teks dapat diproses melalui algoritme *DL*, khususnya dalam studi kasus *text generator*, menggunakan data *long text sequences LSTM* dan *GRU* memiliki performa yang baik. *LSTM* dan *GRU* adalah jawaban dari permasalahan *vanishing gradient* yang ada pada *RNN*. Penelitian dari Mrs. Dipti Pawade et al. menjadi acuan utama untuk penelitian ini, dengan mengimplementasikan metode evaluasi dengan konsep yang sama, serta memiliki kemiripan penggunaan algoritme dan *preprocessing*.

2.2. Dasar Teori

2.2.1. *Data Acquisition*

Dataset diambil dengan menggunakan metode *download* pada platform *repository* kampus Institut Teknologi Telkom Purwokerto (ITTP) pada bagian bab 1, dan bab 2 semua prodi pada mahasiswa angkatan 2014 – 2018 berdasarkan data yang sudah terpublish pada *repository* ITTP.



Gambar 2. 1. *Repository* Prodi S1 Teknik Informatika

Pengambilan *dataset* memerlukan perizinan mengenai kerahasiaan data. Pihak perpustakaan turut membantu dalam pengambilan *dataset*, meskipun demikian karena menggunakan metode *download* sehingga memerlukan jangka waktu kurang lebih satu bulan.

2.2.2. *Preprocessing*

Dalam *Artificial Intelligence (AI)*, data merupakan hal yang sangat penting, karena data adalah hal yang akan pelajari oleh mesin sehingga semakin banyak jumlah suatu data akan semakin baik model akan belajar pada proses *training*. Maka dari itu perlu adanya proses *preprocessing* untuk membersihkan dan mempersiapkan data. Data mentah mempunyai banyak *noise* dan tidak informatif sehingga perlu dilakukan suatu teknik untuk mengubah dari data mentah dalam bentuk lain tanpa mengubah representatif dari data tersebut, setelah data dilakukan proses *preprocessing* data baru dapat diolah ke tahapan selanjutnya yaitu *modelling* [13].

2.2.2.1. *Regular Expression*

Regular Expression (regex) adalah sebuah alat untuk melakukan pencocokan data, secara umum penggunaan *regex* adalah dengan melakukan identifikasi pola terhadap tipe data *string*, selanjutnya menghasilkan data yang dapat digunakan kembali oleh peneliti. *Regex* dapat digunakan dalam banyak Bahasa pemrograman seperti *Python, JavaScript, Java, SQL,* dan banyak lagi. Pola *regex* adalah urutan karakter yang mendefinisikan pencocokan. Dengan hal ini kita dapat mencari kata tertentu, angka, karakter khusus, atau kombinasi pola tertentu dalam suatu teks. Meskipun sangat kuat dan fleksibel, *regex* dapat menjadi rumit dan sulit untuk dipahami, terutama ketika pola yang dicocokan menjadi sangat kompleks sebagai contoh "[A-Za-z]" pola tersebut akan mencocokan teks yang hanya terdiri huruf saja (tanpa ada angka atau karakter khusus) serta mengabaikan karakter lainnya [14].

2.2.2.2. *Tokenization*

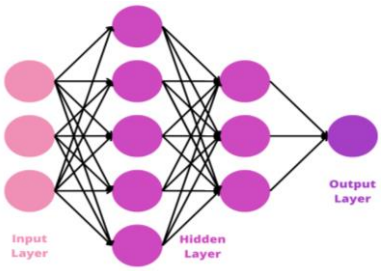
Dalam data teks menjadi suatu hal yang penting untuk menemukan batasan kata. Batas – batas tersebut dibatasi dengan penggunaan spasi dan tanda baca. Proses pemisahan kalimat menjadi bagian yang bermakna dan mengidentifikasi entitas individu didalam suatu kalimat disebut *token, token* dapat berupa kata, frase, atau karakter. Tujuan utama dari teknik ini adalah untuk mengubah teks kontinu menjadi unit diskret yang dapat diolah lebih lanjut. Dalam bahasa manusia, kita cenderung menggunakan spasi untuk memisahkan kata – kata dalam kalimat. Namun, tokenisasi juga dapat lebih kompleks, tergantung pada bahasa dan konteksnya [15].

Tabel 2. 2 *Tokenizing*

No	Data	Tokenized Data
1	Aku suka kamu	[1, 2, 3]
…	…	…
100	Suka sama suka	[2, 4, 2]

2.2.3. Pembelajaran Mendalam

Pembelajaran mendalam merupakan jaringan yang tersusun dari beberapa *layer*, *layer* tersebut merupakan kumpulan dari *node – node*, dalam node tersebut terjadi penghitungan terjadi. Sebuah node input digabung dengan bobot *(weight)* dan menghasilkan nilai pada *output*. Pembelajaran mendalam memiliki system *backpropagation* yaitu metode untuk menyesuaikan *weight* dalam jaringan syaraf. Proses ini melibatkan perhitungan gradien kesalahan *(error)* dan penggunaanya untuk memperbarui parameter agar model dapat menghasilkan performa prediksi yang lebih baik [16].



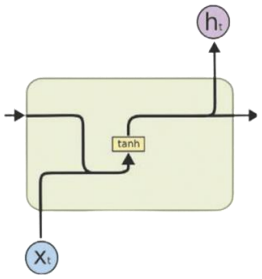
Gambar 2. 2. *Neural Network*

Masing - masing *layer* mempunyai parameter yang dapat disesuaikan jumlah *units*, penyesuaian parameter dapat mempengaruhi *output* model, dalam penelitian ini arsitektur yang dibangun menggunakan :

2.2.3.1. *Embedding Layer*

Sebelum diolah dengan model *Machine Learning (ML)* atau *DL*, data teks harus diubah ke dalam bentuk angka, proses pengubahan angka menjadi *vektor* disebut dengan *embedding*. Konsep dasar dari *embedding layer* adalah merepresentasikan setiap kata pada *corpus* dengan vektor yang mencerminkan hubungan dan struktur semantik masing masing kata. Dalam representas vektor, kata – kata yang serupa atau berkaitan secara semantik akan memiliki vektor yang mendekati satun sama lain dalam dimensi vektor. Hasil *embedding* tersebut digunakan dalam pemodelan dan berfungsi sebagai dasar untuk komputasi *NLP* [15].

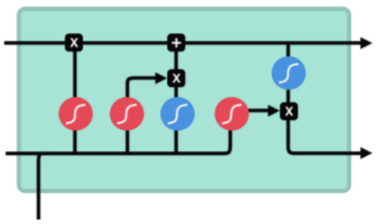
2.2.3.2. *Recurrent Neural Network (RNN)*



Gambar 2. 3. *RNN*

Dalam bentuk paling dasar dari *Recurrent Neural Network (RNN)*. Model paling dasar dari *VRNN* dapat direpresentasikan pada Gambar 2.3. Normalnya fungsi aktivasi *tanh* digunakan pada *layer* recurrent tersembunyi dan fungsi aktivasi pada *output layer* dapat digunakan berdasarkan kebutuhan dari permasalahan yang diselesaikan [4]. *VRNN* memiliki tantangan menyimpan informasi tentang masa lalu yang jauh dalam *VRNN* dikaitkan dengan masalah *gradient* yang hilang pada saat pelatihan. Artinya untuk data dalam urutan yang panjang saat melalui proses *backpropagation*, akan menyusut secara eksponensial menjadi nol. Sehingga membuat lebih sulit untuk mempelajari data dengan urutan yang panjang [4].

2.2.3.3. *Long Short Term Memory (LSTM)*

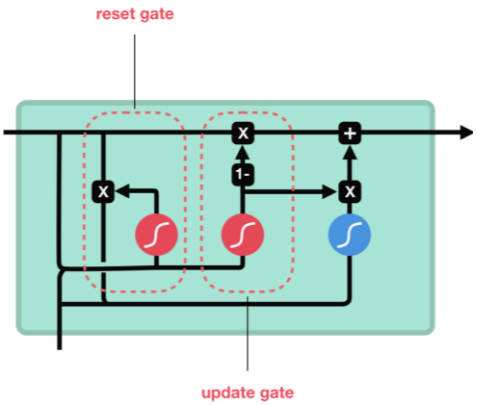


Gambar 2. 4. *LSTM*

LSTM adalah *layer RNN* dengan mekanisme gerbang untuk mempelajari data *sequential* panjang. Memori *LSTM*, disebut *cells*, menerima input saat ini dan *hidden state* sebelumnya, memilih informasi yang disimpan, dan menggabungkan keduanya untuk masukan berikutnya. Dengan cara ini, *LSTM* dapat mengatasi masalah *vanishing gradient* selama pelatihan dan mengoptimalkan kinerja pada data dengan *sequence* panjang[17].

LSTM memiliki mekanisme khusus yang disebut *gates, forget gate* berfungsi untuk memutuskan informasi apa yang akan dihapus dari *cell* sebelumnya. *Gate* ini yang memutuskan apakah informasi harus dihapus atau dipertahankan, *input gate* berfungsi untuk memutuskan informasi apa saja yang akan masuk kedalam *cell* dari input saaat ini dan output dari *cell* sebelumnya. Gate ini memungkinkan model untuk memutuskan seberapa banyak informasi yang harus diingat dari input baru dan seberapa informasi yang harus disimpan pada memori sebelumnya. Terakhir adalah *output gate,* berfungsi untuk memutuskan output dari sel memori berdasarkan *input* saat ini dan informasi dari *cell* sebelumnya. Output gate mengatus seberapa banyak informasi yang akan dikeluarkan [17].

2.2.3.4. *Gated Recurrent Unit (GRU)*

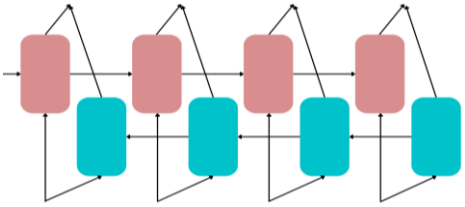


Gambar 2. 5. *GRU*

GRU seperti *LSTM*, dirancang untuk menengani masalah memori pada *VRNN*. *GRU* berbeda dengan *LSTM* dengan memanfaatkan gerbang pembaruan dan gerbang reset. Tugas dari gerbang tersebut adalah menentukan jumlah informasi dari *layer* sebelumnya yang akan dipindahkan ke-*layer* selanjutnya atau dibuang. *GRU* memiliki 2 gerbang, pertama yaitu *reset gate*, berfungsi untuk memutuskan berapa informasi dari sel sebelumnya yang harus diabaikan, gerbang ini berperan untuk mengatur sejauh apa informasi dari *step* sebelumnya harus dilupakan sebelum memproses input saat ini. Selanjutnya adalah update gate, berfungsi untuk memutuskan seberapa banyak informasi harus disimpan dari *step* sebelumnya ke

sel memori saat ini. *Gate* ini berfungsi untuk mengatur seberapa banyak informasi baru dan inputan saat ini yang harus dimasukkan ke dalam sel memori untuk dijadikan input pada *layer* selanjutnya [18].

2.2.3.5. *Bidirectional Architecture*

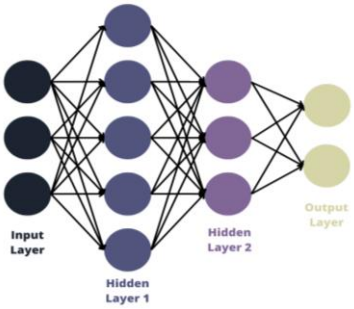


Gambar 2. 6. *Bidirectional architecture*

Pada dasarnya *LSTM* dan *GRU* hanya memiliki memori arah maju *(forward memory)* yang berarti masukan saat ini hanya mempengaruhi proses selanjutnya. Namun, dalam beberapa konteks, informasi dari data di masa depan juga memberikan wawasan yang berharga untuk memahami data saat ini. *Bidirectional Architecture* adalah arsitektur yang berjalan dua arah, alih alih sebuah *network* berjalan ke satu arah, dengan menggunakan arsitektur *bidirectional* mampu membuat *network* berjalan ke arah sebaliknya, sehingga model mampu meningkatkan pemahaman yang lebih dibandingkan dengan arsitektur jaringan syaraf tiruan yang berjalan satu arah [19].

Keuntungan dari penggunaan arsitektur bidirectional adalah kemampuanya untuk mengakses informasi dari kedua arah urutan data, sehingga jaringan dapat memanfaatkan konteks yang lebih kaya dan meningkatkan pemahaman secara keseluruhan. *Bidirectional architecture* sering digunakan dalam studi kasus *NLP* seperti text generation dan sentiment analysis dimana konteks dari kata – kata suatu frasa yang ada disekitarnya menjadi sangat penting untuk menghasilkan hasil yang lebih baik [19].

2.2.3.6. *Fully Connected Layer (FCL)*



Gambar 2. 7. *Fully Connected Layer*

Fully Connected Layer (FCL), juga dikenal sebagai *Dense Layer*, adalah *layer* dalam *neural network* yang menghubungkan setiap neuron dari lapisan sebelumnya ke setiap neuron pada lapisan berikutnya. Pada *FCL*, setiap *neuron* pada lapisan ini menerima input dari seluruh *output* neuron pada lapisan sebelumnya. *FCL* merupakan tipe jaringan yang terhubung sepenuhnya yang terdiri dari lapisan *input* yang menerima masukan, *FCL* merupakan salah satu jenis jaringan yang paling umum dalam *neural network* dan sering digunakan sebagai *layer* terakhir pada arsitektur *DL* [20].

Lapisan ini bertanggung jawab untuk mengambil representasi fitur yang kompleks dari lapisan sebelumnya dan memprosesnya menjadi *output* yang diinginkan, misalnya kelas prediksi dalam studi kasus klasifikasi, atau nilai yang diharapkan dalam studi kasus regresi. *Layer* ini bertanggung jawab untuk mengambil representasi fitur yang kompleks dari lapisan sebelumnya dan memprosesnya menjadi output yang diinginkan, misalnya kelas prediksi dalam tugas klasifikasi, atau nilai yang diharapkan dalam tugas regresi [20].

2.2.4. Evaluasi

Hasil dari model dapat diproyeksikan melalui evaluasi, dengan evaluasi model dapat memvisualisasikan performa model, berikut merupakan beberapa evaluasi yang dapat digunakan dalam penelitian ini :

2.2.4.1. *Loss*

Fungsi *loss* adalah kriteria evaluasi yang digunakan untuk menghitung perbedaan antara prediksi model dengan nilai target yang sebenarnya. Tujuannya adalah untuk mengukur tingkat kesalahan prediksi model dan mengarahkan model agar meminimalkan kesalahan ini selama proses *training* berlangsung. Dalam proses *training* terdapat variabel *loss* dan akurasi, *loss* dapat digunakan untuk melihat performa dari model yang sudah dibuat. Berdasarkan variabel tersebut model yang mendekati nol atau kurang dari satu dengan akurasi yang terus meningkat menunjukan hasil yang baik seiring berjalannya *epoch* [21].

2.2.4.2. Persepsi Kesamaan Manusia

Sebuah pendekatan untuk menghasilkan kalimat baru dievaluasi dalam mereproduksi gaya penulisan berdasarkan peneliti. Sebanyak 5 sampel *output* model dibandingkan dengan 5 tulisan peneliti. Akurasi dihitung berdasarkan ketidakpastian dari penilaian responden, evaluasi ini ditunjukan untuk mengukur seberapa dekat kemiripan teks buatan dari model dengan buatan manusia, evaluasi ini dapat dilakukan dengan responden yang ahli dalam bidang penulisan dengan jumlah terbatas [8].

2.2.5. *Streamlit Framework*

Streamlit dapat digunakan dalam *ML* dan *DL* model *deployment*. *Streamlit* merupakan *framework open-source* untuk membuat web aplikasi khususnya pada bidang data sains. Menggunakan *streamlit* dapat mengubah skrip kedalam aplikasi web yang dapat dibagikan secara online. Setelah membuat web aplikasi pengguna dapat menggunakan *streamlit cloud* untuk melakukan *deployment* [22].

Streamlit framework memiliki kelebihan *simplicity* dengan adanya *components* yang sudah disiapkan sehingga memudahkan penggunaan *user*. Selain itu *streamlit* juga menggunakan *syntax python* sehingga semakin familiar bagi *developer*. *Streamlit* cocok digunakan untuk membangun *prototype* aplikasi, *dashboard* interaktif, visualisasi data, dan berbagai jenis aplikasi web sederhana [22].

2.2.6. SPSS

SPPS merupakan salah satu dari sekian banyak software berfungsi untuk melakukan perbandingan analisis data, SPSS dapat berfungsi untuk menganalisis suatu data. Sehingga data kusioner yang disebarkan dan diisi oleh responden dapat diketahui apakah pertanyaan yang diajukan reliabel dan valid. SPSS berguna dalam menyusun laporan penelitian, analisis data, pengambilan keputusan berdasarkan data, dan membantu pemahaman lebih mendalam tentang data yang ada. Dengan antarmuka yang intuitif dan fitur analisis yang kuat, SPSS dapat digunakan untuk peneliti, analis data, dan profesional di berbagai bidang untuk melakukan analisis statistik dan eksplorasi data secara efisien [23].

2.2.7. Skala *Guttman*

Skala guttman diciptakan oleh Louis Guttman pada tahun 1944, skala ini memiliki kelebihan dengan untuk memprediksi respons terhadap seluruh pertanyaan pada skala dan membuat kusioner yang singkat. Skala Guttman bersifat secara langsung dan jawaban hanya bersifat ya dan tidak. Prinsip dasar dari skala Guttman adalah bahwa jika seseorang setuju dengan pernyataan yang lebih kuat atau kategori lebih tinggi, maka dia juga harus setuju dengan pernyataan atau kategori yang lebih lemah atau lebih rendah. Dengan kata lain, jika seseorang telah menyetujui semua pernyataan dalam skala, maka dapat diasumsikan bahwa dia setuju dengan semua pernyataan di bawahnya dalam hierarki tersebut [24].

2.2.8. Validitas Data dan Reliabilitas Data

Uji validitas dan reliabilitas kuesioner ditunjukan untuk mengetahui sejauh mana data tersebut dapat dipercaya, uji validitas dapat mengukur sejauh mana akurasi suatu variabel. Indikator dalam kuesioner dapat dinyatakan valid apabila nilai r hitung lebih besar dari r tabel (df = N - 2). Pengujian validitas dapat dilakukan melalui 2 metode, pertama adalah dengan mengkorelasikan antara data dari indikator pertanyaan (item) dengan total item, kedua, mengkorelasikan antar masing-masing data indikator dengan total data secara keseluruhan [25].

Reliabilitas adalah nilai yang menunjukan sejauh mana alat pengukur dapat diandalkan, sehingga jika dilakukan pengukuran yang berulang maka pengukur tersebut dapat menghasilkan data yang konsisten. Biasanya sebelum dilakukan uji reliabilitas dilakukan pengujian validitas terlebih dahulu hal dengan tujuan untuk mengetahui validitas masing masing data, sebaliknya jika data yang diukur tidak valid maka tidak perlu dilakukan uji reliabilitas data [25].