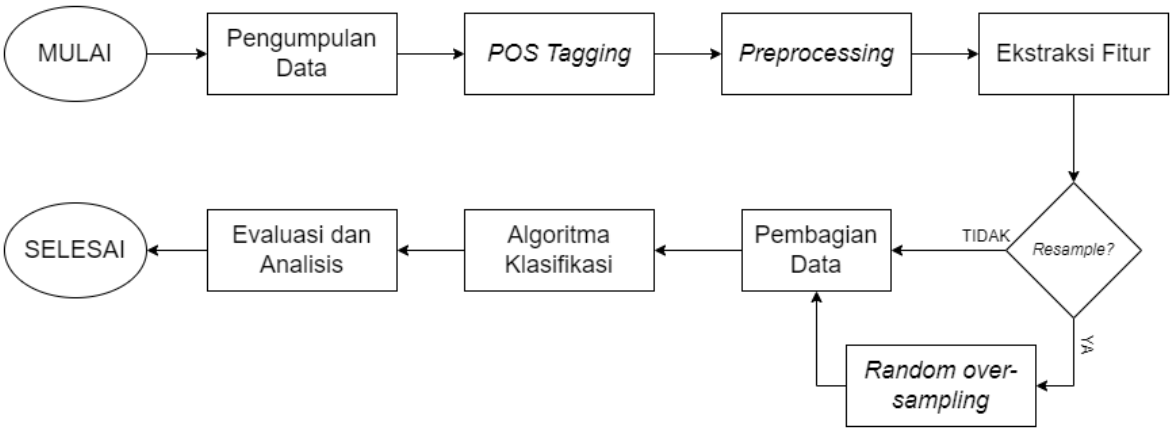
3. Sistem yang Dibangun

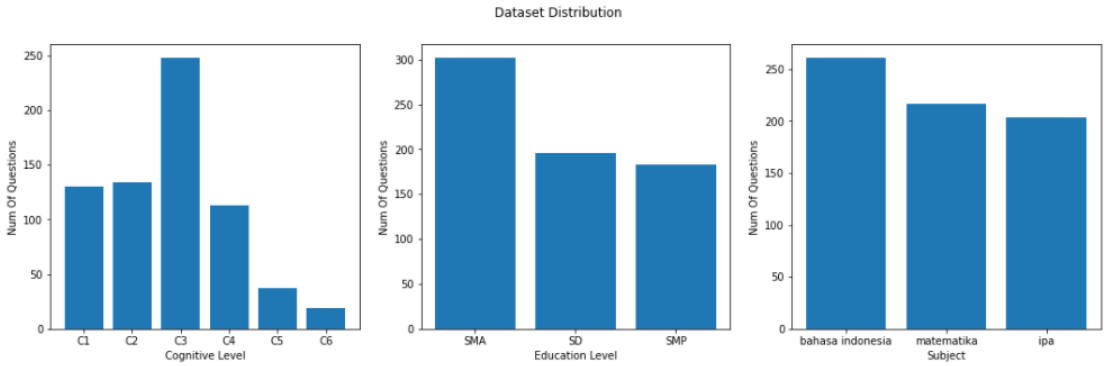
Sistem dibangun menggunakan bahasa pemrograman Python dengan alur seperti pada gambar 2.



Gambar 1 *Alur Kerja Sistem*

Pengumpulan Data

Dataset berupa teks latihan soal Berbahasa Indonesia dikumpulkan secara manual dari berbagai sumber daring seperti EduBox, Blog Ruangguru [14] dan penelitian oleh Syarifah et al. [15]. Dataset dilabeli secara manual berdasarkan tingkatan kognitif dalam Taksonomi Bloom. Data berhasil terkumpul sebanyak 682 soal dengan persebaran seperti pada gambar 3.



Gambar 2 *Distribusi Dataset*

POS Tagging

POS Tagging diimplementasikan menggunakan *library* FlairNLP [17]. Untuk Bahasa Indonesia, FlairNLP menyediakan corpus dan 2 *pre-trained word embedding* yang bersumber dari FastText Wikipedia dan *crawling* situs. Model dilatih menggunakan corpus dan kedua *pre-trained word embedding* dengan parameter learning_rate = 0.1, mini_batch_size = 32 dan max_epochs = 10. Skor yang dihasilkan model setelah dilatih tertera pada gambar 4.

```
Results:
- F-score (micro) 0.9251
- F-score (macro) 0.8646
- Accuracy 0.9251

By class:
              precision    recall  f1-score   support

        NOUN     0.8748    0.9152    0.8945      2511
       PROPN     0.9271    0.8996    0.9131      2162
       PUNCT     0.9982    1.0000    0.9991      1623
        VERB     0.9454    0.9342    0.9398      1261
         ADP     0.9334    0.9560    0.9446      1114
        PRON     0.9321    0.9798    0.9553       644
         ADJ     0.8614    0.7131    0.7803       488
         NUM     0.9352    0.9766    0.9554       384
       CCONJ     0.9783    0.9945    0.9863       362
         ADV     0.8433    0.7775    0.8090       346
         DET     0.9599    0.9120    0.9353       341
         AUX     0.9306    0.9956    0.9620       229
       SCONJ     0.8500    0.7887    0.8182       194
        PART     0.9149    0.9663    0.9399        89
         SYM     1.0000    1.0000    1.0000         6
           X     0.0000    0.0000    0.0000         2

   micro avg     0.9251    0.9251    0.9251     11756
   macro avg     0.8678    0.8631    0.8646     11756
weighted avg     0.9247    0.9251    0.9243     11756
 samples avg     0.9251    0.9251    0.9251     11756
```

Gambar 3 *Hasil Training Model untuk POSTagging*

Preprocessing

Preprocessing perlu dilakukan untuk memastikan dataset siap digunakan untuk pelatihan dan pengujian. Perangkat yang digunakan pada proses ini adalah Microsoft Excel dan bahasa pemrograman Python dengan *library* Scikit Learn [18]. Pada Microsoft Excel, *preprocessing* yang dilakukan adalah sebagai berikut:

1. Pemeriksaan ejaan kata pada setiap latihan soal.
2. Penghapusan spasi yang berjumlah lebih dari 1.

Sementara itu, *preprocessing* yang dilakukan dengan bahasa pemrograman Python adalah sebagai berikut:

1. *Case folding*

   Pada bagian ini, semua huruf kapital pada teks diubah menjadi huruf kecil.
2. Penghapusan tanda baca

Pada bagian ini, semua tanda baca pada teks dihapus.

3. Penghapusan *stopwords*

   Pada bagian ini, ada 2 daftar *stopwords* yang akan dijadikan acuan yaitu *stopwords* dari library PySastrawi (default), dan modifikasi dari PySastrawi. Modifikasi *stopwords* mengacu pada penelitian Mohammed et al. [12] yang mengatakan bahwa *stopwords* tertentu dapat memiliki dampak yang signifikan dalam menentukan tingkat kesulitan sebuah soal. Daftar *stopwords* yang dikecualikan dari *stopwords* PySastrawi dapat dilihat pada gambar 5. Proses ini akan menghasilkan data yang berbeda, dan akan dijadikan pembanding untuk menentukan mana yang lebih baik.

```
sw_keep = ['adalah', 'apa', 'arti', 'artinya', 'berapa', 'berapakah', 'beri',
           'berikan', 'diantaranya', 'disebut', 'jelaskan', 'karena',
           'mengapa', 'menunjukkan', 'merupakan', 'rupa', 'sebut']
```

Gambar 4 *Pengecualian Stopwords*

4. *Stemming*

   Mengubah kata ke dalam bentuk dasar dari kata tersebut.

Ekstraksi Fitur

Ekstraksi fitur dilakukan dengan metode TF-IDF reguler dan TFPOS-IDF. TF-IDF reguler diimplementasikan secara penuh menggunakan *library* Scikit Learn. Sementara itu, untuk TFPOS-IDF program akan dimodifikasi menyesuaikan dengan rumus (5) dan dinormalisasi menggunakan rumus (6). Proses ini akan menghasilkan data yang berbeda, dan akan dijadikan pembanding untuk menentukan mana yang lebih baik. Contoh hasil TF-IDF dan TFPOS-IDF dari dokumen nomor 6 pada dataset dapat dilihat pada Tabel 2.

	Teladan (*NOUN*)	Tokoh (*NOUN*)	Dasar (*ADP*)	Kutip (*NOUN*)
TF-IDF	0.402	0.521	0.638	0.402
TFPOS-IDF	0.689	0.563	0.145	0.434

Tabel 1 *Hasil Ekstraksi Fitur Pada Dokumen nomor 6*

*nilai dibulatkan ke atas.

Random OverSampling

Random Oversampling merupakan salah satu metode *resampling* yang bertujuan untuk mengurangi kesenjangan ukuran kelas pada dataset, sehingga algoritma klasifikasi tidak membias pada kelas mayoritas [20]. *Random oversampling* diimplementasikan menggunakan *library* Imbalanced-learn [19], dengan parameter sampling_strategy = ‘not majority’, dan random_state = 10. Mengacu pada distribusi data dari dataset yang digunakan, maka kelas yang bukan mayoritas, yaitu selain C3, akan diduplikasi hingga jumlahnya setara dengan kelas C3. Hasil *random oversampling* dapat dilihat pada Tabel 3.

Kelas	Sebelum *random oversampling*	Setelah *random oversampling*
C1	130	248
C2	134	248
C3	248	248
C4	113	248
C5	37	248
C6	19	248

Tabel 2 *Hasil Random Oversampling*

Pembagian Data

Menurut penelitian Gholamy et al. [21], pembagian data untuk pelatihan dan pengujian dengan rasio 80:20 merupakan rasio yang terbaik secara empiris. Maka dari itu, pembagian dataset untuk pelatihan dan pengujian pada tugas akhir ini dibagi dengan rasio 80:20 dan parameter random_state = 23 untuk hasil pembagian data yang konsisten pada setiap eksekusi. Data yang digunakan untuk pengujian berjumlah 8 data dengan spesifikasi yang berbeda antar data. Skenario pengujian pada Tugas Akhir ini dapat dilihat pada Tabel 4.

Skenario	Algoritma	Feature Extraction	Stopwords	Random Over-Sampling
1	SVM	TF-IDF	Default	N
	NB	TF-IDF	Default	N
2	SVM	TF-IDF	Modifikasi	N
	NB	TF-IDF	Modifikasi	N
3	SVM	TF-IDF	Default	Y
	NB	TF-IDF	Default	Y
4	SVM	TF-IDF	Modifikasi	Y
	NB	TF-IDF	Modifikasi	Y
5	SVM	TFPOS-IDF	Default	N
	NB	TFPOS-IDF	Default	N
6	SVM	TFPOS-IDF	Modifikasi	N
	NB	TFPOS-IDF	Modifikasi	N
7	SVM	TFPOS-IDF	Default	Y
	NB	TFPOS-IDF	Default	Y
8	SVM	TFPOS-IDF	Modifikasi	Y
	NB	TFPOS-IDF	Modifikasi	Y

Tabel 3 *Skenario Pengujian*

Algoritma Klasifikasi

Klasifikasi dilakukan dengan algoritma SVM dan NB. Kedua algoritma tersebut akan mengeksekusi beberapa skenario pengujian untuk menentukan spesifikasi yang terbaik pada dataset yang digunakan. Setiap algoritma selesai memberi prediksi dan mendapatkan skor, parameter algoritma tersebut akan dioptimasi menggunakan GridSearchCV dari library Scikit Learn dengan parameter scoring = 'f1-micro'. Sebelum dioptimasi, SVM akan dijalankan dengan parameter C = 1 dan kernel = 'linear'. Sementara untuk NB akan menggunakan MultinomialNB dengan parameter *default*.

Evaluasi dan Analisis

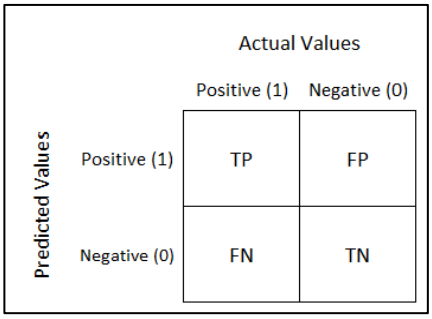
Evaluasi hasil klasifikasi dari algoritma SVM dan NB akan diukur menggunakan metrik utama yaitu F1-Measure. Perhitungan metrik berikut dapat dilakukan dengan bantuan *confusion matrix*. Skor dari *accuracy*, *precision*, *recall* dan *F1-Measure* memiliki rentang nilai 0 hingga 1. Skor dengan nilai mendekati 0 menandakan performa yang buruk, sementara skor dengan nilai mendekati 1 menandakan performa yang baik [12].

$$Accuracy = \frac{TP + TN}{n} \quad (7)$$

$$Precision = \frac{TP}{TP + FP} \quad (8)$$

$$Recall = \frac{TP}{TP + FN} \quad (9)$$

$$F1\text{-}Measure = \frac{2 \times (Precision + Recall)}{(Precision + Recall)} \quad (10)$$



Gambar 5 *Confusion Matrix [13]*