Laporan Tugas Besar

IF4071 Pemrosesan Ucapan

Pembangunan Model ASR Menggunakan Kakas

Diajukan sebagai tugas Mata Kuliah IF4071 Pemrosesan Ucapan

Dipersiapkan oleh:

13518026 Faris Fadhilah

13521069 Louis Caesa Kesuma

13521008 Jason Rivalino

13521168 Satria Octavianus Nababan

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

I. Project Domain Description

Proyek ini berfokus untuk mengembangkan sistem *Automatic Speech Recognition* (ASR) yang disesuaikan dengan domain percakapan tertentu, dengan menggunakan teknik pemodelan *fully* DNN dan menggunakan *framework* PyTorch.

Objective:

A. Membangun sistem pengenal ucapan kontinyu dengan ukuran *vocabulary* sedang.
B. Melakukan eksperimen pengujian model pengenal ucapan dengan berbagai kondisi yang mungkin untuk mendapatkan model dengan akurasi terbaik.

II. Data Description

Pada tugas besar ini, untuk *dataset audio* yang digunakan sebagai data latih untuk pembangunan model adalah *dataset* dengan rincian sebagai berikut:

ASR-IndoCSC: An Indonesian Conversational Speech Corpus
Referensi: *https://magichub.com/datasets/indonesian-conversational-speech-corpus/*

ASR-IndoCSC merupakan sebuah korpus suara percakapan spontan dalam Bahasa Indonesia yang dirancang untuk mendukung pengembangan sistem *Automatic Speech Recognition* (ASR). Dataset ini disediakan sebagai sumber daya *open-source* dengan lisensi *Magic Data open-source license*, sehingga dapat digunakan secara bebas untuk keperluan penelitian dan pengembangan.

Karakteristik Dataset ASR-IndoCSC:

1. Tipe Dataset
   Dataset ini berupa *speech corpus* yang mengandung rekaman percakapan alami antara dua pembicara dengan tema tertentu.
2. Bahasa
   a. Bahasa: id-ID, Bahasa Indonesia.
   b. Gaya bicara: Percakapan spontan, yang mencakup variasi gaya berbicara seperti intonasi, kecepatan, dan pola percakapan alami.
3. Spesifikasi Audio
   a. Frekuensi Sampling: 16 kHz
   b. Resolusi Audio: 16-bit
   c. Saluran: Mono
   d. Format File: WAV (PCM) untuk audio, dan TXT (UTF-8) untuk transkrip.
4. Lingkungan Rekaman
   a. Peralatan: Rekaman dilakukan menggunakan perangkat *mobile*.
   b. Lokasi: Lingkungan rekaman dilakukan di dalam ruangan (*indoor*), sehingga kualitas suara relatif bebas dari gangguan noise eksternal
5. Ukuran Dataset
   a. Total Durasi: 4.54 jam rekaman suara percakapan yang telah ditranskripsi.
   b. Ukuran Total: 322 MB.

c. Isi: Terdapat tujuh percakapan yang melibatkan beberapa pasangan pembicara, sehingga menghasilkan variasi suara dan gaya bicara. Ketujuh audio ini kemudian akan disegmentasi menjadi potongan sebanyak total 2886 audio yang berbeda.

Sedangkan, untuk *dataset testing* yang digunakan untuk pengujian terhadap model akan menggunakan data berupa perekaman suara yang dilakukan oleh masing-masing anggota kelompok. Setiap anggota kelompok mengucapkan kalimat tertentu dan kalimat juga dituliskan dalam bentuk file .txt. Untuk panjang audio akan bervariasi namun semua *file* akan memiliki kesamaan yaitu dengan format *file .wav* dan memiliki frekuensi *sampling rate* sebesar 16 kHz.

III. Pengembangan Model

A. *Preprocessing* Data Audio dan Teks

Pada tahap ini, data mentah berupa file audio dan file teks diproses menjadi segmen-segmen kecil yang memiliki keterkaitan waktu dan transkripsi yang tepat. File audio utama diolah menggunakan program *extracting_data.py*, di mana fungsi *slice_wav* digunakan untuk membagi file audio menjadi beberapa segmen kecil. Proses ini didasarkan pada waktu awal (*start*) dan waktu akhir (*end*) yang ditentukan dalam file teks yang menyertainya. Setiap segmen audio yang dihasilkan memiliki *sampling rate* sebesar 16 kHz, standar yang digunakan oleh model *Wav2Vec2* untuk memastikan data audio terstandarisasi. Segmen-segmen ini kemudian disimpan dalam direktori output untuk mempermudah proses

pelatihan model selanjutnya.Proses *parsing* file teks dilakukan menggunakan program *parse_text.py*. File teks berisi informasi waktu dan transkripsi teks yang relevan, dengan format tertentu seperti [start, end] <word> <word> <text>. Program ini menggunakan ekspresi reguler untuk mengekstrak informasi waktu mulai, waktu akhir, dan transkripsi teks. Informasi ini kemudian diubah menjadi daftar segmen yang merepresentasikan hubungan antara waktu dan transkripsi. *Parsing* yang akurat memastikan bahwa data yang digunakan untuk segmentasi audio sesuai dengan anotasi teks, sehingga menciptakan dataset berkualitas tinggi yang siap digunakan untuk melatih model ASR.

Setelah file teks selesai diproses, langkah berikutnya adalah membagi file audio utama menjadi segmen-segmen kecil berdasarkan informasi waktu yang telah dihasilkan dari proses *parsing*. Program *slicing_audio.py* memulai prosesnya dengan membuka file audio dalam format WAV menggunakan pustaka *pydub*. Setiap segmen audio ditentukan berdasarkan waktu mulai dan akhir dalam milidetik, sesuai dengan hasil *parsing* file teks. Program ini kemudian mengekstrak segmen audio dari file utama dan menyimpannya sebagai file terpisah dengan nama yang unik.

Untuk memastikan kompatibilitas dengan model *Wav2Vec2*, setiap segmen audio dikonversi menjadi array *NumPy* dengan *sampling rate* 16 kHz. Informasi tambahan, seperti transkripsi teks dan nama file segmen, juga disimpan untuk mempermudah pelacakan dan pengelolaan data. Proses *slicing* ini menghasilkan

data audio yang sesuai dengan transkripsi teks, yang sangat penting untuk membangun hubungan antara input audio dan output teks dalam pelatihan model.

B. Pembuatan *Dataset*

Dataset yang dihasilkan mencakup tiga komponen utama, yaitu audio, teks, dan nama file. Data ini kemudian dikonversi ke format yang kompatibel dengan pustaka Hugging Face, sehingga dapat digunakan untuk melatih model berbasis *Wav2Vec2*. Konversi ini dilakukan dengan menjaga struktur data yang rapi dan konsisten untuk memudahkan proses pelatihan model. Dataset yang telah diverifikasi dan divalidasi disimpan secara lokal untuk mempermudah penggunaannya dalam pelatihan atau pengujian model.

C. Pengembangan Model ASR

Pengembangan model ASR dimulai dengan menyiapkan *dataset* untuk pelatihan. *Dataset* kemudian dilakukan *filtering* dan diproses menggunakan fungsi map() dari pustaka Hugging Face untuk menerapkan fitur tambahan, seperti normalisasi audio dan tokenisasi teks. Proses ini diikuti dengan ekstraksi fitur audio menggunakan *Wav2Vec2FeatureExtractor*, yang bertujuan untuk mengubah data audio mentah menjadi representasi fitur yang dapat dipahami oleh model. Sementara itu, teks tokenisasi dilakukan dengan *Wav2Vec2CTCTokenizer*, yang menggunakan token khusus seperti [UNK] dan [PAD] untuk mendukung pelatihan model.

Dataset diproses lebih lanjut menjadi pasangan *input_values* (fitur audio) dan labels (representasi tokenisasi teks). Data ini kemudian disiapkan untuk pelatihan menggunakan padding dan normalisasi melalui *DataCollatorCTCWithPadding*. Padding memastikan bahwa semua data input memiliki panjang yang seragam selama pelatihan, sedangkan normalisasi membantu mengurangi noise dan variasi yang tidak relevan dalam data audio.

D. Pemodelan

Model yang digunakan adalah *Wav2Vec2*, sebuah arsitektur berbasis transformator yang dirancang khusus untuk pemrosesan suara. Model ini dipilih dari pustaka *Transformers* (pretrained model *facebook/wav2vec2-base*) dan dilakukan *finetune* pada dataset spesifik untuk meningkatkan performa pada domain tertentu. Selain itu, kepala CTC (Connectionist Temporal Classification) ditambahkan untuk mengoptimalkan proses *decoding* tanpa memerlukan penyesuaian panjang antara data *input* dan *output*.

Model *Wav2Vec2* yang digunakan adalah arsitektur *Fully DNN (End-to-End / E2E)*. Model ini secara langsung mempelajari hubungan antara data audio mentah dan transkripsi teks tanpa menggunakan komponen berbasis *HMM* atau *GMM*. Loss function yang digunakan adalah *CTC Loss*, yang memungkinkan model untuk mengakomodasi panjang input dan output yang berbeda. Evaluasi berbasis WER dan CER memberikan wawasan tentang akurasi transkripsi, sementara optimisasi seperti gradient checkpointing dan FP16 membantu mempercepat pelatihan dan menghemat sumber daya.

Pemilihan model *Fully DNN* ini didasarkan pada beberapa faktor utama:

1. Kemampuan Generalisasi
   Dengan arsitektur yang lebih fleksibel dan tidak bergantung pada asumsi distribusi tertentu (seperti Gaussian pada *HMM-GMM*), model *Fully DNN* dapat menangkap pola yang lebih kompleks.
2. *End-to-End Learning*
   Fully DNN memungkinkan proses pelatihan *end-to-end*, sehingga mengurangi kompleksitas arsitektur pipeline yang biasanya terdiri dari banyak modul pada *HMM-GMM*.
3. Efisiensi Representasi
   Model *DNN* mampu belajar representasi fitur yang lebih efisien dari data audio mentah, menghilangkan kebutuhan akan *preprocessing* manual yang ekstensif seperti pada *HMM*.

Dengan pendekatan ini, model *Fully DNN* yang digunakan tidak hanya lebih unggul secara teknis tetapi juga memberikan hasil yang lebih baik dalam tugas pengenalan suara dibandingkan pendekatan berbasis HMM.

E. Pelatihan dan Evaluasi

Pelatihan dilakukan dengan menggunakan Trainer dari Transformers, yang secara otomatis mengelola model, *dataset*, data collator, dan evaluasi. Konfigurasi pelatihan ditentukan melalui *TrainingArguments*, yang mencakup pengaturan batch size, jumlah epoch, dan strategi evaluasi. Gradient checkpointing diterapkan untuk mengurangi kebutuhan memori selama pelatihan, sedangkan FP16 digunakan untuk mempercepat pelatihan dengan operasi floating-point yang lebih kecil.

Evaluasi model dilakukan menggunakan dua metrik utama, yaitu *Word Error Rate* (*WER*) dan *Character Error Rate* (*CER*). *WER* mengukur tingkat akurasi transkripsi pada tingkat kata, sementara *CER* memberikan gambaran granular tentang kesalahan pada tingkat karakter.

F. Pengujian Model

Setelah pelatihan *dataset* selesai dan model telah terbentuk, langkah berikutnya yaitu dilakukan pengujian dengan menggunakan data audio yang belum terlihat sebelumnya. Input audio diproses menjadi prediksi teks menggunakan decoder berbasis model *Wav2Vec2* dan tokenizer. Hasil pengujian mencakup transkripsi teks dan evaluasi terhadap transkripsi referensi untuk mengukur performa model.

IV. Eksperimen Model

Program eksperimen terdapat ada file *experiment.ipynb*, Kode dimulai dengan mengimpor modul penting: *transformers*, *torch*, dan *scipy.io.wavfile*. Library Transformers digunakan untuk membuat model *Wav2Vec2* beserta tokenisasi dan ekstraksi fitur. Library Torch berperan dalam manipulasi tensor dan melakukan inferensi dengan model neural network. Sedangkan wavfile dari scipy.io digunakan

untuk membaca file audio berformat WAV. Modul-modul ini bersama-sama membentuk dasar untuk melakukan pemrosesan data audio dan inferensi model.

1. Memuat Model dan Processor
   Selanjutnya, model yang sudah dilatih sebelumnya dimuat menggunakan Wav2Vec2ForCTC, yang dirancang untuk tugas transkripsi berbasis Connectionist Temporal Classification (CTC). Bersama dengan itu, Wav2Vec2Processor dimuat untuk memproses audio mentah menjadi input yang sesuai untuk model. Kedua elemen ini berasal dari jalur lokal ../results/savedModel, yang menunjukkan bahwa model disimpan secara lokal setelah pelatihan. Processor bertugas menggabungkan tokenizer dan feature extractor untuk memastikan audio dikonversi menjadi representasi numerik yang dapat dimengerti oleh model.

2. Membaca dan Memproses Audio
   File audio dibaca menggunakan wavfile.read, yang menghasilkan dua hal: frekuensi sampling (sampling_rate) dan data audio dalam bentuk array. Audio kemudian divalidasi untuk memastikan frekuensi samplingnya adalah 16kHz, karena model Wav2Vec2 hanya mendukung frekuensi ini. Jika tidak sesuai, program akan memunculkan error. Audio yang telah divalidasi kemudian diproses oleh processor untuk menghasilkan tensor PyTorch (input_values). Tensor ini dikonversi ke tipe float agar kompatibel dengan model.

3. Memilih Perangkat untuk Inferensi
   Pada bagian ini, kode menentukan perangkat yang digunakan untuk inferensi. Jika GPU tersedia, tensor dan model akan dipindahkan ke GPU untuk mempercepat proses. Jika tidak, CPU akan digunakan. Proses ini dilakukan dengan memanfaatkan fungsi torch.cuda.is_available(). Pemilihan perangkat ini memastikan program berjalan optimal di berbagai lingkungan, baik dengan atau tanpa GPU.

4. Inferensi dan Transkripsi
   Inferensi dilakukan dalam blok tanpa perhitungan gradien (torch.no_grad()), sehingga mengurangi konsumsi memori. Tensor input (input_values) diberikan ke model, menghasilkan output logits, yaitu skor prediksi untuk setiap token. Skor ini kemudian diproses menggunakan argmax untuk mengambil ID token prediksi dengan probabilitas tertinggi. ID tersebut diterjemahkan menjadi teks menggunakan fungsi batch_decode dari processor, menghasilkan transkripsi audio.

5. Menampilkan Hasil Transkripsi
   Akhirnya, hasil transkripsi ditampilkan di layar. Pada kasus ini, hasilnya adalah "halo pisela," yang menunjukkan teks dari ucapan dalam file audio. Proses ini adalah langkah terakhir dalam pipeline, mengonversi audio mentah menjadi representasi teks yang dapat digunakan untuk analisis lebih lanjut.

V. Analisis Hasil dan Rekomendasi

Hasil dari eksperimen yang dilakukan terhadap model ASR menggunakan audio sebanyak 13 file audio dengan masing-masing anggota kelompok memberikan minimal 3 file audio untuk testing model ASR. Data yang didapat dari hasil testing meliputi Ground Truth, Predicted, Character Error Rate (CER), Word Error Rate (WER), dan Overall. Analisis hasil testing berfokus pada Ground Truth dan Predicted yang didapat dari model ASR ini.

Tabel 1. Analis Hasil

Nama File Audio	Ground Truth	Predicted
Faris_1.wav	saya orang bandung	saiya orang pantung
Faris_2.wav	saya sudah ngantuk	*iya sudah ngantu*
Faris_3.wav	faris sedang lapar	pari* sedang lapar
Faris_4.wav	saya sedang berjalan jalan di kota bandung	sa_iya sedang perjaran jalu andi kutapandung
Jason_1.wav	aku siap	A_ku si_ya
Jason_2.wav	aku pusing nubes	Aku *u_sing nubes
Jason_3.wav	tidur enak kali ya	Didu* ena* kali ya
Louis_1.wav	selamat pagi	selamat *agi
Louis_2.wav	aku mau pergi	oko
Louis_3.wav	aku mau tidur	aku maut *itu*
Satria_1.wav	halo selamat pagi	Oo kamukoni
Satria_2.wav	hai aku satria	Kawok ko
Satria_3.wav	pergi dulu ya	Rigilu ya

```
File ID: Faris_1 (CER: 0.16666666666666666, WER: 0.6666666666666666)
    Predicted: saiya orang pantung
    Ground Truth: saya orang bandung

File ID: Faris_2 (CER: 0.16666666666666666, WER: 0.6666666666666666)
    Predicted: iya sudah ngantu
    Ground Truth: saya sudah ngantuk

File ID: Faris_3 (CER: 0.1111111111111111, WER: 0.3333333333333333)
    Predicted: pari sedang lapar
    Ground Truth: faris sedang lapar

File ID: Faris_4 (CER: 0.23809523809523808, WER: 1.0)
    Predicted: sa iya sedang perjaran jalu andi kutapandung
    Ground Truth: saya sedang berjalan jalan di kota bandung
```

Gambar 1. Eksperimen Dataset Faris

```
File ID: Jason_1 (CER: 0.5, WER: 2.0)
    Predicted: a ku si ya
    Ground Truth: aku siap

File ID: Jason_2 (CER: 0.125, WER: 0.6666666666666666)
    Predicted: aku u sing nubes
    Ground Truth: aku pusing nubes

File ID: Jason_3 (CER: 0.16666666666666666, WER: 0.5)
    Predicted: didu ena kali ya
    Ground Truth: tidur enak kali ya
```

Gambar 2. Eksperimen Dataset Jason

```
File ID: Louis_1 (CER: 0.16666666666666666, WER: 1.0)
    Predicted: selamet agi
    Ground Truth: selamat pagi

File ID: Louis_2 (CER: 0.9230769230769231, WER: 1.0)
    Predicted: oko
    Ground Truth: aku mau pergi

File ID: Louis_3 (CER: 0.3076923076923077, WER: 0.6666666666666666)
    Predicted: aku maut itu
    Ground Truth: aku mau tidur
```

Gambar 3. Eksperimen Dataset Louis

```
File ID: Satria 1 (CER: 0.7647058823529411, WER: 1.0)
    Predicted: oo komukoni
    Ground Truth: Halo selamat pagi

File ID: Satria 2 (CER: 0.7857142857142857, WER: 1.0)
    Predicted: kawok ko
    Ground Truth: Hai aku satria

File ID: Satria 3 (CER: 0.46153846153846156, WER: 0.6666666666666666)
    Predicted: rigilu ya
    Ground Truth: Pergi dulu ya
```

Gambar 4. Eksperimen Dataset Satria

```
Evaluation Results:
Overall Character Error Rate (CER): 0.3409090909090909
Overall Word Error Rate (WER): 0.8333333333333334
```

Gambar 5. Hasil Ekperimen Keseluruhan

Banyak kesalahan terjadi dalam transkripsi data, terutama pada file dengan ID "Louis" dan "Satria". Tingginya Character Error Rate (CER) dan Word Error Rate (WER) pada kedua dataset ini menunjukkan kualitas hasil prediksi yang kurang memuaskan. Hal ini kemungkinan besar disebabkan oleh kualitas audio yang mendem (kurang jelas atau terdistorsi), yang menyulitkan sistem dalam mengenali ucapan secara akurat.

Pada data "Louis", terlihat bahwa CER berkisar antara 0,16 hingga 0,92, dan WER konsisten di angka tinggi, yaitu 1,0 untuk sebagian besar file. Sebagai contoh, pada file "Louis_2", prediksi "oko" sangat jauh dari transkrip sebenarnya, yaitu "aku mau pergi", yang mengindikasikan bahwa sistem kesulitan mengenali ucapan dalam kondisi audio yang buruk. Begitu juga pada file "Louis_3", prediksi "aku maut itu" memiliki perbedaan signifikan dengan transkrip "aku mau tidur", menunjukkan kesalahan pada level kata maupun karakter. Data "Satria" juga menunjukkan pola serupa dengan CER yang cukup tinggi, yaitu antara 0,46 hingga 0,76, dan WER mencapai 1,0 dalam banyak kasus. Sebagai contoh, pada file "Satria_1", prediksi "oo komukoni" jauh berbeda dari transkrip asli "Halo selamat pagi", menunjukkan bahwa distorsi dalam audio sangat mempengaruhi kemampuan pengenalan sistem.

Sebaliknya, data "Faris" menunjukkan hasil yang lebih baik dengan CER yang lebih rendah (0,11 hingga 0,23) dan WER yang bervariasi namun lebih terkendali, yaitu antara 0,33 hingga 1,0. Sebagai contoh, pada file "Faris_3", prediksi "pari sedang lapar" hanya memiliki sedikit kesalahan dibandingkan dengan transkrip asli "faris sedang lapar". Hal ini menunjukkan bahwa kualitas audio pada data "Faris" lebih baik, sehingga menghasilkan transkripsi yang lebih akurat.

Pada Data Jason, CER dan WER cenderung lebih tinggi dibandingkan Data Faris. Hal ini menunjukkan bahwa prediksi lebih sering meleset, terutama pada File ID: Jason_1, yang memiliki WER sebesar 2.0, mencerminkan kesalahan yang sangat

besar dalam menangkap makna ucapan. Kesalahan ini mungkin disebabkan oleh variasi fonetik atau struktur audio yang lebih sulit dikenali oleh model.

Kesalahan utama dalam transkripsi dipengaruhi oleh beberapa faktor, seperti kualitas data pelatihan, variasi aksen, dan pengucapan dalam audio. Kata-kata yang mirip secara fonetik, seperti **ngantu** dan **ngantuk**, menjadi tantangan bagi model untuk membedakannya secara konsisten. Hal ini menunjukkan bahwa model membutuhkan pengolahan data lebih lanjut untuk menangani variasi ini.

Untuk meningkatkan akurasi transkripsi, perlu dilakukan *fine-tuning* model menggunakan *dataset* yang lebih besar, beragam, dan spesifik terhadap aksen atau pola bicara tertentu. Selain itu, perhatian terhadap *preprocessing* audio, seperti pengurangan *noise* atau peningkatan kualitas *input* audio, akan membantu meningkatkan hasil transkripsi. Dengan langkah-langkah ini, diharapkan model dapat memberikan hasil yang lebih akurat dan andal.

VI. References

A. https://medium.com/@marko.briesemann/fine-tuning-and-deployment-of-open-source-asr-models-with-pytorch-d3264fd1b160
B. https://www.gladia.io/blog/fine-tuning-asr-models
C. https://huggingface.co/learn/audio-course/chapter5/asr_models
D. https://www.youtube.com/watch?v=Ffw9TZqiFVM

VII. Presentation Video Link

https://youtu.be/ei8VLRiL6sE

VIII. Link Repository GitHub

https://github.com/satrianababan/IF4071_TubesASR.git