1. Pendahuluan

Seiring berkembangnya era digital, penyebaran informasi menjadi sangat cepat dan masif. Namun, kemudahan ini juga diiringi oleh tantangan besar, yaitu maraknya berita palsu (fake news) yang dapat meresahkan masyarakat. Kemampuan untuk membedakan berita asli dan palsu secara otomatis menjadi sangat krusial. Klasifikasi teks menggunakan machine learning, khususnya dengan arsitektur Recurrent Neural Network (RNN), menawarkan solusi yang efektif karena kemampuannya dalam memahami konteks dan pola dalam data sekuensial seperti teks.

Tugas ini bertujuan untuk membangun sebuah model klasifikasi teks menggunakan arsitektur LSTM (Long Short-Term Memory). Ruang lingkup proyek ini mencakup proses dari persiapan dataset, pra-pemrosesan teks, pembangunan model, hingga evaluasi performa model dalam mengklasifikasikan judul berita sebagai "asli (real)" atau "palsu (fake)".

2. Dataset

Sumber Data:

Dataset yang digunakan adalah FakeNewsNet, yang bersumber dari platform terbuka Kaggle.

Deskripsi Dataset:

Sesuai ketentuan tugas, saya menggunakan sampel data sebanyak 200 judul berita, yang dibagi secara seimbang menjadi 100 data untuk kelas berita asli dan 100 data untuk kelas berita palsu. Fitur utama yang digunakan sebagai input untuk model adalah title (judul berita), sedangkan real (dengan nilai 1 untuk asli dan 0 untuk palsu) digunakan sebagai label. Terdapat variasi gaya bahasa yang jelas, di mana judul berita palsu cenderung lebih sensasional dan provokatif, sementara judul berita asli lebih objektif.

Alasan Pemilihan Dataset:

Topik deteksi berita palsu dipilih karena relevansinya yang tinggi dengan kondisi sosial saat ini. Dataset ini menyediakan kasus klasifikasi biner yang jelas dan memiliki variasi gaya bahasa yang menantang, sehingga sangat cocok untuk menguji kemampuan model RNN dalam analisis teks.

3. Implementasi Model

3.1 Arsitektur RNN

Model yang diimplementasikan menggunakan jenis LSTM (Long Short-Term Memory). Arsitektur ini dibangun secara sekuensial menggunakan Keras dan terdiri dari:

- Embedding Layer: Mengubah input berupa urutan indeks kata menjadi vektor padat. Parameter yang digunakan: input_dim=5000 (ukuran kosakata) dan output_dim=64 (dimensi vektor).
- LSTM Layer: Lapisan inti yang memproses urutan vektor untuk menangkap dependensi dan konteks dalam teks. Lapisan ini memiliki 64 unit.
- Dense Layer: Lapisan output dengan 1 unit dan fungsi aktivasi sigmoid untuk menghasilkan probabilitas klasifikasi biner.

3.2 Preprocessing

Langkah pra-pemrosesan yang dilakukan adalah sebagai berikut:

- Tokenisasi: Teks judul diubah menjadi urutan integer menggunakan Tokenizer dari Keras dengan batasan 5000 kata yang paling sering muncul.
- Padding: Setiap urutan disamakan panjangnya menjadi 50 token menggunakan pad_sequences untuk memastikan input model seragam.

3.3 Pengaturan Eksperimen

- Optimizer: adam
- Loss Function: binary_crossentropy
- Metrics: accuracy
- Epochs: 10
- Batch Size: 16

3.4 Log Eksperimen

Proses eksperimen difokuskan pada analisis performa model dasar.

Percobaan	Model	Dropout	Optimizer	Akurasi Validasi	Catatan
#1	LSTM (64 unit)	0	Adam	~40-50% (tidak stabil)	Model dasar gagal total untuk belajar. Akurasi yang dihasilkan lebih buruk dari tebakan acak. Hal ini mengindikasikan masalah fundamental pada data yang digunakan (jumlahnya terlalu sedikit).

Karena hasil percobaan pertama sangat buruk, eksplorasi dengan mengubah arsitektur (seperti menambah Dropout) menjadi tidak relevan sebelum masalah utama (kekurangan data) diatasi.

4. Evaluasi Hasil

- Metrik: Model final dievaluasi pada data uji dan hanya berhasil mencapai akurasi akhir sebesar 42.50%.
- Visualisasi: Grafik learning curve menunjukkan kurva akurasi dan loss yang tidak beraturan (erratic) dan tidak menunjukkan tren belajar yang jelas. Kurva validasi berfluktuasi secara liar, mengonfirmasi bahwa model tidak menemukan pola yang dapat digeneralisasi.
- Analisis Performa: Performa model sangat buruk. Kegagalan ini secara konklusif disebabkan oleh jumlah data yang tidak memadai (hanya 200 judul) untuk melatih model deep learning yang relatif kompleks seperti LSTM. Model tidak memiliki cukup contoh untuk memahami perbedaan antara berita asli dan palsu.

5. Refleksi Pribadi

- Tantangan Utama:

  Tantangan utama yang teridentifikasi setelah melihat hasil adalah jumlah data yang sangat terbatas (200 sampel). Model deep learning seperti LSTM membutuhkan ribuan contoh untuk dapat belajar secara efektif. Dengan data yang minim, model justru "bingung" dan tidak bisa belajar.

- Solusi dan Hasil:

  Solusi yang paling jelas untuk masalah ini adalah dengan menambah jumlah data secara signifikan. Percobaan dengan 200 data ini menjadi bukti bahwa kuantitas data adalah prasyarat utama sebelum arsitektur model dapat bekerja secara efektif.

- Penggunaan AI Assistant:

  Dalam tugas ini, saya memanfaatkan AI Assistant untuk: 1) membantu menyusun kerangka kode awal, 2) memberikan penjelasan fungsi setiap library, dan 3) membantu menyusun draf laporan ini setelah saya mendapatkan hasil. Seluruh kode, hasil, dan analisis telah saya verifikasi dan jalankan secara mandiri.

- Pelajaran Paling Penting:

  Pelajaran terpenting dari tugas ini adalah bahwa keberhasilan sebuah model tidak hanya bergantung pada arsitektur yang canggih, tetapi juga pada kualitas dan kuantitas data. Kegagalan dalam percobaan awal ini adalah bagian fundamental dari siklus pengembangan model yang mengarahkan pada identifikasi masalah inti.

6. Kesimpulan dan Saran

Kesimpulan:

Percobaan untuk membangun model LSTM untuk klasifikasi berita palsu dengan 200 data judul tidak berhasil. Model gagal belajar dan menghasilkan akurasi (42.50%) yang lebih rendah dari tebakan acak. Kegagalan ini secara jelas disebabkan oleh jumlah data pelatihan yang tidak mencukupi.

Rekomendasi untuk Pengembangan Selanjutnya:

Berdasarkan analisis kegagalan, langkah perbaikan yang paling prioritas adalah:

1. Menambah Jumlah Data (Prioritas Utama): Meningkatkan jumlah sampel data secara drastis (misalnya, menjadi 2000 data, 1000 per kelas) adalah langkah pertama dan terpenting yang harus dilakukan.
2. Menggunakan Konten Penuh: Menggunakan seluruh isi berita (text) sebagai input, bukan hanya judul, untuk memberikan konteks yang jauh lebih kaya kepada model.
3. Mencoba Model yang Lebih Sederhana: Sebagai perbandingan, mencoba model klasik seperti Naive Bayes untuk melihat apakah data yang sedikit lebih cocok untuk model yang lebih sederhana.

7. Referensi

- Deep-Learning-Course/courses/week05 at master · virgantara/Deep-Learning-Course
- Fake News