BAB 2
LANDASAN TEORI

2.1 Pembelajaran Bahasa Mandarin

Pembelajaran bahasa Mandarin merupakan proses penguasaan kemampuan berbahasa yang mencakup aspek kosakata, tata bahasa, keterampilan membaca, menulis, mendengarkan, dan berbicara. Dalam konteks bahasa asing, pembelajaran bahasa Mandarin memiliki karakteristik yang unik karena melibatkan penguasaan sistem hanzi, pinyin, serta penggunaan nada yang memengaruhi makna kata [26].

HSK 2.0 (*Hanyu Shuiping Kaoshi*) merupakan standar kemampuan bahasa Mandarin yang diterbitkan oleh Hanban dan terdiri atas enam tingkatan, yaitu HSK 1 hingga HSK 6. Setiap tingkatan memiliki jumlah kosakata yang harus dikuasai oleh pembelajar. HSK 1 terdiri dari 150 kosakata, HSK 2 sebanyak 300 kosakata kumulatif, HSK 3 sebanyak 600 kosakata kumulatif, HSK 4 sebanyak 1.200 kosakata kumulatif, HSK 5 sebanyak 2.500 kosakata kumulatif, dan HSK 6 sebanyak 5.000 kosakata kumulatif. Standar HSK 2.0 tersebut digunakan sebagai acuan dalam penyusunan materi pembelajaran kosakata pada penelitian ini [27].

2.2 Digital Flashcard

Digital *flashcard* merupakan media pembelajaran yang menyajikan materi dalam bentuk potongan informasi (*flashcards*) yang ditampilkan secara berulang kepada pengguna sesuai jadwal tertentu untuk membantu proses pembelajaran dan meningkatkan retensi memori. Setiap *flashcard* umumnya berisi informasi singkat, seperti kosakata, definisi, atau pasangan pertanyaan dan jawaban, yang dipelajari melalui proses pengulangan secara bertahap menggunakan algoritma *spaced repetition* sehingga materi dapat diingat dalam jangka waktu yang lebih lama [21].

2.2.1 Pengambilan dan Pemilihan Jumlah Flashcard

Jumlah *flashcard* yang dipelajari dalam setiap sesi perlu disesuaikan agar tidak membebani kapasitas belajar pengguna sekaligus tetap efektif dalam meningkatkan penguasaan kosakata. Rata-rata jumlah kosakata baru yang disarankan berada pada kisaran 8–12 kata dalam setiap sesi pembelajaran, sehingga

proses akuisisi kosakata dapat berlangsung secara optimal [28]. Selain itu, pemilihan kosakata juga mempertimbangkan tingkat frekuensi kemunculan kata, karena kosakata yang lebih sering digunakan memiliki peluang lebih besar untuk ditemui dan dipelajari lebih awal oleh pembelajar bahasa kedua [28]. Oleh karena itu, pada penelitian ini jumlah *flashcard* baru yang diberikan kepada pengguna dibatasi pada rentang tersebut serta diprioritaskan berdasarkan frekuensi kemunculan kata. [28, 29].

2.2.2 Normalized Gain (N-Gain)

Normalized Gain (N-Gain) merupakan metode yang digunakan untuk mengukur besarnya peningkatan hasil pembelajaran dengan membandingkan peningkatan aktual yang dicapai terhadap peningkatan maksimum yang masih mungkin diperoleh. Metode ini tidak hanya memperhitungkan selisih antara nilai sebelum dan sesudah pembelajaran, tetapi juga mempertimbangkan peluang peningkatan yang masih tersedia sehingga dapat digunakan untuk mengevaluasi efektivitas suatu intervensi pembelajaran. Selain digunakan untuk menghitung rata-rata peningkatan suatu kelompok, *Normalized Gain* juga dapat diterapkan pada setiap individu (*single-student normalized gain*) untuk mengidentifikasi variasi peningkatan hasil belajar antar peserta [30, 31].

$$g_i = \frac{Post_i - Pre_i}{1 - Pre_i} \tag{2.1}$$

dengan keterangan sebagai berikut:

- g_i = nilai *Normalized Gain* individu,
- Pre_i = nilai sebelum intervensi (*pre-test*),
- $Post_i$ = nilai setelah intervensi (*post-test*).

Nilai *Normalized Gain* kemudian diinterpretasikan menjadi tiga kategori untuk menunjukkan tingkat peningkatan hasil pembelajaran. Menurut Hake, nilai $g < 0,30$ termasuk kategori *Low Gain*, nilai $0,30 \leq g < 0,70$ termasuk kategori *Medium Gain*, sedangkan nilai $g \geq 0,70$ termasuk kategori *High Gain*. Klasifikasi tersebut digunakan untuk menginterpretasikan besarnya peningkatan hasil belajar setelah suatu intervensi dilakukan [32].

Tabel 2.1. Kategori Normalized Gain

Nilai N-Gain	Kategori
$g < 0,30$	Low Gain
$0,30 \leq g < 0,70$	Medium Gain
$g \geq 0,70$	High Gain

2.3 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) merupakan salah satu jenis *Recurrent Neural Network* (RNN) yang dirancang untuk mengatasi permasalahan *gradient vanishing* dan *gradient explosion* yang sering terjadi pada proses pelatihan RNN tradisional. GRU menggunakan mekanisme *gating* dan kemampuan penyimpanan keadaan (*state memory*) untuk mengatur aliran informasi yang relevan sepanjang urutan data. GRU memiliki struktur yang menggabungkan *cell state* dan *gate structure* dalam satu mekanisme sehingga jumlah parameter yang digunakan lebih sedikit dibandingkan Long Short-Term Memory (LSTM). Dengan kompleksitas yang lebih rendah, GRU tetap mampu mempertahankan informasi jangka panjang dan menghasilkan performa yang baik dalam pemrosesan data sekuensial maupun deret waktu (*time series*) [33].

A Update Gate

Proses pertama pada GRU adalah menghitung nilai *update gate*. *Update gate* berfungsi untuk menentukan seberapa besar informasi dari *hidden state* sebelumnya akan dipertahankan dan diteruskan ke *hidden state* saat ini. Nilai *update gate* dihitung menggunakan Persamaan 2.2.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{2.2}$$

dengan:

- z_t = nilai *update gate* pada waktu ke-t
- σ = fungsi aktivasi sigmoid
- x_t = input pada waktu ke-t

- h_{t-1} = *hidden state* pada waktu sebelumnya
- W_z = matriks bobot input untuk *update gate*
- U_z = matriks bobot *hidden state* untuk *update gate*
- b_z = nilai bias pada *update gate*

Nilai z_t berada pada rentang 0 hingga 1. Semakin besar nilai z_t, semakin banyak informasi dari kondisi sebelumnya yang dipertahankan [34].

B Reset Gate

Setelah memperoleh nilai *update gate*, GRU menghitung *reset gate*. *Reset gate* berfungsi untuk menentukan seberapa besar informasi dari *hidden state* sebelumnya akan diabaikan ketika membentuk memori baru. Persamaan *reset gate* ditunjukkan pada Persamaan 2.3.

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{2.3}$$

dengan:

- r_t = nilai *reset gate* pada waktu ke-*t*
- σ = fungsi aktivasi sigmoid
- x_t = input pada waktu ke-*t*
- h_{t-1} = *hidden state* sebelumnya
- W_r = matriks bobot input untuk *reset gate*
- U_r = matriks bobot *hidden state* untuk *reset gate*
- b_r = nilai bias pada *reset gate*

Nilai *reset gate* menentukan seberapa besar kontribusi informasi masa lalu yang akan digunakan dalam pembentukan kandidat memori baru [34].

C Candidate Hidden State

Setelah nilai *reset gate* diperoleh, GRU membentuk *candidate hidden state*. Komponen ini merepresentasikan informasi baru yang akan dipertimbangkan untuk dimasukkan ke dalam memori jaringan. Persamaan *candidate hidden state* ditunjukkan pada Persamaan 2.4.

$$\tilde{h} * t = \tanh\left(W_h x_t + U_h(r_t \odot h * t - 1) + b_h\right) \tag{2.4}$$

dengan:

- $\tilde{h} * t$ = *candidate hidden state*
- tanh = fungsi aktivasi hiperbolik tangent
- W_h = matriks bobot input
- U_h = matriks bobot *hidden state*
- b_h = nilai bias
- r_t = nilai *reset gate*
- $h * t - 1$ = *hidden state* sebelumnya
- $\odot$ = operasi perkalian elemen demi elemen (*element-wise multiplication*)

Pada tahap ini, informasi dari *hidden state* sebelumnya terlebih dahulu difilter oleh *reset gate* sebelum digunakan untuk menghasilkan kandidat memori baru [34].

D Hidden State

Tahap terakhir pada GRU adalah memperbarui *hidden state*. Nilai *hidden state* baru diperoleh dengan menggabungkan informasi dari *hidden state* sebelumnya dan *candidate hidden state* menggunakan *update gate*. Persamaan pembaruan *hidden state* ditunjukkan pada Persamaan 2.5.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.5}$$

dengan:

- h_t = *hidden state* pada waktu ke-t
- h_{t-1} = *hidden state* pada waktu sebelumnya
- z_t = nilai *update gate*
- $\tilde{h}_t$ = *candidate hidden state*
- $\odot$ = operasi perkalian elemen demi elemen

Berdasarkan Persamaan 2.5, nilai *update gate* menentukan proporsi informasi lama dan informasi baru yang akan digunakan untuk membentuk *hidden state* akhir. Mekanisme tersebut memungkinkan GRU mempertahankan informasi penting dari masa lalu sekaligus memasukkan informasi baru yang relevan sehingga efektif digunakan untuk memodelkan data sekuensial [34].

2.3.1 Kelebihan dan Keterbatasan GRU

GRU memiliki beberapa keunggulan dibandingkan RNN konvensional maupun LSTM. Melalui mekanisme *update gate* dan *reset gate*, GRU mampu mempertahankan informasi penting dari urutan data sehingga mengurangi permasalahan *vanishing gradient*. Selain itu, GRU memiliki struktur yang lebih sederhana dibandingkan LSTM karena tidak menggunakan *cell state* dan *output gate*. Jumlah parameter yang lebih sedikit menyebabkan proses pelatihan menjadi lebih cepat, kebutuhan memori lebih rendah, serta tetap mampu menghasilkan performa yang kompetitif pada berbagai tugas pemodelan data sekuensial [33].

Di sisi lain, penyederhanaan arsitektur tersebut juga menjadi salah satu keterbatasan GRU. Tidak adanya *cell state* menyebabkan kapasitas GRU dalam menyimpan ketergantungan jangka sangat panjang lebih terbatas dibandingkan LSTM. Pada permasalahan yang memiliki urutan data sangat panjang atau hubungan temporal yang sangat kompleks, LSTM dapat memberikan performa yang lebih baik karena mampu mempertahankan informasi dalam jangka waktu yang lebih lama. Oleh karena itu, pemilihan antara GRU dan LSTM perlu disesuaikan dengan karakteristik data dan kebutuhan aplikasi [33].

2.3.2 Alasan pemilihan GRU dibanding LSTM

Pada penelitian ini, GRU dipilih sebagai arsitektur utama karena mampu mempelajari pola perubahan memori pengguna dari riwayat *review* secara efisien. Riwayat pembelajaran yang digunakan berupa urutan interval *review*, hasil *recall*, dan probabilitas *recall* yang membentuk data sekuensial dengan panjang yang bervariasi. Karakteristik tersebut dapat dimodelkan dengan baik menggunakan GRU tanpa kompleksitas parameter sebesar LSTM. Selain itu, penelitian GRU-HLR yang menjadi acuan penelitian ini juga menggunakan GRU untuk menangkap dinamika memori pengguna dan menunjukkan bahwa penggunaan fitur deret waktu (*time-series*) yang dipelajari oleh GRU mampu meningkatkan akurasi prediksi dibandingkan model HLR konvensional [25]. :contentReference[oaicite:0]index=0

2.3.3 Half-Life Regression (HLR)

Half-Life Regression (HLR) merupakan model *spaced repetition* berbasis *machine learning* yang digunakan untuk memperkirakan retensi memori pengguna terhadap suatu informasi [35]. HLR memodelkan kekuatan memori menggunakan nilai *half-life*, yaitu waktu yang diperlukan hingga probabilitas mengingat suatu informasi berkurang setengahnya. Estimasi *half-life* dihitung menggunakan:

$$\hat{h} = 2^{\Theta \cdot x} \tag{2.6}$$

dengan:

- $\hat{h}$ = estimasi *half-life*,
- Θ = parameter model yang dipelajari selama proses pelatihan,
- x = vektor fitur pembelajaran pengguna,
- $\Theta \cdot x$ = hasil perkalian parameter model dan fitur masukan.

$$p = 2^{-\frac{\Delta}{\hat{h}}} \tag{2.7}$$

- p = probabilitas pengguna mengingat suatu informasi,

- Δ = interval waktu sejak peninjauan terakhir,
- $\hat{h}$ = estimasi *half-life*.

dengan p menyatakan probabilitas mengingat dan Δ menyatakan interval waktu sejak peninjauan terakhir dilakukan [35]. Pendekatan ini memungkinkan sistem menghasilkan jadwal pengulangan yang lebih adaptif berdasarkan riwayat pembelajaran pengguna.

2.3.4 GRU-HLR

GRU-HLR (*Gated Recurrent Unit Half-Life Regression*) merupakan pengembangan dari model Half-Life Regression (HLR) dengan memanfaatkan jaringan GRU untuk mempelajari pola perubahan memori dari riwayat aktivitas belajar secara berurutan. Model ini menggunakan riwayat interval peninjauan, hasil recall, dan probabilitas recall untuk mengestimasi nilai *half-life* yang kemudian digunakan dalam perhitungan probabilitas recall. [25]

Estimasi *half-life* pada GRU-HLR dirumuskan sebagai berikut:

$$\hat{h}_i = GRU(\Delta t_{1:i-1}, r_{1:i-1}, p_{1:i-1}) \tag{2.8}$$

di mana $\hat{h}_i$ adalah estimasi *half-life*, $\Delta t_{1:i-1}$ merupakan riwayat interval antar review, $r_{1:i-1}$ adalah riwayat hasil recall, dan $p_{1:i-1}$ adalah riwayat probabilitas recall. [25]

Selain mengestimasi *half-life*, GRU-HLR juga memodelkan perubahan keadaan memori pengguna melalui *hidden state* sebagai berikut:

$$h_i, s_i = GRU\text{-}HLR(s_{i-1}, \Delta t_{i-1}, r_{i-1}, p_{i-1}) \tag{2.9}$$

dengan s_i sebagai *hidden state* yang merepresentasikan kondisi memori pengguna pada proses pembelajaran. Melalui mekanisme ini, GRU-HLR mampu menangkap dinamika memori secara berkelanjutan dan menghasilkan estimasi *half-life* yang lebih adaptif dibandingkan pendekatan HLR konvensional. [25]

2.3.5 Kelebihan GRU-HLR

GRU-HLR memiliki keunggulan dalam memodelkan dinamika memori pengguna berdasarkan riwayat aktivitas belajar yang bersifat sekuensial. Berbeda dengan Half-Life Regression (HLR) konvensional yang hanya memanfaatkan fitur statistik, GRU-HLR memanfaatkan jaringan *Gated Recurrent Unit* (GRU) untuk mempelajari pola perubahan memori dari urutan interval peninjauan, hasil *recall*, dan probabilitas *recall*. Pendekatan ini memungkinkan model menangkap hubungan temporal antaraktivitas belajar secara lebih efektif sehingga estimasi *half-life* dan probabilitas *recall* menjadi lebih akurat. Selain itu, penggunaan *hidden state* pada GRU memungkinkan representasi kondisi memori pengguna diperbarui secara berkelanjutan setelah setiap sesi peninjauan, sehingga model lebih adaptif terhadap perubahan kemampuan mengingat pengguna dibandingkan pendekatan HLR konvensional [25].

2.3.6 Evaluasi Model

Evaluasi model merupakan tahapan yang dilakukan untuk mengukur kemampuan model dalam menghasilkan prediksi yang akurat terhadap data yang belum pernah digunakan selama proses pelatihan. Pada penelitian ini, model GRU-HLR memprediksi nilai *half-life* yang bersifat kontinu sehingga evaluasi dilakukan menggunakan metrik regresi. Metrik yang digunakan adalah *Mean Absolute Error* (MAE) dan *Mean Absolute Percentage Error* (MAPE) karena keduanya mampu mengukur besarnya kesalahan prediksi baik dalam nilai absolut maupun dalam bentuk persentase. Nilai MAE dan MAPE yang lebih kecil menunjukkan bahwa hasil prediksi model semakin mendekati nilai sebenarnya [36, 37].

A Mean Absolute Error (MAE)

Mean Absolute Error (MAE) merupakan metrik evaluasi yang menghitung rata-rata selisih absolut antara nilai prediksi dan nilai sebenarnya. Metrik ini mempertahankan satuan yang sama dengan variabel target sehingga mudah diinterpretasikan sebagai besarnya kesalahan prediksi rata-rata. Semakin kecil nilai MAE, semakin baik performa model dalam melakukan prediksi [36].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.10}$$

dengan:

- MAE = nilai *Mean Absolute Error*,
- y_i = nilai sebenarnya (*actual value*),
- $\hat{y}_i$ = nilai hasil prediksi model,
- n = jumlah sampel pada data pengujian.

B Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) merupakan metrik evaluasi yang mengukur rata-rata persentase kesalahan antara nilai prediksi dan nilai sebenarnya. Berbeda dengan MAE, MAPE menghasilkan nilai dalam bentuk persentase sehingga memudahkan interpretasi tingkat kesalahan model tanpa dipengaruhi oleh skala data. Semakin kecil nilai MAPE, semakin tinggi tingkat akurasi prediksi model. Namun, metrik ini kurang sesuai digunakan apabila nilai aktual mendekati nol karena dapat menghasilkan nilai kesalahan yang sangat besar [36, 37].

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{2.11}$$

dengan:

- $MAPE$ = nilai *Mean Absolute Percentage Error*,
- y_i = nilai sebenarnya (*actual value*),
- $\hat{y}_i$ = nilai hasil prediksi model,
- n = jumlah sampel pada data pengujian.

2.4 Next.js

Next.js merupakan *framework* berbasis React yang digunakan untuk membangun aplikasi web dengan dukungan *Server-Side Rendering* (SSR) dan *Static Site Generation* (SSG). Framework ini menyediakan konfigurasi yang ringan dan fleksibel, mendukung *automatic routing* berbasis direktori, serta memiliki

berbagai fitur bawaan seperti *automatic code splitting*, *Client-Side Rendering* (CSR), dukungan CSS dan Sass, serta integrasi dengan berbagai *plugin* sehingga mempermudah proses pengembangan aplikasi web [38]. Selain itu, Next.js menggabungkan keunggulan SSR dan CSR melalui teknik seperti *server-side rendering*, *pre-fetching*, *incremental static regeneration*, dan *static site generation* untuk meningkatkan performa aplikasi serta optimasi *Search Engine Optimization* (SEO) [39]. Hasil penelitian juga menunjukkan bahwa Next.js memberikan performa yang lebih baik dibandingkan React.js tanpa menurunkan pengalaman interaksi pengguna, sehingga menjadi salah satu *framework* yang sesuai untuk pengembangan aplikasi web modern [39].

2.5 Black Box Testing

Black box testing adalah teknik pengujian perangkat lunak yang digunakan untuk memverifikasi apakah fungsi-fungsi yang tersedia pada suatu sistem telah bekerja sesuai dengan kebutuhan yang ditetapkan. Pada metode ini, penguji tidak perlu mengetahui logika program, struktur basis data, maupun kode sumber yang digunakan dalam pengembangan sistem. Proses pengujian dilakukan dengan memberikan berbagai skenario masukan dan kemudian mengamati respons atau keluaran yang dihasilkan oleh sistem. Dengan pendekatan tersebut, pengujian dapat digunakan untuk menilai kesesuaian perilaku sistem terhadap spesifikasi fungsional serta memastikan bahwa fitur yang tersedia dapat digunakan sebagaimana mestinya oleh pengguna [40].

2.5.1 Pengujian Sistem Menggunakan Metode Black Box

Pada penelitian ini, *black box testing* digunakan untuk mengevaluasi fungsi-fungsi yang tersedia pada website pembelajaran kosakata bahasa Mandarin. Pengujian dilakukan dengan menjalankan berbagai skenario penggunaan sistem dan kemudian memeriksa apakah keluaran yang dihasilkan telah sesuai dengan perilaku yang diharapkan. Pendekatan ini dipilih karena mampu memvalidasi kebutuhan fungsional sistem secara langsung dari perspektif pengguna tanpa harus menganalisis kode program yang digunakan. Melalui pengujian tersebut, setiap fitur dapat diperiksa untuk memastikan bahwa proses yang terjadi pada sistem berjalan dengan benar dan mampu mendukung aktivitas pembelajaran pengguna [41].

Fitur-fitur yang menjadi objek pengujian dalam penelitian ini meliputi:

1. **Manajemen Pengguna** Pengujian dilakukan pada fitur autentikasi dan pengelolaan akun, meliputi proses registrasi, login, logout, serta perubahan informasi profil pengguna.

2. **Pembelajaran Kosakata** Pengujian dilakukan pada fitur pembelajaran menggunakan *flashcard*, termasuk proses menampilkan kosakata, perpindahan kartu, validasi jawaban, dan pencatatan hasil pembelajaran pengguna.

3. **Review dan Prediksi Retensi** Pengujian dilakukan untuk memastikan sistem mampu menghasilkan jadwal *review* secara otomatis serta menampilkan hasil prediksi retensi memori yang diperoleh dari model GRU-HLR berdasarkan riwayat pembelajaran pengguna.

2.6 End-User Computing Satisfaction (EUCS)

End-User Computing Satisfaction (EUCS) merupakan instrumen evaluasi yang digunakan untuk menilai tingkat kepuasan pengguna terhadap suatu sistem informasi. Metode ini dikembangkan oleh Doll dan Torkzadeh sebagai pendekatan untuk mengukur keberhasilan implementasi sistem berdasarkan persepsi pengguna yang berinteraksi secara langsung dengan sistem tersebut. Melalui EUCS, peneliti dapat mengetahui bagaimana pengguna menilai kualitas sistem yang digunakan, sehingga hasil pengukuran dapat menjadi indikator apakah sistem telah mampu memenuhi kebutuhan dan ekspektasi pengguna [42].Jumlah responden pada penelitian ini mengacu pada pendapat Sugiyono[43], yang menyatakan bahwa jumlah sampel yang layak dalam suatu penelitian berkisar antara 30 hingga 500 responden.

Penilaian dalam metode EUCS dilakukan melalui lima dimensi yang merepresentasikan berbagai aspek kualitas sistem, yaitu [42]:

- **Content** (*Konten*), yaitu aspek yang menilai relevansi, kelengkapan, dan kegunaan informasi yang disajikan oleh sistem kepada pengguna.

- **Accuracy** (*Akurasi*), yaitu aspek yang menilai tingkat ketepatan informasi, data, maupun hasil yang dihasilkan oleh sistem.

- **Format** (*Format*), yaitu aspek yang berkaitan dengan tampilan antarmuka, keterbacaan, dan cara sistem menyajikan informasi kepada pengguna.

- **Ease of Use** (*Kemudahan Penggunaan*), yaitu aspek yang menilai kemudahan pengguna dalam mempelajari, memahami, dan mengoperasikan sistem.
- **Timeliness** (*Ketepatan Waktu*), yaitu aspek yang menilai kemampuan sistem dalam menyediakan informasi dan memberikan respons secara cepat ketika digunakan.

2.6.1 Penggunaan Skala Likert

Pengukuran kepuasan pengguna pada penelitian ini menggunakan Skala Likert 5 poin pada kuesioner EUCS. Skala Likert digunakan untuk mengubah persepsi dan tingkat persetujuan responden menjadi data numerik yang dapat dianalisis secara kuantitatif [44]. Rentang penilaian yang digunakan ditunjukkan pada Tabel 2.2.

Tabel 2.2. Skala Likert Penelitian

Nilai	Tingkat Persetujuan
1	Sangat Tidak Setuju
2	Tidak Setuju
3	Netral
4	Setuju
5	Sangat Setuju

Nilai yang diperoleh dari setiap pernyataan kemudian diolah untuk mengukur tingkat kepuasan pengguna pada dimensi *content*, *accuracy*, *format*, *ease of use*, dan *timeliness* dalam metode EUCS.

2.6.2 Interpretasi Skor Skala Likert

Setelah diperoleh nilai persentase kepuasan dari hasil pengolahan kuesioner menggunakan Skala Likert, langkah selanjutnya adalah menginterpretasikan nilai tersebut ke dalam kategori tingkat kepuasan pengguna. Interpretasi dilakukan untuk memudahkan analisis terhadap tingkat kepuasan pengguna berdasarkan persentase skor yang diperoleh dari setiap dimensi yang diukur. Pada penelitian ini, kategori interpretasi persentase mengacu pada penelitian yang dilakukan oleh Suwandi dkk.

[45], yang mengelompokkan tingkat kepuasan ke dalam empat kategori, yaitu sangat tidak puas, kurang puas, puas, dan sangat puas.

Tabel 2.3 menunjukkan interval persentase yang digunakan untuk menginterpretasikan hasil pengukuran kepuasan pengguna.

Tabel 2.3. Interpretasi Persentase Skala Likert

Interval (%)	**Kategori**
0 – 24.9	Sangat Tidak Puas
25 – 49.9	Kurang Puas
50 – 74.9	Puas
75 – 100	Sangat Puas

Kategori interpretasi tersebut digunakan untuk menentukan tingkat kepuasan pengguna berdasarkan nilai persentase yang diperoleh dari setiap dimensi metode *End User Computing Satisfaction* (EUCS). Semakin tinggi persentase yang diperoleh, semakin tinggi pula tingkat kepuasan pengguna terhadap sistem yang dikembangkan.

2.6.3 Penelitian Terdahulu

Berbagai penelitian telah mengembangkan metode *spaced repetition* untuk meningkatkan retensi memori dan penguasaan kosakata pada pembelajaran bahasa. Perkembangan penelitian dimulai dari penggunaan algoritma heuristik seperti SuperMemo dan Leitner, kemudian berkembang menjadi model prediksi berbasis *machine learning* seperti Half-Life Regression (HLR), hingga pendekatan berbasis *deep learning* menggunakan *Gated Recurrent Unit Half-Life Regression* (GRU-HLR). Ringkasan penelitian terdahulu yang relevan disajikan pada Tabel 2.4.

Tabel 2.4. Ringkasan Penelitian Terdahulu dan Research Gap

No	Judul	Penulis	Kontribusi Penelitian	Research Gap
1	Investigating the Use of a Mobile Flashcard Application Rememba on the Vocabulary Development and Motivation of EFL Learners	Tugce Kose	Mengembangkan aplikasi *digital flashcard* berbasis *spaced repetition*.	Belum menerapkan model prediksi retensi memori seperti HLR maupun GRU-HLR sehingga jadwal *review* belum adaptif.
2	Rancang Bangun Spaced Repetition Software untuk Menghafal Huruf Jepang Menggunakan Algoritma SuperMemo-2 Berbasis iOS	Agustyan Hidayat	Mengembangkan aplikasi *flashcard* berbasis algoritma SuperMemo-2.	Masih menggunakan algoritma heuristik sehingga belum mempersonalisasi jadwal *review* berdasarkan kondisi memori pengguna.

No	Judul	Penulis	Kontribusi Penelitian	Research Gap
3	Mobile-assisted Academic Vocabulary Learning with Digital Flashcards: Exploring the Impacts on University Students' Self-regulatory Capacity	Tahereh Boroughani	Membuktikan efektivitas Quizlet terhadap penguasaan kosakata.	Berfokus pada efektivitas media belajar tanpa mengembangkan model prediksi retensi memori.
4	Optimizing Human Learning	Behzad Tabibian	Mengoptimalkan penjadwalan *spaced repetition*.	Pendekatan masih berbasis optimasi heuristik dan belum memanfaatkan model *deep learning*.
5	Unbounded Human Learning: Optimal Scheduling for Spaced Repetition	Siddharth Reddy	Mengembangkan algoritma penjadwalan optimal *spaced repetition*.	Belum memodelkan dinamika memori berdasarkan riwayat pembelajaran pengguna.

No	Judul	Penulis	Kontribusi Penelitian	Research Gap
6	A Trainable Spaced Repetition Model for Language Learning	Burr Settles	Memperkenalkan model Half-Life Regression (HLR).	HLR belum mampu mempelajari pola perubahan memori secara sekuensial sehingga akurasinya masih terbatas.
7	Adaptive Forgetting Curves for Spaced Repetition Language Learning	Ahmed Zaidi	Mengembangkan HLR berbasis *neural network*.	Belum menggunakan arsitektur recurrent untuk mempelajari riwayat belajar berbentuk *time-series*.
8	A Stochastic Shortest Path Algorithm for Optimizing Spaced Repetition Scheduling	Junyao Ye	Mengembangkan model GRU-HLR berbasis *time-series*.	Berfokus pada pengembangan algoritma penjadwalan dan belum diimplementasikan pada aplikasi pembelajaran kosakata.

No	Judul	Penulis	Kontribusi Penelitian	Research Gap
9	Optimizing Spaced Repetition Schedule by Capturing the Dynamics of Memory	Jingyong Su	Menyempurnakan model GRU-HLR sehingga meningkatkan akurasi prediksi retensi memori.	Belum mengintegrasikan model GRU-HLR ke dalam aplikasi pembelajaran kosakata bahasa Mandarin berbasis *digital flashcard*.