Midterm Task (UTS)

Aletha Tanisha (2106722770)

October 25, 2024

1 Pendahuluan

Dalam era digital saat ini, penyebaran informasi melalui email dan pesan teks telah menjadi bagian dari kehidupan sehari-hari. Namun, masalah utama yang sering muncul adalah *spam*, yaitu pesan yang tidak diinginkan dan seringkali berisi iklan, penipuan, atau konten berbahaya. Untuk mengatasi masalah ini, deteksi *spam* secara otomatis menjadi sangat penting agar pengguna dapat terhindar dari gangguan dan potensi risiko keamanan.

Berbagai metode telah dikembangkan untuk mendeteksi *spam*, seperti Naive Bayes dan Support Vector Machines (SVM). Akan tetapi, seiring dengan kemajuan di bidang *Natural Languange Processing* (NLP), pendekatan baru seperti model Transformers telah terbukti lebih efektif dalam memahami dan mengolah bahasa alami. Salah satu model yang sangat populer adalah BERT (*Bidirectional Encoder Representations from Transformers*), yang mampu memahami konteks kalimat secara lebih baik dengan pendekatan bidirectional.

Dalam penelitian ini, penulis mengaplikasikan model BERT untuk melakukan klasifikasi teks pada dataset *SMS Spam Collection* yang diperoleh dari Kaggle. Dataset ini berisi kumpulan pesan teks yang telah dilabeli sebagai *spam* atau *ham*. Tujuan dari penelitian ini adalah mengevaluasi performa model BERT dalam membedakan pesan *spam* dari pesan yang sah. Dengan kemampuan BERT dalam memahami konteks kalimat secara menyeluruh, diharapkan model ini dapat mengenali pola-pola dalam teks dan memberikan prediksi yang akurat.

Melalui penerapan BERT dalam deteksi *spam* ini, diharapkan model dapat meningkatkan efisiensi deteksi *spam* secara otomatis. Selain itu, hasil dari penelitian ini juga diharapkan dapat memberikan wawasan lebih lanjut mengenai potensi penerapan teknologi *deep learning* dalam klasifikasi teks dan analisis perilaku berbasis teks.

2 Tinjauan Pustaka

Spam atau pesan yang tidak diinginkan telah menjadi masalah yang signifikan dalam era komunikasi digital, terutama melalui email dan pesan teks. Deteksi *spam* adalah proses yang bertujuan untuk mengidentifikasi pesan yang tidak diinginkan dan menyaringnya sebelum mencapai pengguna. Berbagai metode telah dikembangkan untuk melakukan deteksi *spam* secara otomatis, mulai dari metode tradisional seperti *Naive Bayes* hingga metode berbasis *deep learning* yang lebih canggih, seperti *Transformers*, termasuk model BERT (*Bidirectional Encoder Representations from Transformers*).

2.1 Deteksi Spam dalam Pesan Teks

Deteksi *spam* pada pesan teks merupakan masalah klasifikasi biner, di mana pesan harus diklasifikasikan sebagai *spam* atau *ham* (bukan *spam*). Metode deteksi *spam* tradisional, seperti *Naive Bayes* dan *Support Vector Machines* (SVM), telah digunakan secara luas karena kesederhanaan dan efektivitasnya dalam menangani tugas klasifikasi teks berbasis fitur statistik, seperti frekuensi kata. Namun, seiring dengan perkembangan konten *spam* yang semakin kompleks, pendekatan ini sering kali gagal dalam memahami konteks bahasa yang lebih rumit.

Spam sering kali dirancang untuk menghindari deteksi melalui variasi pola teks dan konten. Oleh karena itu, pendekatan yang lebih kuat diperlukan untuk mengidentifikasi pola bahasa yang tidak eksplisit. Dalam hal ini, model berbasis *deep learning* telah menunjukkan keunggulannya dalam menangkap hubungan yang lebih dalam antara kata dan konteks kalimat.

2.2 Perkembangan Natural Language Processing (NLP)

Natural Language Processing (NLP) merupakan bidang yang memfokuskan pada interaksi antara komputer dan bahasa manusia. Sebelum adanya model berbasis *Transformers*, pendekatan NLP lebih banyak menggunakan teknik berbasis statistik sederhana, seperti *bag-of-word*s dan TF-IDF, yang hanya memperhatikan frekuensi kata tanpa mempertimbangkan urutan kata atau konteks.

Word embeddings seperti Word2Vec dan GloVe kemudian diperkenalkan untuk menangkap makna kata dalam bentuk vektor dan mengatasi keterbatasan metode berbasis frekuensi kata. Namun, metode ini tetap memiliki keterbatasan dalam memahami konteks penuh dari sebuah kalimat atau urutan teks yang lebih panjang. Masalah ini kemudian diatasi dengan diperkenalkannya model berbasis *Transformers*.

2.3 Model Transformers dan BERT

Transformers diperkenalkan oleh Vaswani et al. (2017) sebagai arsitektur baru untuk menangani tugas-tugas NLP. *Transformers* mengandalkan mekanisme *self-attention*, yang memungkinkan model untuk memperhatikan setiap kata dalam sebuah kalimat, terlepas dari posisinya, sehingga hubungan antar kata dapat dipelajari dengan lebih baik.

BERT (*Bidirectional Encoder Representations from Transformers*), yang diperkenalkan oleh Devlin et al. (2018), merupakan salah satu varian *Transformers* yang paling populer dan efektif dalam berbagai tugas NLP, termasuk klasifikasi teks. Berbeda dengan model transformer lainnya seperti GPT yang *unidirectional*, BERT mampu memahami konteks kalimat dari kedua arah (*bidirectional*). Ini memungkinkan BERT untuk menangkap informasi konteks yang lebih kaya, yang sangat penting untuk tugas-tugas yang memerlukan pemahaman konteks penuh seperti deteksi *spam*.

2.4 Penerapan BERT dalam Klasifikasi Teks

BERT telah diterapkan secara luas dalam berbagai tugas klasifikasi teks, termasuk analisis sentimen, deteksi *spam*, dan analisis opini. Dalam tugas klasifikasi teks, BERT menggunakan proses tokenisasi untuk memecah teks menjadi token-token yang dapat diproses oleh model. BERT kemudian memproses token tersebut melalui beberapa lapisan *self-attention*, yang memungkinkan model untuk memperhatikan setiap kata dalam konteks kalimat penuh.

Dalam tugas deteksi *spam*, BERT dapat digunakan untuk mengidentifikasi pola-pola dalam teks yang secara tidak langsung menunjukkan bahwa pesan tersebut adalah *spam*. Misalnya, dengan memahami pola bahasa promosi, klaim yang berlebihan, atau kata-kata yang sering digunakan dalam skenario penipuan, BERT mampu melakukan klasifikasi yang lebih akurat dibandingkan dengan metode tradisional.

2.5 PyTorch dan Hugging Face Transformers

Dalam penerapan BERT, pustaka seperti PyTorch dan *Hugging Face Transformers* memegang peran penting. PyTorch adalah *framework deep learning* yang fleksibel dan mudah digunakan untuk membangun serta melatih model-model *neural networks*. PyTorch mendukung akselerasi GPU, yang memungkinkan percepatan proses pelatihan dan inferensi.

Hugging Face Transformers menyediakan akses ke berbagai model *pre-trained* seperti BERT, yang dapat langsung digunakan atau di-*fine-tune* untuk tugas-tugas tertentu. Dalam kasus deteksi *spam*, model BERT yang telah di-*pre-train* pada berbagai tugas NLP umum dapat diadaptasi atau di-*fine-tune* dengan dataset spesifik seperti *SMS Spam Collection*, untuk menghasilkan model yang mampu mendeteksi *spam* secara efektif.

2.6 Fine-tuning BERT untuk Deteksi Spam

Fine-tuning adalah proses di mana model *pre-trained* BERT diadaptasi pada dataset khusus untuk tugas tertentu, seperti deteksi *spam*. Dalam konteks ini, dataset *SMS Spam Collection* yang diperoleh dari *Kaggle* digunakan untuk melatih model BERT agar mampu mengklasifikasikan pesan teks sebagai *spam* atau *ham*. Proses *fine-tuning* ini biasanya lebih cepat dibandingkan melatih model dari awal, karena BERT sudah memiliki pemahaman dasar tentang struktur bahasa.

Selama *fine-tuning*, model dilatih untuk memahami pola-pola spesifik dalam teks pesan SMS, seperti kata-kata atau frasa yang lebih sering muncul dalam pesan *spam*dibandingkan dengan pesan *ham*. Setelah proses *fine-tuning*, model diharapkan mampu memberikan prediksi yang lebih akurat dalam mendeteksi pesan *spam*.

2.7 Optimasi Kinerja BERT

Menggunakan *Google Colab* dengan akselerasi GPU dapat mempercepat proses pelatihan dan inferensi pada model BERT, terutama untuk tugas yang memerlukan analisis sekuens panjang. Beberapa strategi optimasi yang dapat diterapkan antara lain:

1. **Reduksi Panjang Input**

   Mengurangi panjang urutan input yang diproses oleh BERT dapat membantu menurunkan beban komputasi. BERT memiliki kompleksitas komputasi yang meningkat dengan panjang sekuens.

2. **Penyesuaian Learning Rate**

   Pada pelatihan awal, *learning rate* yang lebih besar dapat digunakan untuk mempercepat konvergensi, namun secara bertahap, *scheduler* menurunkan *learning rate* agar model dapat melakukan penyesuaian yang lebih halus dan mengurangi risiko *overfitting*. Dengan menggunakan *learning rate scheduler*, performa model dapat ditingkatkan tanpa melampaui batas optimal pada setiap iterasi.

3. **Batching**

   Menggunakan batch yang lebih besar selama pelatihan dapat meningkatkan efisiensi dalam memanfaatkan GPU.

4. **Model Optimization**

   Teknik seperti *quantization* dan *model distillation* juga dapat digunakan untuk mengurangi ukuran model dan mempercepat inferensi, tanpa mengorbankan akurasi secara signifikan.

2.8 Evaluasi Kinerja Model

Dalam tugas deteksi *spam*, evaluasi kinerja model merupakan langkah penting untuk memahami seberapa baik model dalam mengklasifikasikan pesan sebagai *spam* atau *ham*. Beberapa metrik yang umum digunakan untuk mengevaluasi model klasifikasi adalah *accuracy*, *precision*, *recall*, dan *F1-score*. Setiap metrik ini memberikan wawasan yang berbeda mengenai performa model, terutama ketika ada ketidakseimbangan dalam distribusi kelas.

1. **Confusion Matrix**

   Evaluasi model klasifikasi biasanya dimulai dengan *confusion matrix*, yang merupakan tabel yang menampilkan jumlah prediksi yang benar dan salah untuk setiap kelas. Dalam kasus deteksi *spam*, *confusion matrix* dapat ditulis sebagai berikut:

	Predicted Spam	**Predicted Ham**
Actual Spam	True Positive (TP)	False Negative (FN)
Actual Ham	False Positive (FP)	True Negative (TN)

Table 1: Confusion Matrix

2. **Accuracy**

   Accuracy mengukur persentase prediksi yang benar dari seluruh prediksi yang dilakukan. Ini adalah metrik yang paling umum digunakan, tetapi bisa menjadi tidak memadai ketika terdapat ketidakseimbangan kelas (misalnya, jika sebagian besar pesan adalah ham).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- True Positive (TP): Pesan *spam* yang berhasil dideteksi sebagai *spam*.
- True Negative (TN): Pesan *ham* yang berhasil dideteksi sebagai *ham*.
- False Positive (FP): Pesan *ham* yang salah diklasifikasikan sebagai *spam* (*false alarm*).
- False Negative (FN): Pesan spam yang salah diklasifikasikan sebagai *ham*(*spam* terlewat).

3. **Precision**

   Precision mengukur persentase prediksi *spam* yang benar dari semua prediksi yang diklasifikasikan sebagai *spam*. *Precision* penting ketika *false positive* perlu diminimalkan, misalnya untuk menghindari kesalahan mengklasifikasikan pesan sah (*ham*) sebagai *spam*.

$$\text{Precision} = \frac{TP}{TP + FP}$$

   Precision tinggi menunjukkan bahwa ketika model memprediksi pesan sebagai spam, kemungkinan besar prediksinya benar.

4. **Recall (Sensitivity)**

   Recall mengukur persentase pesan spam yang benar-benar terdeteksi oleh model dari seluruh pesan yang sebenarnya adalah *spam*. Recall penting ketika kita ingin meminimalkan *false negative*, yaitu mencegah pesan *spam* lolos dari deteksi.

$$\text{Recall} = \frac{TP}{TP + FN}$$

   Recall tinggi menunjukkan bahwa model berhasil mendeteksi sebagian besar pesan spam.

5. **F1-Score**

   F1-Score adalah rata-rata harmonik dari *precision* dan *recall*. *F1-Score* memberikan keseimbangan antara *precision* dan *recall*, dan sangat berguna ketika terdapat ketidakseimbangan kelas, di mana salah satu dari *precision* atau *recall* mungkin lebih penting tergantung pada konteks.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

   F1-Score memberikan pandangan yang lebih seimbang tentang performa model, terutama ketika ada ketidakseimbangan antara spam dan ham dalam dataset.

3 Hasil

Pada penelitian ini, penulis menggunakan model BERT *pre-trained* untuk mengklasifikasikan pesan *spam* dan *ham* dengan bantuan pustaka PyTorch dan Transformers dari Hugging Face di lingkungan Google Colab. Bagian ini menjelaskan proses dari instalasi pustaka hingga evaluasi model, yang mencakup persiapan data, pelatihan, optimisasi, dan analisis hasil model. File *code* yang digunakan dalam penelitian ini dapat diakses di GitHub: https://github.com/alethat/MachineLearningTasks/.

3.1 Instalasi Pustaka dan Peran Setiap Pustaka

1. PyTorch

   Sebagai pustaka *deep learning*, PyTorch menyediakan kerangka kerja berbasis *tensor* untuk membangun dan melatih model *neural network*. Ini sangat penting dalam mengelola komputasi dan alokasi perangkat keras (seperti GPU) yang dibutuhkan oleh model BERT.

2. Hugging Face Transformers

   Pustaka ini menyederhanakan penggunaan model BERT yang telah dilatih sebelumnya untuk berbagai tugas NLP, termasuk klasifikasi teks. *Transformers* menyediakan antarmuka yang mudah digunakan untuk mengunduh model dan *tokenizer*, memungkinkan integrasi cepat untuk tugas-tugas NLP.

GPU diaktifkan di Google Colab untuk meningkatkan kinerja selama pelatihan dan inferensi model, karena GPU mempercepat komputasi yang intensif seperti operasi tensor besar yang digunakan oleh model BERT.

3.2 Memuat Model BERT dari Hugging Face Model Hub

Dengan pustaka PyTorch dan Transformers, penulis mengunduh dan menginisialisasi model 'bert-base-uncased' yang sudah di-*fine-tune* untuk klasifikasi biner.

Model BERT ini siap digunakan untuk memprediksi dua label, yaitu *spam* dan *ham*. Tokenizer mengubah teks input menjadi representasi token yang dapat diproses oleh model, serta menambahkan token `[CLS]` di awal dan `[SEP]` di akhir teks untuk menunjukkan keseluruhan urutan.

3.3 Mempersiapkan Data untuk Input BERT

Dataset yang digunakan berisi pesan teks yang diberi label sebagai *spam* atau *ham*. Setelah mengunggah dan membaca dataset, kolom `[label]` di-*map* ke `[0]` (*ham*) dan `[1]` (*spam*), kemudian data ini diubah menjadi format tensor untuk kompatibilitas dengan BERT. Tokenisasi akan menghasilkan `[input_ids]` dan `[attention_mask]` , di mana:

1. Padding

   Menambahkan token kosong agar semua input memiliki panjang yang sama, karena BERT membutuhkan urutan yang konsisten.

2. Attention Mask

   Mask ini membantu model untuk fokus pada token yang relevan dan mengabaikan token *padding*.

3.4 Membagi Dataset untuk Pelatihan dan Pengujian

Dataset dibagi menjadi 80% *training set* dan 20% *test set* menggunakan `[train_test_split]`. Selanjutnya, tokenizer diterapkan pada kedua set data ini.

3.5 Pelatihan Model Menggunakan Trainer

Model dilatih menggunakan API `[Trainer]` dari *Transformers*, yang mengelola proses pelatihan dan evaluasi. Konfigurasi hyperparameter meliputi jumlah epoch sebanyak 3, yang memungkinkan model belajar dari keseluruhan dataset dalam tiga kali putaran, dan *batch size* sebesar 16 untuk efisiensi memori dan kecepatan proses.

Setelah proses pelatihan selesai, `[TrainOutput]` menunjukkan beberapa hasil kinerja model sebagai berikut:

1. **Global Step**: 837, menunjukkan total langkah pelatihan yang diambil.
2. **Training Loss**: 0.0789, yang menunjukkan rata-rata kehilangan selama pelatihan. Nilai ini relatif rendah, menunjukkan bahwa model berhasil belajar pola dari data dengan baik.
3. **Train Runtime**: 575.7757 detik, yang merupakan waktu total pelatihan.
4. **Train Samples per Second**: 23.223, menunjukkan kecepatan pelatihan dalam jumlah sampel yang diproses per detik.
5. **Train Steps per Second**: 1.454, menunjukkan kecepatan pelatihan dalam langkah per detik.
6. **Total FLOPS (Floating Point Operations)**: 879.514 triliun, yang mengukur jumlah operasi komputasi selama pelatihan.

Dengan hasil ini, model telah melewati tiga *epoch* pelatihan dengan kehilangan yang rendah, dan hasil menunjukkan bahwa model telah dioptimalkan dengan efisien dalam hal waktu dan sumber daya komputasi.

3.6 Evaluasi dan Hasil Model

Setelah pelatihan, model dievaluasi menggunakan beberapa metrik utama untuk mengukur performanya dalam mengklasifikasikan pesan *spam* dan *ham*. Hasil evaluasi model berdasarkan *accuracy*, *precision*, *recall*, dan *F1-score* adalah sebagai berikut:

Metrik	Skor
Accuracy	0.993722
Precision	0.993743
Recall	0.993722
F1-Score	0.993731

Table 2: Hasil Evaluasi Model

Accuracy, *precision*, *recall*, dan *F1-Score* semuanya berada di sekitar 99.37%, yang menunjukkan bahwa model memiliki performa yang sangat baik dan konsisten dalam mengklasifikasikan pesan *spam* dan *ham* dengan tingkat kesalahan yang sangat rendah.

Selain itu, *confusion matrix* juga digunakan untuk memberikan gambaran lebih rinci terkait distribusi prediksi model pada masing-masing kelas. Berikut adalah *confusion matrix* hasil evaluasi model:

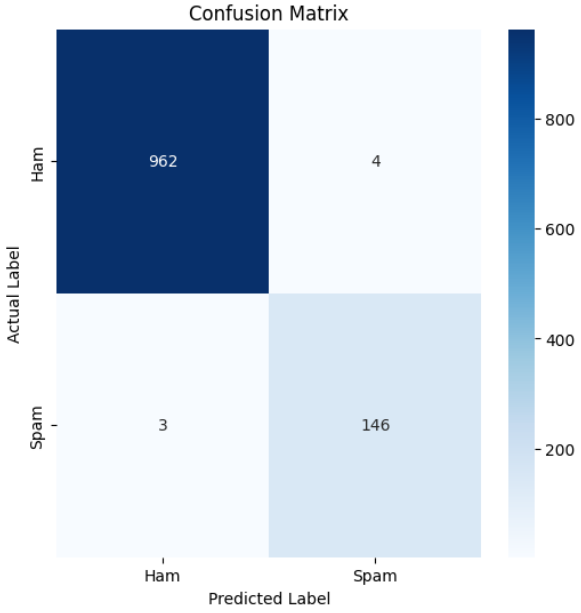

Figure 1: Confusion Matrix hasil prediksi model pada data uji

- Model berhasil mengklasifikasikan **962** pesan *ham* dengan benar, tetapi terdapat **4** pesan *ham* yang salah diklasifikasikan sebagai *spam*.
- Model juga berhasil mengklasifikasikan **146** pesan *spam* dengan benar, namun terdapat **3** pesan *spam* yang salah diklasifikasikan sebagai *ham*.

Secara keseluruhan, model menunjukkan performa yang sangat baik dengan kesalahan klasifikasi yang sangat sedikit pada kedua kelas, seperti terlihat pada Confusion Matrix. Tingkat kesalahan yang rendah ini mendukung nilai *accuracy*, *precision*, *recall*, dan *F1-score* yang tinggi (sekitar 99.37%).

3.7 Optimisasi Model

BERT dioptimalkan dengan penyesuaian *learning rate* dan penggunaan *scheduler* untuk memperbaiki hasil pelatihan. *Learning rate* mengatur seberapa besar langkah yang diambil model saat

memperbarui bobot di setiap iterasi pelatihan. Dalam kasus ini, *learning rate* dipilih kecil untuk memastikan pembelajaran yang stabil dan menghindari overshooting, yaitu ketika pembaruan bobot terlalu besar sehingga melampaui nilai optimal.

Penggunaan *scheduler* membantu menyesuaikan *learning rate* secara dinamis selama proses pelatihan. Scheduler yang digunakan menurunkan *learning rate* seiring bertambahnya epoch, yang membantu model mengkonsolidasi pembelajarannya dan mencegah terjadinya pembelajaran yang berlebihan (*overfitting*) di akhir pelatihan.

Setelah proses pelatihan selesai, `[TrainOutput]` menunjukkan beberapa hasil kinerja model sebagai berikut:

1. **Eval Loss**: 0.0569, menunjukkan bahwa rata-rata kehilangan (*loss*) selama evaluasi sangat rendah, yang mengindikasikan model telah belajar dengan baik dari data.
2. **Eval Runtime**: 7.93 detik menunjukkan waktu yang diperlukan untuk mengevaluasi model pada data uji.
3. **Eval Samples per Second**: menunjukkan bahwa model dapat memproses sekitar 140 sampel per detik selama evaluasi.
4. **Eval Steps per Second**: menunjukkan bahwa model melakukan sekitar 8.83 langkah per detik selama evaluasi.
5. **Epoch**: 3, menunjukkan bahwa model telah menyelesaikan pelatihannya pada tiga epoch.

Dengan hasil ini, model tidak hanya menunjukkan performa yang baik dari segi metrik evaluasi, tetapi juga efisiensi waktu dan sumber daya selama pelatihan dan evaluasi.

3.8 Evaluasi dan Hasil Model Setelah Optimisasi

Setelah dilakukan optimisasi, performa model dievaluasi kembali menggunakan beberapa metrik utama, yaitu *accuracy*, *precision*, *recall*, dan *F1-score*. Hasil evaluasi model setelah optimisasi ditampilkan pada tabel berikut:

Metrik	**Skor**
Accuracy	0.991928
Precision	0.991909
Recall	0.991928
F1-Score	0.991917

Table 3: Hasil Evaluasi Model Setelah Optimisasi

Terlihat bahwa semua metrik sedikit turun dari 99.37% menjadi 99.19%. Penurunan ini sangat kecil dan tidak signifikan, sehingga model masih memiliki performa yang sangat baik. Meskipun metrik seperti *accuracy*, *precision*, *recall*, dan *F1-score* sedikit menurun, optimisasi yang dilakukan bertujuan untuk meningkatkan stabilitas model dalam jangka panjang dan mencegah *overfitting*.

Untuk memberikan gambaran lebih jelas terkait distribusi prediksi model, berikut ditampilkan *confusion matrix* setelah optimisasi:

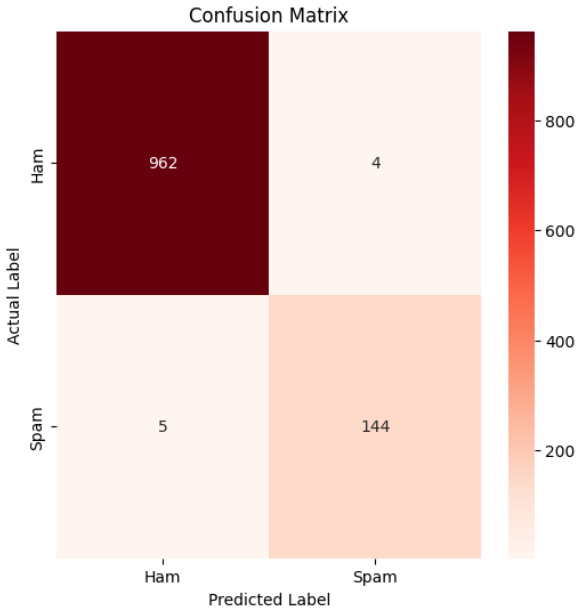

Figure 2: Confusion Matrix Setelah Optimisasi

Berdasarkan *confusion matrix*, model berhasil mengklasifikasikan hampir semua sampel dengan benar. Berikut adalah rincian hasil *confusion matrix*:

- Model mengklasifikasikan **962** pesan *ham* dengan benar, tetapi terdapat **4** pesan *ham* yang salah diklasifikasikan sebagai *spam*.
- Model mengklasifikasikan **146** pesan *spam* dengan benar, namun terdapat **3** pesan *spam* yang salah diklasifikasikan sebagai *ham*.

Secara keseluruhan, optimisasi memberikan dampak positif pada kestabilan dan generalisasi model tanpa mengorbankan performa. Meskipun ada sedikit penurunan pada nilai metrik, hasil akhir tetap menunjukkan bahwa model ini sangat efektif dalam melakukan klasifikasi dengan akurasi tinggi dan efisiensi waktu yang baik.

4 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa model BERT (*Bidirectional Encoder Representations from Transformers*) menunjukkan performa yang sangat baik dalam melakukan klasifikasi pesan teks, khususnya dalam deteksi *spam* dan *ham*. Melalui penerapan model BERT yang di-*fine-tune* menggunakan dataset *SMS Spam Collection*, model ini mampu mencapai tingkat akurasi, presisi, *recall*, dan *F1-score* yang sangat tinggi, yaitu sekitar 99.37% sebelum optimisasi dan 99.19% setelah optimisasi.

Optimisasi yang dilakukan, seperti penyesuaian *learning rate* dan penggunaan *scheduler*, terbukti dapat memperbaiki kestabilan model tanpa mengorbankan performa secara signifikan. Meskipun terdapat sedikit penurunan pada nilai metrik setelah optimisasi, hasil tersebut tetap berada pada tingkat yang sangat baik, yang menunjukkan bahwa model dapat melakukan generalisasi dengan lebih baik terhadap data uji serta menghindari masalah *overfitting*.

Evaluasi menggunakan *confusion matrix* menunjukkan bahwa model ini mampu mengklasifikasikan hampir semua pesan dengan benar, dengan kesalahan klasifikasi yang sangat sedikit. Model berhasil mendeteksi sebagian besar pesan *spam* dan *ham* dengan tingkat kesalahan yang rendah, baik sebelum maupun setelah optimisasi.

Secara keseluruhan, penelitian ini menunjukkan bahwa model BERT adalah pilihan yang sangat efektif untuk tugas deteksi *spam*, khususnya dalam pesan teks. Selain itu, penggunaan *deep learning*

dan teknologi *Transformers* terbukti mampu meningkatkan akurasi dan efisiensi dalam tugas klasifikasi teks yang lebih kompleks, melebihi metode tradisional seperti *Naive Bayes* dan *Support Vector Machines*.

Hasil ini juga memberikan wawasan bahwa penerapan teknologi *deep learning* yang canggih, seperti BERT, dapat secara signifikan meningkatkan kinerja sistem otomatis dalam memfilter dan mendeteksi pesan yang tidak diinginkan. Untuk penelitian selanjutnya, optimisasi lebih lanjut atau pengujian dengan dataset yang lebih besar dan beragam dapat dilakukan untuk memperkuat temuan ini dan memperluas aplikasinya ke berbagai domain klasifikasi teks lainnya.

References

[1] SMS Spam Collection Dataset. (2016, December 2). Kaggle. Retrieved from: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

[2] Ismail, S. S. I., Mansour, R. F., El-Aziz, R. M. A., & Taloba, A. I. (2022). Efficient E-Mail Spam Detection Strategy Using Genetic Decision Tree Processing with NLP Features. *Computational Intelligence and Neuroscience*, 2022. https://doi.org/10.1155/2022/7710005

[3] Amazon Web Services, Inc. *Apa itu Pemrosesan Bahasa Alami (NLP)?* Retrieved from: https://aws.amazon.com/id/what-is/nlp/

[4] Barrihadianto, N. (2023, December 17). Mengungkap Keterkaitan Transformer dengan NLP (Natural Language Processing). Medium. Retrieved from: https://noerbarry.medium.com/mengungkap-keterkaitan-transformer-dalam-dunia-pemrosesan-bahasa-alami-nlp-49cbe3829f16

[5] Zakaria, A., & Zakaria, A. (2024, June 13). Text Classification & Sentiment Analysis. Machine Learning Archive. Retrieved from: https://mlarchive.com/natural-language-processing/text-classification-sentiment-analysis/

[6] Som. (2022, March 26). Using Huggingface Transformers with PyTorch for NLP tasks. Medium. Retrieved from: https://someshfengde.medium.com/using-huggingface-transformers-with-pytorch-for-nlp-tasks-afc430190e22

[7] Fatyanosa, T. (2021, December 24). Fine-Tuning Pre-Trained Transformer-Based Language Model. Medium. Retrieved from: https://fatyanosa.medium.com/fine-tuning-pre-trained-transformer-based-language-model-c542af0e7fc1

[8] Esairina. (2021). Memahami Confusion Matrix: Accuracy, Precision, Recall, Specificity, dan F1 Score. Medium. Retrieved from: https://esairina.medium.com/memahami-confusion-matrix-accuracy-precision-recall-specificity-dan-f1-score-610d4f0db7cf