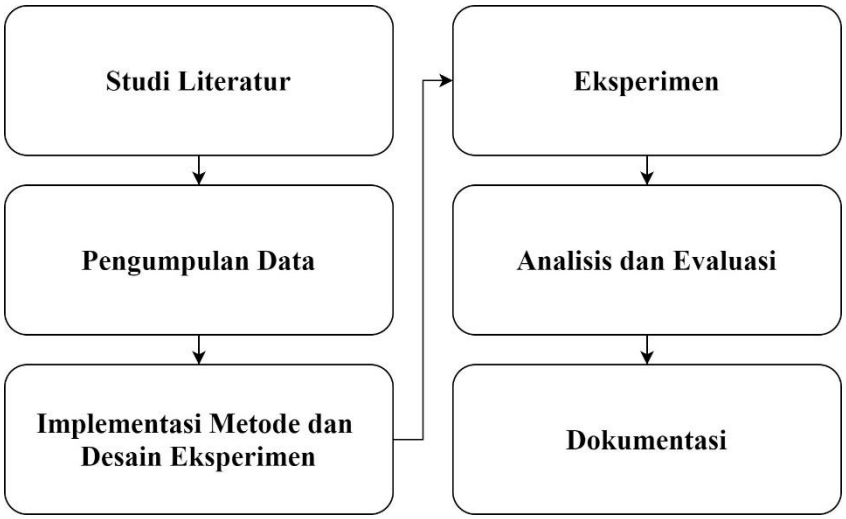
BAB III
METODOLOGI PENELITIAN

3.1 Desain Penelitian

Dalam subbab ini, penulis akan memaparkan langkah-langkah yang akan dilakukan dalam pelaksanaan penelitian ini. Langkah-langkah tersebut beserta penjelasan singkatnya diilustrasikan di Gambar 13.



Gambar 13. Desain penelitian

Bab ini dibagi menjadi beberapa subbab yang menjelaskan masing-masing tahap dari penelitian seperti yang digambarkan di atas, yaitu:

1) Studi Literatur
2) Pengumpulan Data
3) Implementasi dan Desain Eksperimen
4) Eksperimen
5) Analisis dan Evaluasi
6) Dokumentasi

3.1.1 Studi Literatur

Di tahap studi literatur, penulis melakukan survei mengenai materi serta metode yang diperlukan untuk mewujudkan penelitian ini. Materi yang pertama disurvei adalah mengenai karakteristik komentar yang dibuat peserta didik mata kuliah pemrograman. Penulis melakukan wawancara dengan dosen yang

mengampu mata kuliah pemrograman di Prodi Ilmu Komputer UPI. Penulis menemukan bahwa komentar peserta didik cenderung tidak memiliki struktur atau standar tertentu dan bersifat bebas sesuai kepahaman dan keinginan. Terdapat tiga kriteria utama yang ditetapkan oleh beliau sebagai syarat komentar yang diterima, yaitu (1) ada komentar janji di awal kode, (2) komentar ada di setiap fungsi kode, dan (3) komentar berhubungan dengan soal. Materi berikutnya adalah *online judge* yang akan menjadi lingkungan implementasi penelitian ini, yaitu CSPC. Materi berikutnya adalah pendekatan yang akan digunakan yaitu *corpus-based text similarity*. Setelah itu, ada materi *Word Mover's Distance* (WMD), yang merupakan perhitungan *text similarity* yang akan digunakan. Hal ini adalah untuk memastikan bahwa komentar yang dibuat peserta didik tetap relevan atau berhubungan dengan soal. Materi terakhir adalah koefisien korelasi, yang digunakan di tahap evaluasi untuk mengukur hubungan linear antara nilai *similarity* dengan nilai manusia.

3.1.2 Pengumpulan Data

Penelitian ini melibatkan data dari basis data CSPC, yang akan digunakan sebagai *input* utama dari penelitian ini. Data yang akan digunakan berbentuk *submission* atau jawaban yang dikirim peserta dalam bentuk *source code* untuk soal tertentu. Satu datum jawaban terdiri atas ID jawaban, soal, dan *source code*. Data yang digunakan berjumlah 500 baris dan disaring sehingga hanya dari rentang waktu 2017 – 2019. Karena peserta bisa mengirim lebih dari satu jawaban untuk satu soal, maka untuk menghindari redundansi, data yang akan digunakan adalah jawaban terakhir setiap peserta untuk setiap soal.

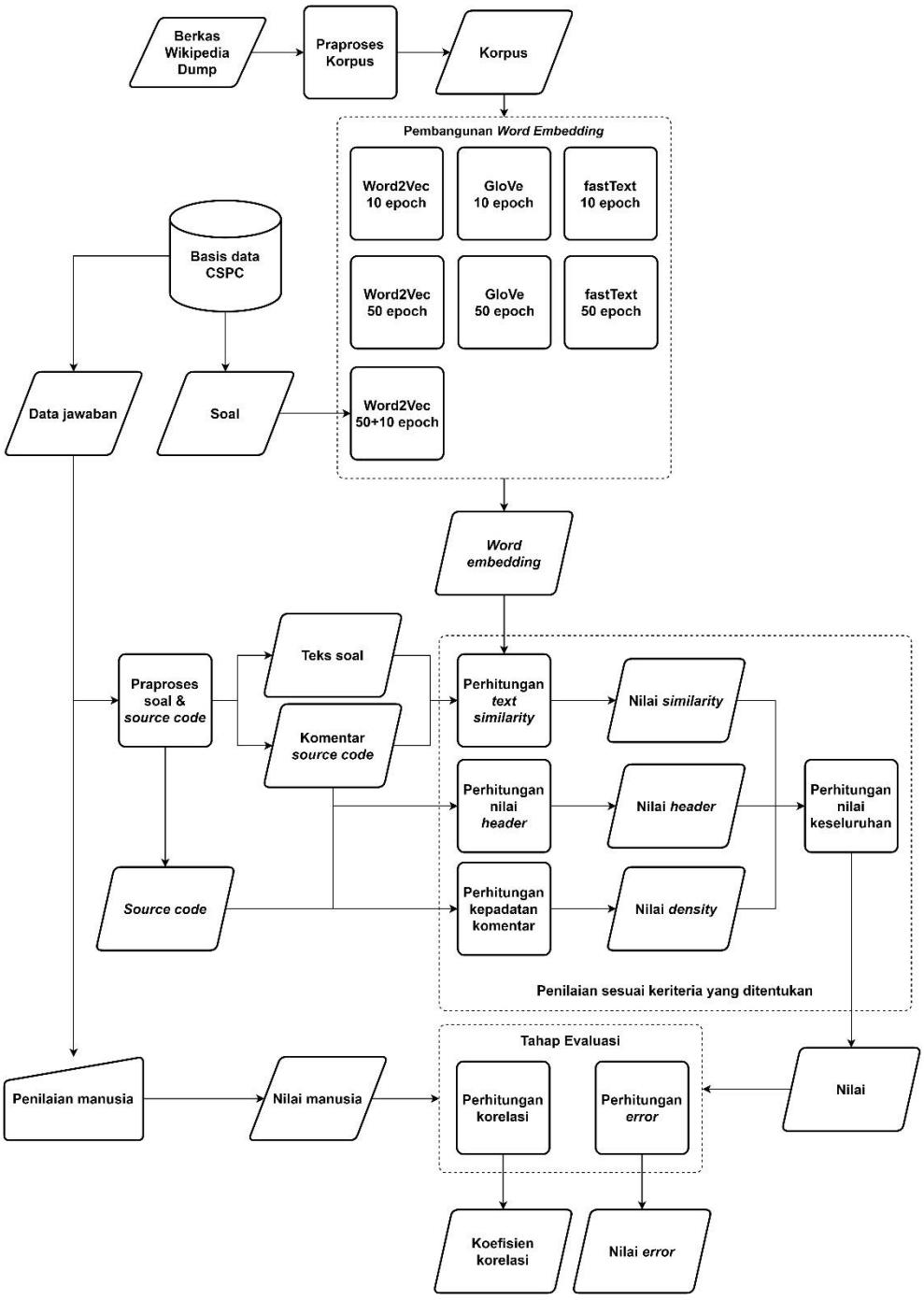
Data akan diambil menggunakan *mysql-connector-python*, yaitu sebuah *library* untuk Python yang dapat menghubungkan program dengan basis data *MySQL* (https://pypi.org/project/mysql-connector-python/). Data diambil dari beberapa tabel dan berikutnya akan disimpan di sebuah tabel baru untuk proses penilaian manusia.

Untuk penelitian ini, data jawaban perlu diberi nilai manusia. Hal ini dilakukan agar ada pembanding bagi nilai dari metode yang diusulkan penulis. Nilai manusia diberikan kepada 500 baris data jawaban oleh mahasiswa dan sudah

diverifikasi oleh dosen pengampu mata kuliah Algoritma dan Pemrograman. Peran nilai manusia di tahap evaluasi akan dijelaskan lebih lanjut di penjelasan perhitungan koefisien korelasi.

3.1.3 Implementasi dan Desain Eksperimen

Desain eksperimen penelitian ini diilustrasikan di Gambar 14.



Gambar 14. Alur eksperimen

Setiap proses dalam desain eksperimen mewakili sebuah fungsi yang akan diimplementasikan ke dalam bentuk program Python. Bagian ini akan menjelaskan masing-masing proses tersebut.

3.1.3.1 Praproses Korpus

Berkas Wikipedia Dump yang diunduh dari dumps.wikimedia.org memiliki format xml yang mengandung judul, teks, serta berbagai *metadata* dari semua artikel Wikipedia. Berkas ini perlu melewati praproses terlebih dahulu sebelum menjadi korpus yang digunakan untuk membangun *word embedding*. Praproses untuk berkas Wikipedia Dump terdiri dari beberapa tahap yaitu:

1) *Parsing*

Tahap ini dilakukan untuk mengambil hanya teks artikel-artikel, dan menyimpannya dalam format *txt* di mana setiap baris akan memuat satu artikel. *Parsing* akan dilakukan dengan bantuan program WikiExtractor dari repositori github.com/attardi/wikiextractor.

2) *Tokenization*

Tahap ini akan mengubah teks mentah menjadi potongan-potongan kata terpisah (*token*). Tahap ini juga akan menghilangkan semua kata yang lebih pendek dari 3 karakter (meliputi angka), mengubah semua huruf menjadi huruf kecil, serta melakukan *stop word removal*. Proses tersebut yang terakhir akan dijelaskan berikutnya.

3) *Stop word removal*

Di tahap ini, kata-kata yang sering muncul seperti kata sambung, kata ganti, dan sejenisnya, akan dihilangkan dari teks.

3.1.3.2 Pembuatan *Word Embedding*

Seperti yang dipaparkan di Gambar 14, penelitian ini akan menggunakan 3 model untuk membangun *word embedding* yaitu *Word2vec*, *GloVe*, dan *fastText*. Penjelasan pembuatan *word embedding* adalah sebagai berikut.

1) *Word2vec*

Library Gensim (https://pypi.org/project/gensim/) yang tersedia untuk Python sudah menyediakan fitur untuk membuat, memuat, serta mengolah *embedding* model *Word2vec*.

2) *GloVe*

Penelitian ini akan menggunakan repositori *GloVe* asli yang tersedia di GitHub (https://github.com/stanfordnlp/GloVe) untuk membangun *embedding* model *GloVe*. Program dalam repositori tersebut ditulis dengan bahasa C, namun sudah menyediakan parameter-parameter yang bisa diatur oleh pengguna. Hasil *word embedding* dari program tersebut dapat dimuat ke Python dengan *library Gensim*.

3) *FastText*

Penelitian ini akan membangun *embedding* model *fastText* menggunakan *library* resminya untuk Python (https://pypi.org/project/fasttext/). *Library Gensim* akan digunakan memuat dan mengolah model hasil *fastText.*

3.1.3.3 Praproses Soal dan *Source Code*

Data yang diambil dari basis data CSPC adalah *source code* beserta soal yang bersangkutan. Seperti yang dijelaskan sebelumnya, *source code* yang diambil adalah dari jawaban terakhir dari setiap peserta untuk setiap soal. Data *source code* disimpan dalam basis data dengan tipe data *blob* (*binary large object*). Sedangkan soal disimpan dengan tipe data *string* yang masih mengandung simbol-simbol HTML yang berguna saat menampilkan soal dalam halaman web CSPC. Semua data *source code* dan soal juga perlu melalui praproses sebelum bisa digunakan dalam eksperimen.

Praproses yang dilakukan kepada teks soal adalah sebagai berikut.

1) *HTML tag removal*

Praproses ini menghilangkan simbol-simbol HTML dari *string* soal dengan bantuan pola *regex*.

2) *Whitespace trimming*

Praproses ini menghilangkan karakter-karakter tidak terlihat dari teks seperti *newline* (\n), *tab* (\t), *carriage return* (\r), dan spasi berlebih di awal dan akhir tiap baris.

3) *Tokenization*

Praproses ini mengubah teks soal menjadi potongan-potongan kata (*token*). Penulis juga menghilangkan semua kata yang lebih pendek dari 3 karakter (meliputi

angka), mengubah semua huruf menjadi huruf kecil, serta melakukan *stop word removal*.

Praproses yang harus dilakukan kepada *source code* adalah sebagai berikut.

1) *Decoding*

Praproses ini dilakukan untuk mengubah *source code* yang awalnya bertipe data *blob* menjadi *string* yang bisa diolah lebih lanjut.

2) *Whitespace trimming*

Praproses ini dilakukan untuk menghapus karakter-karakter tidak terlihat (*whitespace*) dari *source code*, yaitu karakter *newline* (\n), *tab* (\t), *carriage return* (\r), dan spasi berlebih di awal dan akhir tiap baris.

3) *Comment extraction*

Praproses ini memisahkan semua baris komentar dari *source code*. Hasil dari praproses ini adalah semua komentar yang terpisah per baris.

4) *Tokenization*

Praproses *tokenization* yang dilakukan pada komentar sama dengan yang dilakukan pada praproses korpus dan soal. Komentar akan dipisah menjadi *token*, dihapuskan semua kata di dalamnya yang kurang dari 3 karakter, diubah menjadi huruf kecil, serta dilakukan *stop word removal*.

3.1.3.4 Perhitungan Nilai

Berdasarkan hasil wawancara dengan dosen pengampu mata kuliah Algoritma dan Pemrograman di Ilmu Komputer UPI, terdapat tiga kriteria penilaian terhadap komentar *source code*. Ketiga kriteria tersebut adalah:

a. terdapat komentar janji pada *header* (nilai *header*)
b. komentar menjelaskan tiap proses (nilai *density*)
c. komentar harus berhubungan dengan soal (nilai *similarity*)

Perhitungan nilai keseluruhan akan dilakukan dengan menggabungkan nilai tiga kriteria tersebut dengan bobot tertentu. Penulis akan melaksanakan eksperimen di sini dengan menggunakan beberapa variasi bobot penilaian untuk melihat perubahan nilai korelasi terhadap pencampuran nilai *similarity* dengan nilai-nilai lain. Penjelasan masing-masing nilai adalah sebagai berikut.

1) Nilai *header*

Nilai *header* diberikan sesuai keberadaan komentar janji di *header source code*. Cara perhitungan nilai ini dilakukan dengan mencocokkan komentar yang terdapat dalam *source code* dengan sebuah *template* komentar janji. Sebuah *source code* akan diberi nilai *header* 1.0 (dalam skala 0.0 – 1.0) bila semua kata dari *template* janji ada dalam komentar dalam urutan yang sama. Nilai akan berkurang untuk setiap kata dalam janji yang tidak ditulis. Cara ini juga dilakukan di penelitian sebelumnya (Sukamto dkk., 2019).

2) Nilai *density*

Nilai *density* diukur dengan menghitung kepadatan komentar relatif dengan *source code*. Semakin banyak komentar yang ditulis, maka nilai *density* yang didapat juga semakin tinggi. Perhitungan nilai *density* dilakukan dengan menghitung jumlah karakter dalam komentar dan dibandingkan dengan jumlah semua karakter dalam *source code* (setelah praproses *whitespace trimming*). Kepadatan dihitung di tingkat karakter untuk menghindari manipulasi penilaian oleh peserta dengan sengaja memisahkan komentar menjadi banyak baris. Rasio jumlah karakter komentar dengan jumlah karakter *source code* akan dijadikan nilai *density* dengan skala 0.0 – 1.0.

3) Nilai *similarity*

Nilai *similarity* menentukan koherensi antara komentar *source code* dan soal. Hal ini dilakukan dengan menghitung *text similarity* menggunakan WMD dan *word embedding* yang sudah dibuat. Nilai yang didapatkan dari tahap ini adalah besar kemiripan berupa persentase dalam skala 0.0 – 1.0.

3.1.3.5 Perhitungan Koefisien Korelasi dan *Error*

Koefisien korelasi digunakan untuk mengukur seberapa kuat hubungan linear antara data prediksi dengan data yang sebenarnya. Penulis akan menghitung koefisien korelasi antara data yang dihasilkan perhitungan *text similarity* dengan data nilai manusia. Hal ini dilakukan dengan asumsi bahwa naik turunnya data nilai hasil *similarity* harus berhubungan juga dengan naik turunnya data nilai manusia, dan kuatnya hubungan ini akan dijadikan evaluasi untuk metode yang diusulkan

penelitian ini. Selain koefisien korelasi, penulis juga akan menghitung *error* atau deviasi data hasil *similarity* terhadap data nilai manusia.

3.1.3.6 Pengukuran Performa

Pengukuran performa dilakukan untuk mencari tahu model mana yang paling efisien dalam penelitian ini. Metrik yang diukur adalah penggunaan CPU, penggunaan memori (RAM), serta lama eksekusi program.

3.1.4 Eksperimen

Seperti yang dipaparkan di Gambar 14, penelitian ini akan menggunakan beberapa jenis *word embedding*. Terdapat total 7 jenis *word embedding* yang akan digunakan, dari 3 model dengan variasi *epoch* berbeda untuk proses *training*. Penelitian ini menggunakan beberapa *epoch* untuk melihat apakah ada pengaruh dari lama *training* untuk membangun *word embedding* terhadap kualitas vektor yang dihasilkan. Bagus atau tidaknya akan ditentukan dari nilai kemiripan teks yang didapatkan. Untuk penjelasan ini, *word embedding* tersebut akan dipisah sesuai modelnya, yaitu sebagai berikut.

1) *Word2vec*

Model *Word2vec* akan membuat 3 *embedding* berbeda. *Word embedding* pertama dilakukan *training* dengan 10 *epoch*, lalu 50 *epoch*, dan terakhir 50+10 *epoch*. Untuk 50+10 *epoch*, *word embedding* yang sudah dilakukan *training* dengan korpus dari Wikipedia Dump selama 50 *epoch* akan dilanjutkan dengan menggunakan data soal CSPC selama 10 *epoch*. Sayangnya untuk saat ini, hanya model *Word2vec* yang sudah diimplementasi di Python dengan fitur pelanjutan *training*.

2) *GloVe*

Model *GloVe* akan digunakan untuk membangun 2 *embedding* berbeda dengan proses *training* masing-masing selama 10 *epoch* dan 50 *epoch*.

3) *FastText*

Model *fastText* akan digunakan untuk membangun 2 *embedding* berbeda dengan proses *training* masing-masing selama 10 *epoch* dan 50 *epoch.*

3.1.5 Analisis dan Evaluasi

Di tahap ini, penulis akan melakukan analisis dan evaluasi terhadap hasil eksperimen yang dilakukan. Aspek yang akan digunakan untuk evaluasi adalah nilai koefisien korelasi yang dihasilkan setiap *word embedding* serta pengukuran performa. Penulis juga akan menggunakan hasil perhitungan kemiripan teks dari penelitian sebelumnya oleh Sukamto dkk. untuk membandingkan hasil dari pendekatan statistik dan pendekatan *corpus-based text similarity*. Hasil dari evaluasi akan disajikan dalam bentuk tabel dan grafik.

3.1.6 Dokumentasi

Tahap ini adalah tahap terakhir dalam penelitian ini. Segala hasil dari penelitian ini akan didokumentasikan menjadi bentuk skripsi.

3.2 Perbandingan dengan Penelitian Terdahulu

Sebelumnya telah dijelaskan bahwa sudah ada sejumlah penelitian terdahulu yang merancang sistem penilaian komentar. Namun, tujuan mereka adalah mengimplementasikannya di lingkungan profesional, di mana metrik penilaian yang digunakan juga berbeda dengan penelitian ini. Penulis mengambil tiga contoh penelitian yang merancang penilaian komentar otomatis (Khamis dkk., 2010; Steidl dkk., 2013; Wang, 2019). Berbagai parameter akan dibandingkan antara metodologi penelitian ini dengan dua penelitian tersebut. Perbandingan tersebut dipaparkan di Tabel 2.

Tabel 2. Perbandingan metodologi penelitian ini dengan dua penelitian terdahulu

Parameter	**Sistem yang diusulkan**	**Khamis, Witte, & Rilling (2010)**	**Steidl dkk. (2013)**	**Wang dkk. (2019)**
Metrik penilaian	Kemiripan antara komentar dan soal	Kualitas bahasa Konsistensi antara *source*	Koherensi antara *source code* dan komentar	Koherensi antara *source code* dan komentar

Parameter	Sistem yang diusulkan	Khamis, Witte, & Rilling (2010)	Steidl dkk. (2013)	Wang dkk. (2019)
	Kepadatan komentar dalam *source code* Keberadaan komentar *header*	*code* dan komentar	Panjang komentar	
Metode	*Text similarity* dengan *word embedding* dan *word mover's distance*	Ontologi OWL yang dibangun berdasarkan metrik penilaian	Klasifikasi berdasarkan tujuh kategori	Data diberi label "*coherent*" atau "*not coherent*", lalu dibuat representasi vektor dengan B-LSTM dan *weighted GloVe* yang mengekstrak fitur-fitur penting, lalu dilakukan klasifikasi dengan *Multi-Layer Perceptron*
Data uji	Data jawaban CSPC yang terdiri atas soal dan *source code* jawaban	*Source code* aplikasi ArgoUML dan Eclipse	Mengambil berkas sampel secara acak dari 12 proyek *open source*	*Source code* dan komentar *method* program Java

Parameter	Sistem yang diusulkan	Khamis, Witte, & Rilling (2010)	Steidl dkk. (2013)	Wang dkk. (2019)
Dukungan bahasa	Bahasa Indonesia	Bahasa Inggris	Bahasa Inggris	Bahasa Inggris

Dari tabel di atas dapat disimpulkan bahwa penelitian terdahulu umumnya mengukur koherensi antara komentar dengan *source code* untuk menilai kualitas dari *source code*. Dari metrik penilaian, sudah terlihat perbedaan antara penelitian ini dengan penelitian terdahulu yang bertujuan menilai komentar *source code*.

3.3 Lingkungan Komputasi

Lingkungan komputasi penelitian ini adalah sebuah *laptop PC* dan beberapa perangkat lunak pendukung. Spesifikasi dari *laptop PC* yang digunakan dipaparkan di Tabel 3.

Tabel 3. Spesifikasi *laptop PC* sebagai perangkat keras

CPU	Intel Core i5-9300H
GPU	Intel UHD Graphics 630 (*display*) Nvidia GeForce GTX 1650 (*render*)
Memory	8GB DDR4
Storage	512 GB SSD
Aksesoris	Mouse dan Keyboard

Ada beberapa perangkat lunak yang digunakan untuk membantu penelitian ini. Perangkat lunak tersebut dipaparkan di Tabel 4.

Tabel 4. Perangkat lunak dan versi yang digunakan

Sistem operasi	Windows 10 20H2
Bahasa pemrograman	Python 3.7.7
Library	mysql-connector-python 8.0.21 nltk 3.5 gensim 3.8.3 fasttext 0.9.2 scipy 1.4.1 sklearn 0.23.1 psutil 5.7.2 pandas 1.0.4 matplotlib 3.2.0
IDE	PyCharm 2020.1.1
CLI	Windows PowerShell 5.1.19041.1