

**LAPORAN HASIL PROYEK
PENGOLAHAN SINYAL DIGITAL**



OLEH KELOMPOK 2:

Shinta Usaila Farachin	: 23031554160
Ikhrima Atusifah	: 23031554181
Arina Tri Yuni W. T.	: 23031554203

Kelas 2023A

**UNIVERSITAS NEGERI SURABAYA
2024**

TEXT TO SPEECH

Latar Belakang

Kemajuan teknologi di era digital telah memberikan dampak besar terhadap berbagai aspek kehidupan, termasuk cara manusia berkomunikasi dan mengakses informasi. Salah satu inovasi yang berkembang pesat adalah teknologi Text-to-Speech (TTS), yaitu teknologi yang mengubah teks menjadi suara. Teknologi ini telah digunakan secara luas di berbagai bidang, seperti pendidikan, layanan pelanggan, dan aksesibilitas bagi penyandang disabilitas.

Pengembangan teknologi TTS menjadi sangat penting, terutama dalam mendukung inklusi sosial dan memberikan kemudahan bagi pengguna dengan kebutuhan khusus, seperti tunanetra. Selain itu, TTS memiliki potensi untuk meningkatkan efisiensi dalam aktivitas sehari-hari, seperti membaca teks panjang tanpa harus melakukannya secara manual.

Proyek ini bertujuan untuk mengembangkan sistem TTS yang mampu menghasilkan suara yang jelas, alami, dan sesuai konteks. Penelitian ini memanfaatkan pendekatan modern berbasis machine learning, khususnya menggunakan deep learning, agar hasil suara yang dihasilkan mendekati suara manusia.

Tujuan

1. Mengembangkan teknologi TTS yang responsif dan fleksibel untuk berbagai jenis teks, baik dalam bahasa formal maupun informal.
2. Meningkatkan kualitas suara yang dihasilkan, sehingga terdengar lebih alami dan mudah dipahami oleh pengguna.
3. Mempermudah aksesibilitas informasi bagi kelompok yang memiliki keterbatasan visual atau membaca.
4. Menyediakan solusi efisien untuk membaca dokumen panjang atau konten digital secara otomatis.
5. Menyesuaikan TTS untuk berbagai keperluan seperti pendidikan, layanan publik, dan hiburan.

Manfaat

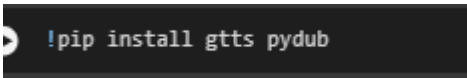
1. Membantu penyandang disabilitas, terutama tunanetra, untuk mengakses informasi berbasis teks secara mandiri.
2. Mempercepat konsumsi informasi dari teks dengan mendengarkan alih-alih membaca.
3. Mendukung pembelajaran jarak jauh dengan menyediakan alat bantu suara yang mempermudah proses belajar.
4. Memberikan pengalaman pelanggan yang lebih baik melalui implementasi dalam chatbot atau layanan otomatis berbasis suara.
5. Mengurangi kebutuhan akan narator manusia untuk menciptakan konten audio dalam jumlah besar.

DATASET

Dataset yang digunakan dalam proyek ini terdiri dari data teks yang diinputkan secara manual. Data ini mencakup berbagai jenis kalimat, seperti kalimat formal, informal, pertanyaan, dan pernyataan. Proses penginputan manual dilakukan untuk memastikan teks yang digunakan relevan, berkualitas, dan mencakup variasi pola bahasa yang dibutuhkan.

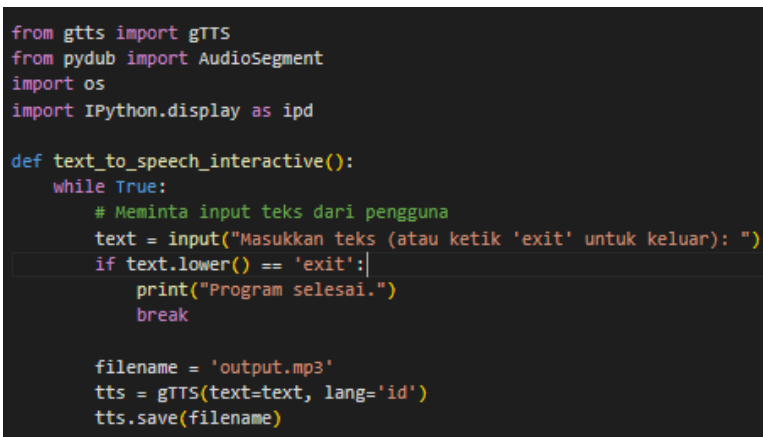
ALUR DAN TAHAPAN Pengerjaan

Ada beberapa tahapan yang dikerjakan dalam project ini yaitu:

1. 

Menginstal Pustaka

Tahap pertama adalah menginstal pustaka yang diperlukan menggunakan perintah `!pip install gtts pydub`. gTTS (Google Text-to-Speech): Pustaka ini digunakan untuk mengubah teks menjadi suara (TTS). pydub: Pustaka ini digunakan untuk memproses atau mengedit file audio, seperti mengubah format, memotong, atau menggabungkan file audio.

2. 

Kode di atas berfungsi untuk mengonversi teks menjadi suara menggunakan pustaka gTTS. Program berjalan dalam loop yang meminta pengguna memasukkan teks. Jika pengguna mengetikkan "exit", program akan berhenti. Jika pengguna memasukkan teks lainnya, teks tersebut diubah menjadi suara menggunakan fungsi gTTS, dengan bahasa Indonesia sebagai pengaturannya (*lang='id'). Hasilnya disimpan dalam file MP3 dengan nama output.mp3.

Proses ini berulang setiap kali pengguna memasukkan teks baru. Dengan pendekatan ini, program memungkinkan pengguna untuk membuat file audio dari teks secara cepat dan mudah. Hal ini bermanfaat untuk berbagai kebutuhan, seperti membantu membaca teks secara otomatis atau membuat konten audio. Program juga fleksibel, karena terus berjalan hingga pengguna memutuskan untuk keluar.

3.

```
# Konversi file ke format lain
audio = AudioSegment.from_mp3(filename)
wav_filename = filename.replace('.mp3', '.wav')
audio.export(wav_filename, format='wav')
```

Kode di atas digunakan untuk mengonversi file audio dari format MP3 ke format WAV menggunakan pustaka pydub. Pertama, file MP3 yang sudah dibuat sebelumnya (output.mp3) dibaca menggunakan fungsi `AudioSegment.from_mp3(filename)`. Setelah itu, nama file diubah dari `.mp3` menjadi `.wav` dengan mengganti ekstensi menggunakan fungsi `replace`. Akhirnya, file audio tersebut diekspor ke format WAV dan disimpan dengan nama baru (output.wav) menggunakan perintah `audio.export(wav_filename, format='wav')`. Proses ini bermanfaat jika format WAV diperlukan untuk aplikasi atau perangkat tertentu yang tidak mendukung MP3.

4.

```
# Menampilkan hasil audio di bawah teks
print("\n=== Hasil Audio ===")
print(f"Teks: {text}")
ipd.display(ipd.Audio(filename))
```

Kode di atas digunakan untuk menampilkan teks yang sudah diubah menjadi audio serta memutar hasil audio tersebut secara langsung. Pertama, program mencetak teks yang dimasukkan oleh pengguna dengan perintah `print(f"Teks: {text}")`. Kemudian, menggunakan pustaka `IPython.display`, file audio yang telah dibuat (dalam format MP3) diputar secara langsung di dalam program dengan perintah `ipd.display(ipd.Audio(filename))`. Ini memungkinkan pengguna mendengar hasil konversi teks ke suara tanpa harus membuka file audio di aplikasi lain. Proses ini memudahkan pengguna untuk memeriksa hasil konversi secara cepat dan langsung.

5.

```
# File dengan kecepatan ditingkatkan
faster_audio = audio.speedup(playback_speed=1.5)
faster_filename = 'faster_' + filename
faster_audio.export(faster_filename, format='mp3')
print("Audio dengan kecepatan ditingkatkan:")
ipd.display(ipd.Audio(faster_filename))
```

Kode di atas digunakan untuk membuat versi audio dengan kecepatan yang lebih cepat, yaitu 1,5 kali dari kecepatan aslinya. Pertama, file audio yang sudah ada diproses menggunakan fungsi `audio.speedup(playback_speed=1.5)`, yang meningkatkan kecepatan pemutaran tanpa mengubah nada suara. File hasilnya kemudian disimpan dengan nama baru, yaitu `faster_output.mp3`, menggunakan perintah `faster_audio.export(faster_filename, format='mp3')`. Setelah itu, program menampilkan teks "Audio dengan kecepatan ditingkatkan:" dan memutar file audio tersebut langsung di program menggunakan `ipd.display(ipd.Audio(faster_filename))`. Fitur ini berguna untuk pengguna yang ingin mendengar versi audio yang lebih cepat, misalnya untuk kebutuhan percepatan pemutaran informasi.

6.

```
# File dengan volume ditingkatkan
louder_audio = audio + 10
louder_filename = 'louder_' + filename
louder_audio.export(louder_filename, format='mp3')
print("Audio dengan volume ditingkatkan:")
ipd.display(ipd.Audio(louder_filename))
```

Kode di atas digunakan untuk meningkatkan volume file audio. Pertama, file audio yang ada diproses dengan menambahkan nilai +10 ke objek audio, yang berarti volume dinaikkan sebesar 10 dB (desibel). File hasilnya kemudian disimpan dengan nama baru, yaitu `louder_output.mp3`, menggunakan perintah `louder_audio.export(louder_filename, format='mp3')`. Setelah itu, program menampilkan teks "Audio dengan volume ditingkatkan:" dan memutar file audio tersebut langsung di program menggunakan `ipd.display(ipd.Audio(louder_filename))`. Proses ini berguna jika hasil audio asli terdengar terlalu pelan, sehingga pengguna dapat mendengar versi yang lebih jelas.

7.

```
# Potongan audio
start_time = 5000
end_time = min(len(audio), 10000)
sliced_audio = audio[start_time:end_time]
sliced_filename = 'sliced_' + filename
sliced_audio.export(sliced_filename, format='mp3')
print("Potongan audio (5-10 detik):")
ipd.display(ipd.Audio(sliced_filename))
```

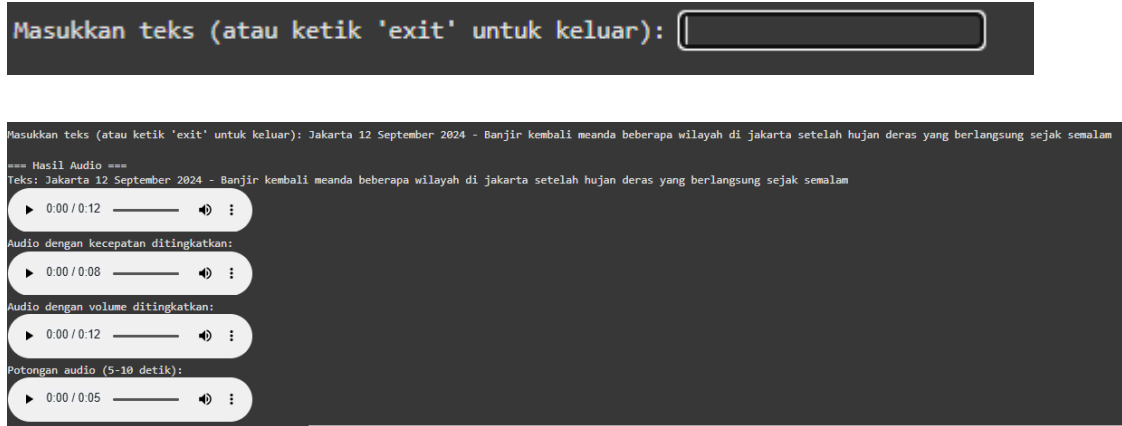
Kode di atas digunakan untuk memotong bagian tertentu dari file audio. Potongan audio diambil dari detik ke-5 hingga detik ke-10, dengan waktu mulai (`start_time`) ditentukan sebagai 5000 milidetik (5 detik) dan waktu akhir (`end_time`) diatur menjadi nilai terkecil antara panjang total audio atau 10000 milidetik (10 detik). Bagian audio yang terpotong disimpan dalam variabel `sliced audio`. Potongan ini kemudian diekspor sebagai file baru dengan nama `sliced_output.mp3` menggunakan perintah `sliced_audio.export(sliced_filename, format='mp3')`. Terakhir, program menampilkan teks "Potongan audio (5-10 detik):" dan memutar file audio hasil potongan menggunakan `ipd.display(ipd.Audio(sliced_filename))`. Proses ini berguna untuk mengambil bagian tertentu dari audio yang relevan atau dibutuhkan tanpa harus memutar keseluruhan file.

8.

```
# Menjalankan fungsi
text_to_speech_interactive()
```

Fungsi ini memulai program interaktif yang memungkinkan pengguna memasukkan teks untuk diubah menjadi suara. Program akan terus meminta input teks, mengkonversinya menjadi file audio, dan memutarnya sampai pengguna mengetikkan "exit" untuk menghentikan program. Fungsi ini menggabungkan semua langkah yang telah dijelaskan, termasuk konversi teks ke audio, pengolahan file, dan pemutaran hasilnya.

Hasil



Gambar di atas menunjukkan hasil dari program Text-to-Speech (TTS) yang mengubah teks menjadi suara dengan berbagai pengolahan sinyal. Teks yang dimasukkan adalah "Jakarta 12 September 2024 - Banjir kembali melanda beberapa wilayah di Jakarta setelah hujan deras yang berlangsung sejak semalam."

1. Hasil Audio Awal: Bagian pertama menampilkan teks asli yang dikonversi menjadi suara dalam format audio standar (MP3).
2. Audio dengan Kecepatan Ditingkatkan: Pada bagian ini, file audio diproses untuk mempercepat kecepatan putaran sebesar 1,5 kali menggunakan fungsi speedup. Ini adalah salah satu bentuk pengolahan sinyal audio untuk mengubah durasi tanpa mengubah pitch (nada).
3. Audio dengan Volume Ditingkatkan: Bagian ini menunjukkan hasil audio dengan volume yang dinaikkan sebesar 10 dB. Pengolahan sinyal pada bagian ini bertujuan meningkatkan amplitudo suara untuk memperjelas audio.
4. Potongan Audio (5-10 detik): Pada bagian terakhir, file audio dipotong untuk menampilkan bagian tertentu, yaitu dari detik ke-5 hingga detik ke-10. Ini merupakan pengolahan sinyal yang membatasi durasi audio ke interval waktu tertentu.

Kesimpulan

Program Text-to-Speech (TTS) ini dapat mengubah teks menjadi audio dengan berbagai pengolahan sinyal, seperti mempercepat suara, meningkatkan volume, dan mempersingkat durasi audio sesuai kebutuhan. Proses ini sangat bermanfaat untuk menghasilkan audio yang lebih fleksibel dan diinginkan oleh pengguna, contohnya untuk mempercepat penyampaian informasi, membuat suara lebih jelas, atau hanya memanfaatkan bagian tertentu dari audio. Di samping itu, program ini sangat berguna bagi pengguna tunanetra, karena memungkinkan mereka untuk mendengarkan informasi yang sebelumnya hanya tersedia dalam teks, sehingga membantu mereka untuk mengakses informasi dengan lebih cepat dan efisien.