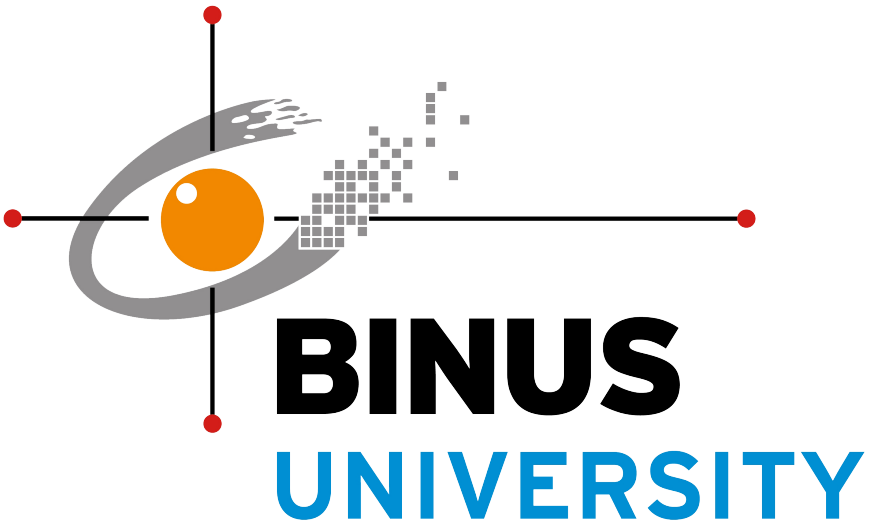
Klasifikasi Multilabel Artikel Ilmiah menggunakan *Scientific Bidirectional Encoder Representations from Transformers* (SciBERT)



Dibuat Oleh:
2440009093 - Grady Simanjaya
2440013071 - Felicia Ferren
2440015442 - Diana Petrina Santoso

Universitas Bina Nusantara
Jalan Kebon Jeruk Raya No. 27 Kebon Jeruk
Jakarta Barat 11530

1 Pendahuluan

1.1 Latar Belakang

Pembuatan artikel ilmiah tidak pernah berhenti. Bahkan, jumlah artikel ilmiah terus bertumbuh dengan cepat dari hari ke hari. Menurut Larsen dan Von, jumlah artikel ilmiah bertambah dua kali lipat setiap lima tahunnya. Sebagai contoh, pada tahun 2015, Ware dan Mabe menerbitkan sekitar 28.100 jurnal yang akhirnya menghasilkan 2.5 juta artikel penelitian setiap tahunnya. Sayangnya, mesin pencarian (*search engine*) tidak mampu mengklasifikasi tema yang tepat untuk setiap artikel penelitian berdasarkan kontennya. Sebagai buktinya, seringkali ketika seseorang mencari artikel berdasarkan sebuah *keyword*, hanya sedikit artikel yang relevan dengan *keyword* tersebut yang muncul dari sekian banyak artikel yang ditampilkan pada user. Kekacauan masif ini menjadi sebuah tantangan tersendiri bagi para peneliti dalam mengklasifikasi dokumen sedemikian rupa sehingga informasi yang relevan dapat diperoleh.

Salah satu cara untuk meningkatkan performa *search engine* terhadap pencarian artikel pada topik tertentu sekaligus untuk mengatasi masalah ini adalah dengan memberikan tag kategori yang tepat dan relevan untuk setiap artikel ilmiah, agar artikel-artikel ilmiah tersebut dapat dikelompokkan berdasarkan kategorinya. Setiap artikel dapat diklasifikasikan ke dalam satu atau lebih kategori/tema dengan menggunakan metode pada *Natural Language Processing* (NLP), yaitu *Multilabel Classification*, yang mampu memprediksi tema yang tepat untuk setiap artikel ilmiah dan mengelompokkannya berdasarkan tema yang relevan, dimana tema dari setiap artikel tidak terbatas hanya pada satu tema.

Banyak algoritma *Machine Learning* dan *Deep Learning* yang dapat digunakan untuk melakukan *multilabel classification* secara efisien, salah satunya adalah BERT (*Bidirectional Encoder Representations from Transformers*). BERT merupakan model *Deep Learning* berbasis *transformer* untuk NLP yang mampu menghasilkan representasi kata yang kontekstual dan mendalam. BERT menggunakan metode *unsupervised learning* dengan memanfaatkan korpus yang telah tersedia untuk dapat mempelajari representasi kata yang kontekstual dengan memprediksi kata yang hilang (*masked language model*) dan memprediksi apakah dua kalimat berurutan dalam teks, yang kemudian akan dilakukan *fine-tuning* untuk mengadaptasikan model tersebut sesuai dengan konteksnya secara spesifik, misalnya klasifikasi.

Model BERT memiliki beberapa pengembangan berdasarkan konteks datasetnya secara spesifik, salah satunya adalah SciBERT. SciBERT merupakan model pengembangan dari BERT yang secara khusus dilatih menggunakan korpus besar teks ilmiah sehingga dapat bekerja dengan sangat baik pada pemrosesan teks ilmiah.

Hal inilah yang menjadikan SciBERT sebagai model dalam project ini. Dataset yang akan digunakan merupakan dataset dari Janatahack: Independence Day 2020 ML Hackathon yang memuat teks ilmiah sebagai datanya. Dataset ini terdiri dari 20.972 non-null data yang berisikan *title* atau judul artikel beserta abstraknya, berdasarkan pengkategorian artikel bersarakan enam topik, yaitu *Computer Science, Physics, Mathematics, Statistics, Quantitative Biology,* dan *Quantitative Finance*. Persebaran data pada tiap kategori tidak merata, dimana hal ini dapat dilihat dari topik Computer Science yang memiliki lebih dari 8000 artikel, sedangkan jumlah artikel pada Quantitative Finance tidak lebih dari 500 artikel. Selain itu, diketahui juga bahwa mayoritas, sekitar 80% data, hanya memiliki 1 label. Dengan kata lain, observasi dengan klasifikasi lebih dari 1 label adalah tidak lebih dari 6000 paper.

Klasifikasi kategori artikel ini diharapkan dapat berkontribusi dalam memfasilitasi akses dan penelusuran artikel ilmiah sehingga dapat meningkatkan efisiensi dalam pencarian, pengelolaan, dan penyebaran artikel yang relevan dengan bidang minat peneliti maupun pembaca, yang kemudian diharapkan dapat membantu percepatan perkembangan di berbagai bidang ilmu.

2 Metodologi

2.1 Natural Language Processing (NLP)

Natural Language Processing (NLP) merupakan bagian dari *artificial intelligence* (AI) yang berfokus pada interaksi antara manusia dan komputer dimana software akan dapat memproses dan menghasilkan bahasa manusia secara alami. NLP mencakup serangkaian metode dan teknik untuk memproses, menganalisa, hingga memahami bahasa manusia, baik berupa teks maupun audio. Serangkaian metode tersebut mencakup *text processing*, *semantic analysis*, *topic prediction*, *text classification*, *information extraction*, dan lain sebagainya. Pada project ini, yang menjadi tujuan dari NLP yang dilakukan adalah *multilabel classification*. *Multilabel Classification* merupakan teknik dalam NLP yang mengelompokkan teks sesuai dengan label kategorinya, dimana setiap teks dapat memiliki lebih dari satu label.

Dalam proses NLP, input teks akan dibersihkan (*data cleansing*) dan kemudian diubah menjadi sebuah representasi struktural yang dipahami oleh komputer, melalui tahapan seperti normalisasi dan tokenisasi. Hasil dari proses ini akan berupa sebuah vektor numerik yang merepresentasikan teks tersebut berdasarkan konteks di sekitar tiap kata

pada dokumen. Vektor numerik ini kemudian akan digunakan pada proses vectorization agar makna kontekstual dan semantik dari input teks dapat dipahami oleh komputer.

2.2 Scientific Bidirectional Encoder Representations from Transformers (SciBERT)

Scientific BERT (SciBERT) merupakan pengembangan dari BERT (*Bidirectional Encoder Representations from Transformer*), model yang dikembangkan secara khusus untuk domain ilmiah, seperti penelitian biomedis, kecerdasan buatan, maupun ilmu komputer. SciBERT memanfaatkan teknik *transfer learning*, dimana model telah melalui proses pre-training terlebih dahulu pada data teks ilmiah yang berjumlah besar, misalnya seperti artikel ilmiah ataupun sumber ilmiah lainnya, dan kemudian dapat disesuaikan dengan tujuan yang lebih spesifik sesuai dengan keinginan dan kebutuhan user.

SciBERT melibatkan proses *Masked Word Prediction* (prediksi kata yang hilang) dan *Next Sentence Prediction* (prediksi kalimat selanjutnya) agar model dapat memahami konteks dan hubungan antar kata dalam sebuah teks ilmiah. Pemahaman model terhadap istilah teknis, struktur kalimat, maupun konteks domain khusus dalam dataset dapat ditingkatkan melalui proses *fine tuning*. *Fine tuning* dilakukan dengan melatih ulang model pada dataset yang lebih spesifik dalam domain yang diinginkan sehingga model dapat belajar representasi yang lebih spesifik dan sesuai dengan konteks data ilmiah yang digunakan.

SciBERT model telah dilengkapi dengan komponen dan teknik preprocessing dasar yang akan digunakan dalam NLP, seperti *stopwords removal*, *normalization* (stemming dan lemmatization), tokenization, hingga vectorization. Hal ini mempermudah dalam penyelesaian proses NLP karena tahapan-tahapan data preprocessing dasar telah disediakan oleh SciBERT secara default. Meskipun demikian, bila dirasa bahwa metode yang disediakan secara default oleh SciBERT kurang mampu dalam menangani raw input data, maka tahap preprocessing terhadap data mentah tersebut dapat dilakukan menggunakan metode-metode preprocessing lain yang bukan merupakan default method dari SciBERT model.

Berdasarkan penelitian yang dilakukan oleh Beltagy (2019), SciBERT telah terbukti efektif dalam berbagai tugas NLP di domain ilmiah. Dengan menggunakan pengetahuan yang ditanamkan dalam model saat di pretraining pada teks ilmiah, SciBERT mampu memahami istilah teknis, struktur kalimat yang kompleks, dan konteks domain khusus dalam teks ilmiah. Hal ini membuatnya menjadi alat yang berguna bagi para peneliti, praktisi, dan ahli di bidang ilmu pengetahuan dan teknologi.

2.3 Evaluasi Performa Model

Dalam evaluasi performa model, terdapat 4 istilah yang sering digunakan dalam perhitungan metrik untuk menentukan kebaikan performa model:

TP : Data positif yang diprediksi sebagai data positif (*True Positive*)
TN : Data negatif yang diprediksi sebagai data negatif (*True Negative*)
FP : Data negatif yang diprediksi sebagai data positif (*False Positive*)
FN : Data positif yang diprediksi sebagai data negatif (*False Negative*)

Perhitungan evaluasi performa model dapat dilakukan dengan mengukur *accuracy*, *precision*, *recall*, *F1-score* dan *loss*.

2.3.1 Accuracy

Accuracy merupakan persentase data yang diprediksi dengan tepat dari seluruh prediksi. Nilai accuracy dapat dihitung dengan menggunakan rumus sebagai berikut:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2.3.2 Precision

Precision merupakan persentasi data positif yang diprediksi dengan tepat dari seluruh prediksi positif. Nilai precision dapat dihitung dengan menggunakan rumus sebagai berikut:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

2.3.3 Recall

Recall merupakan persentase data positif yang diprediksi dengan tepat dari seluruh data positif. Nilai recall dapat dihitung dengan menggunakan rumus sebagai berikut:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

2.3.4 F1-Score

F1-Score merupakan rata-rata harmonic (*harmonic mean*) dari precision dan recall. Metrik ini memberikan nilai gabungan antara presisi dan recall, dan berguna ketika terdapat ketidakseimbangan kelas dalam dataset. Nilai F1-Score dapat dihitung dengan menggunakan rumus sebagai berikut:

$$F1Score = \frac{2 * precision * recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{4}$$

2.3.5 Loss

Loss merupakan nilai yang menggambarkan sejauh mana model melakukan kesalahan prediksi. Nilai ini digunakan selama proses training dan validation untuk mengoptimalkan parameter sehingga mispredikси dan misklasifikasi dapat diminimalisir. Nilai loss dihitung berdasarkan loss function yang digunakan. Pada project ini, loss function yang digunakan adalah *BCEWithLogitsLoss*. Menurut Jason (2020), *Binary Cross Entropy with Logits* (BCEWithLogitsLoss) efektif dalam *multilable classification task* karena loss function ini akan memperlakukan setiap kategori secara independen untuk menghasilkan nilai loss untuk setiap kategori secara terpisah dan memberikan bobot yang berbeda untuk setiap kategori sehingga permasalahan seperti imbalansi data dapat diatasi dengan baik.

2.4 Dataset

Dataset yang akan digunakan untuk membangun model ini merupakan data dari *Janatahack: Independence Day 2020 ML Hackathon*, dimana terdapat 3 dataset di dalamnya, yaitu *train.csv*, *test.csv*, dan *sample_submission.csv*. Dataset ini berisikan 'title' atau judul artikel, abstrak, beserta pengkategorian artikel berdasarkan enam topik, yaitu *Computer Science, Physics, Mathematics, Statistics, Quantitative Biology,* dan *Quantitative Finance*. Sayangnya, setelah dilakukan pengecekkan menggunakan *value_counts()*, diketahui bahwa sample_submission.csv dan test.csv tidak memberikan label pada observasinya. Oleh karena itu, project ini hanya akan menggunakan dataset dari *train.csv* yang memiliki 20.972 non-null observasi yang kemudian akan dibagi menjadi 80% data training, 10% data validation, dan 10% data testing. Dengan kata lain, data training akan terdiri dari 16778 observasi, sedangkan data validation dan data testing masing-masing akan memiliki 2097 observasi.

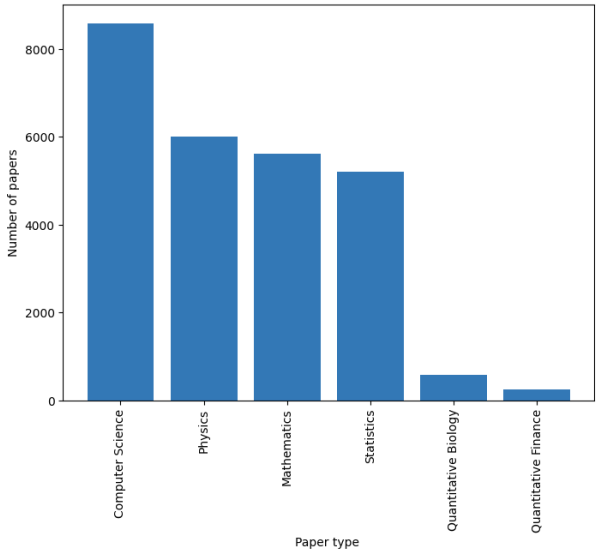


Figure 1: Persebaran Data per Label Kategori

Seperti yang ditunjukkan pada Figure 1, jumlah data antar topik tidak menyebar secara merata pada train.csv. Dari plot tersebut, dapat diketahui bahwa terjadi imbalansi data dimana topik Computer Science memiliki lebih dari 8000 artikel (sekitar 41% dari keseluruhan dataset), sedangkan jumlah artikel pada Quantitative Finance tidak lebih dari 500 artikel.

Selain itu, diketahui juga bahwa mayoritas, sekitar 80% data, hanya memiliki 1 label. Dengan kata lain, data yang memiliki lebih dari 1 label adalah tidak lebih dari 6000 paper, sedangkan sekitar 16000 paper lainnya masing-masing hanya memiliki 1 label.

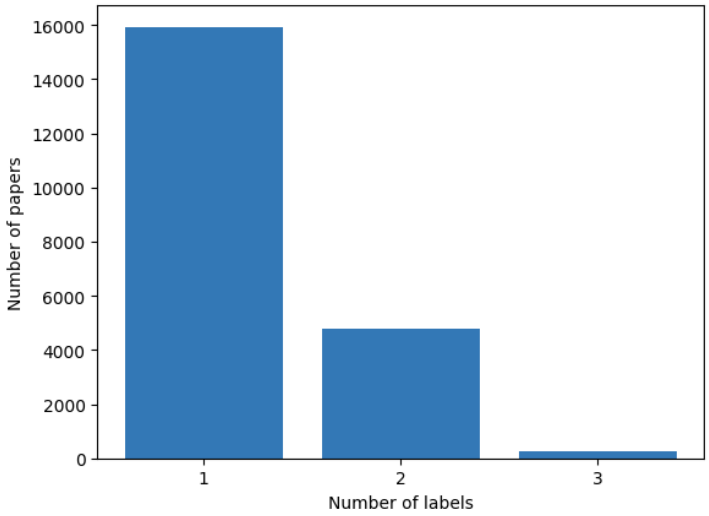


Figure 2: Jumlah Data berdasarkan Banyaknya Label per Artikel

Bila dilihat persebaran panjang teks dengan menggunakan histogram seperti yang ditunjukkan pada Figure 3, diketahui bahwa terdapat *skewness* di dalamnya, dimana mayoritas text menggunakan 100 hingga 200 kata. Jumlah kata paling banyak yang dimiliki oleh sebuah text tidak lebih dari 500 kata, sedangkan jumlah kata paling sedikit yang dimiliki text adalah kurang dari 50 kata. Dan jika diperhatikan dari grafik di bawah, diketahui bahwa hanya ada sangat sedikit sampel yang jumlah katanya lebih dari 400 kata, dan karena perbedaannya yang jauh dibandingkan mayoritas sampel lainnya, maka sampel-sampel ini dapat dianggap sebagai *outliers*.

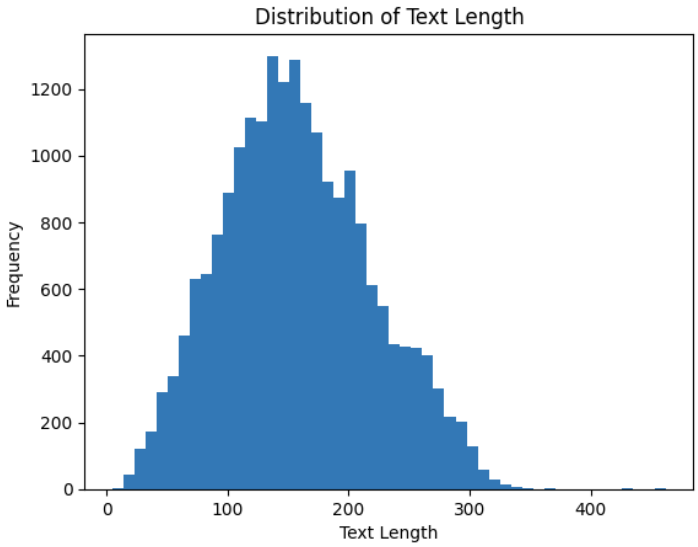


Figure 3: Text Length Distribution

2.5 Metode Penyelesaian Masalah

Alur dari penyelesaian project:

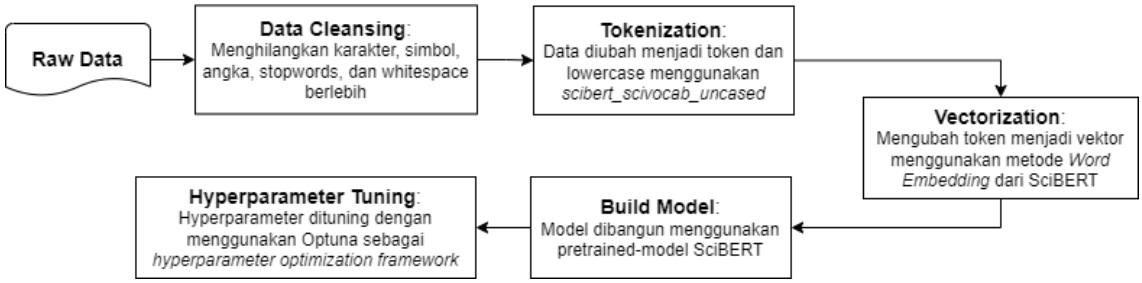


Figure 4: Grafik Metode Penyelesaian Masalah

Raw Data: Input data

Data Cleaning: Data dibersihkan dari simbol, karakter, angka, *stopwords*, dan *whitespace* berlebih yang tidak relevan

Tokenization: Dengan menggunakan *scibert_scivocab_uncased*, data dipisah per kata dan diubah menjadi lower case. Hasil dari pemecahan kata ini disebut token.

Vectorization: Mengubah data text menjadi vector representasi menggunakan metode *Word Embbeding* dari SciBERT

Build Model: Model dibangun dengan menggunakan *pretrained model* SciBERT karena kompatibilitasnya dengan artikel ilmiah

Hyperparameter Tuning: Metode Optuna digunakan sebagai *hyperparameter optimization framework* dalam project ini. Tahap ini dilakukan untuk meningkatkan kinerja model dan menghindari terjadinya *overfitting* maupun *underfitting*

2.6 Proses Penyelesaian Masalah

Seperti yang telah ditunjukkan pada Figure 4 pada section sebelumnya, project ini dimulai dari data preprocessing yang mencakup data cleansing, normalization, tokenization, hingga splitting data. Hasilnya kemudian akan digunakan untuk proses vectorization hingga modelling, yang kemudian model akan dituning untuk mendapatkan performa yang lebih baik. Pada section ini akan dijelaskan dengan lebih detail mengenai proses pada setiap tahapnya.

2.6.1 Data Preprocessing

Data Preprocessing merupakan serangkaian tahapan yang dilakukan pada data mentah (*raw input data*) sebelum data tersebut masuk dalam tahap pemrosesan lebih lanjut. Tahap preprocessing ini dilakukan untuk membersihkan dan mentransformasi sesuai kebutuhan agar siap digunakan pada model.

Tujuan lain dari data preprocessing adalah untuk mengantisipasi menurunnya kecepatan runtime yang dapat terjadi akibat banyaknya jumlah variabel yang digunakan. Variabel yang banyak dapat menurunkan kecepatan runtime karena pemrosesan variable tersebut akan meningkatkan penggunaan memori dan menyebabkan *overhead*. Oleh karena itu, pada project ini kolom *title* dan *abstract* digabungkan di langkah awal data preprocessing. Penggabungan antara *title* dan *abstract* ini juga diharapkan dapat lebih merepresentasikan kategori dari teks tersebut sehingga performa model dalam mengklasifikasi dapat meningkat.

Proses pembersihan data meliputi pembersihan dari tanda baca, karakter, simbol, angka, maupun *whitespcae* berlebih yang tidak relevan. Selain itu, penggunaan huruf kapital, *stopwords*, pengelompokkan kata (untuk tokenisasi), serta normalisasi (stemming dan lemmatization) juga perlu dipertimbangkan agar dapat menghasilkan hasil model yang terbaik dan akurat.

Tokenisasi merupakan pembagian teks menjadi unit yang lebih kecil, yaitu kata atau frasa, yang disebut token. Tokenisasi merupakan proses penting dalam SciBERT karena proses yang dilakukan di SciBERT merupakan pemrosesan teks pada tingkat token. Kemudian, untuk mengurangi variasi token, token-token tersebut akan dikembalikan ke bentuk dasarnya melalui proses stemming dan lemmatization. Pada project ini, proses stemming dan lemmatization akan

menggunakan default dari model SciBERT, sehingga tidak akan dilakukan proses tambahan khusus untuk stemming dan lemmatization.

Seperti yang telah diketahui dari Pendahuluan, bahwa data yang akan digunakan hanya berasal dari *train.csv*. Oleh karena itu pada tahap preprocessing ini, akan dilakukan pula pembagian data–data dalam *train.csv* menjadi 80% data training, 10% data validation, dan 10% data testing. Atau dengan kata lain, data training akan terdiri dari 16778 observasi, sedangkan data validation dan data testing masing-masing akan memiliki 2097 observasi.

2.6.2 Vectorization

Vectorization merupakan proses mengubah data non-numerik menjadi sebuah vektor numerik yang merepresentasikan teks berdasarkan konteks di sekitar tiap kata pada dokumen. Proses vectorization ini memungkinkan mesin untuk memahami dan memproses teks dengan lebih terstruktur dan terukur sehingga memungkinkan komputer untuk melakukan tugasnya, seperti klasifikasi, analisa sentimen, dan lain sebagainya.

Ada berbagai metode vectorization yang dapat digunakan dalam NLP, seperti *Bag of Words* (BoW), *Term Frequency-Inverse Document Frequency* (TF-IDF), *Word Embeddings* (misalnya Word2Vec, GloVe), dan *Transformer-Based Models* (misalnya SciBERT). Pada project ini, karena model akan dibangun dengan menggunakan SciBERT, maka metode vectorization yang digunakan adalah metode vectorization default dari model tersebut, yaitu SciBERT. Proses vectorization merupakan komponen penting dalam SciBERT karena semua model BERT memanfaatkan mekanisme *attention* untuk menangkap informasi kontekstual dari konteks di sekitar kata dalam teks.

Proses vectorization akan dilakukan setelah proses tokenisasi karena proses ini membutuhkan token-token tersebut sebagai inputnya. SciBERT akan memberikan *embedding* pada setiap token untuk mengkonversi setiap token menjadi sebuah vektor representasi, dimana proses ini akan menangkap makna kontekstual dan semantik dari token-token tersebut. Embedding dapat mencakup makna kata maupun hubungan antar kata, dengan kata lain, embedding mencerminkan pemahaman konteks dan representasi kata yang ada pada model. Setelah pemberian embedding pada token, maka akan terbentuklah matriks vektor representasi yang berisi vektor-vektor representasi yang kemudian akan menjadi input ketika pembangunan model.

2.6.3 Modelling

Dalam konteks NLP, modelling merupakan proses pembuatan dan pengembangan model komputasional yang dapat memahami dan memproses teks dengan cara yang mirip dengan kemampuan manusia. Model yang digunakan dalam project ini adalah SciBERT model. SciBERT merupakan salah satu *language-pretrained model* yang dikembangkan dari arsitektur BERT (*Bidirectional Encoder Representations from Transformers*) yang secara spesifik dibuat untuk memahami teks ilmiah karena model ini dilatih menggunakan corpus teks ilmiah dan teknis.

SciBERT menggunakan metode *multilayer bidirectional transformer*, dimana arsitektur ini terdiri dari beberapa layer dan menggunakan mekanisme *attention* serta representasi vektor yang mengalir secara dua arah (*bidirectional*). Hal ini membuat SciBERT tidak hanya mampu melakukan *scanning* terhadap data, namun juga mampu memahami konteks dan hubungan antar kata pada data dengan lebih baik.

Bila melihat kembali pada Figure 3, diketahui bahwa jumlah kata yang digunakan dalam sebuah artikel tidak lebih dari 320 kata. Oleh karena itu, parameter *max_length* pada model akan diatur sebesar 320. Pengaturan ini diperlukan karena mdoel SciBERT memiliki batas maksimum panjang sequence yang dapat ditangani. Sehingga, semua input sequence akan mengelami penyesuaian panjang kata, dimana akan dilakukan *truncation* (pemendekkan/pemotongan panjang kata) untuk teks yang memiliki jumlah kata yang lebih dari 320, dan penambahan *padding* untuk teks yang memiliki jumlah kata lebih sedikit dari 320.

Kemudian, model yang dibangun akan ditraining dan divalidasi dengan menggunakan Sigmoid sebagai *activation function* dan parameter seperti *learning rate* akan diperbarui dan dioptimasi pada setiap epoch berdasarkan perhitungan *gradient loss* pada proses *backpropagation*.

AdamW (*Adam with Weight Decay*) akan digunakan sebagai *optimizer* karena metode ini akan menurunkan bobot (*weight decay*) kepada parameter selama proses update nilai parameter. Penggunaan *weight decay* ini akan mencegah terjadinya overfitting dan meningkatkan generalisasi model. Dan untuk *loss function*, akan digunakan BCEWithLogitsLoss (*Binary Cross-Entropy with Logits Loss*). BCEWithLogitsLoss akan menghitung nilai loss berdasarkan hasil prediksi logits dengan label targetnya.

2.6.4 Hyperparameter Tuning

Hyperparameter merupakan parameter pada model yang digunakan untuk mengatur dan mengendalikan kinerja model dalam proses belajar dan mengeluarkan prediksi. Hyperparameter dapat berupa *learning rate*, jumlah epoch, jumlah neuron (*hidden layer*), *dropout rate*, dan banyak parameter lainnya. Namun pada project ini, hyperparameter yang akan dituning hanya mencakup *learning rate*, *optimizer*, dan *weight_decay*.

Dari penjelasan mengenai hyperparameter tersebut, maka dapat disimpulkan bahwa *Hyperparameter Tuning* merupakan proses pencarian kombinasi hyperparameter model yang optimal untuk mendapatkan kinerja terbaik dari model ataupun algoritma tersebut. Ada beberapa metode yang dapat digunakan untuk mencari kombinasi hyperparameter, misalnya GridSearchCV, Optuna, dan lain sebagainya.

Pada project ini, metode hyperparameter tuning yang akan digunakan adalah Optuna. Menurut Akiba (2019), Optuna merupakan *hyperparameter optimization framework* yang sangat dapat diandalkan, terutama dalam model deep learning, karena memiliki prinsip *define-by-run* yang memungkinkan user untuk membuat *search space* secara dinamis dan efisien dalam *searching and pruning strategies*. Dengan menggunakan pendekatan optimasi Bayesian, Optuna mampu memberikan hasil yang lebih baik dengan menggunakan jumlah percobaan yang lebih sedikit dibandingkan GridSearchCV, sehingga dapat menghemat waktu dan sumber daya komputasi. Dan karena kemampuannya yang mampu mengeksplorasi dan mengeksploitasi dalam ruang pencarian hyperparameter dengan lebih cepat, Optuna mampu bekerja dengan baik untuk hyperparameter tuning, bahkan yang kompleks sekalipun.

Penggunaan Optuna dimulai dengan menentukan *objective function* yang akan dioptimasi, dimana fungsi ini akan mengembalikan sebuah nilai yang akan digunakan Optuna untuk membandingkan dan memilih kombinasi hyperparameter yang lebih baik. Kemudian, dibuat pula *search space* yang berisi hyperparameter yang ingin dioptimasi. Selanjutnya, dengan menggunakan *looping*, Optuna akan mengevaluasi kombinasi hyperparameter, mencatat hasilnya, kemudian menggunakan informasi tersebut untuk mencari kombinasi hyperparameter yang memberikan hasil yang lebih baik.

Nilai masing-masing hyperparameter dari kombinasi terbaik itu kemudian akan digunakan dalam membangun ulang model. Dengan kata lain, setelah dilakukan proses hyperparameter tuning, model kembali dibangun, dilatih, dan divalidasi dengan menggunakan nilai parameter yang didapat dari proses *hyperparameter optimization* menggunakan Optuna.

Performa model tersebut kemudian dibandingkan dengan performa model sebelum tuning untuk melihat apakah ada perbaikan performa model. Sehingga dengan demikian, model yang memberikan performa terbaik akan diambil sebagai *best_model* yang kemudian diuji kestabilan performanya menggunakan data testing. Apabila hasil yang didapat konsisten dengan hasil training dan validation sebelumnya, maka model tersebut dapat dan siap digunakan untuk memprediksi dan mengklasifikasi data input baru.

3 Hasil dan Analisa

3.1 Data Preprocessing

Sebelum data digunakan lebih lanjut dalam pembuatan model, input data akan melalui proses *cleansing*, dimana semua simbol, tanda baca, angka, dan *multiple whitespace* akan dihapus. Berikut adalah contoh perbandingan data sebelum dan sesudah melalui tahap *cleansing*:

```
Rotation Invariance Neural Network  Rotation invariance and translation invariance have great values in image
recognition tasks. In this paper, we bring a new architecture in convolutional
neural network (CNN) named cyclic convolutional layer to achieve rotation
invariance in 2-D symbol recognition. We can also get the position and
orientation of the 2-D symbol by the network to achieve detection purpose for
multiple non-overlap target. Last but not least, this architecture can achieve
one-shot learning in some cases using those invariance.
```

Figure 5: Before Cleansing Process

```
'Rotation Invariance Neural Network Rotation invariance and translation invariance have great values in image
recognition tasks In this paper we bring a new architecture in convolutional neural network CNN named cyclic c
onvolutional layer to achieve rotation invariance in D symbol recognition We can also get the position and ori
entation of the D symbol by the network to achieve detection purpose for multiple non overlap target Last but
not least this architecture can achieve one shot learning in some cases using those invariance '
```

Figure 6: After Cleansing Process

3.2 Evaluasi Model

Untuk menentukan model terbaik, maka perlu dilakukan perbandingan performa model sebelum dan sesudah dilakukannya hyperparameter tuning. Perbandingan evaluasi performa model dilakukan dengan membandingkan *Training Loss*, *Validation Loss*, *Validation Accuracy*, serta *Validation Micro F1 Score*. Hasil perbandingan performa kedua model ditampilkan pada Table 1.

Table 1: Hasil Evaluasi Performa Model Sebelum dan Sesudah Tuning

Model	Training Loss	Val Loss	Val Accuracy	Val Micro F1 Score
Before Tuning	0.1833	0.1560	0.7077	0.8476
After Tuning	0.3217	0.2558	0.6381	0.7897

Jika dilihat dari segi *training loss* dan *val loss*, model sebelum tuning menunjukkan nilai yang lebih rendah dibandingkan model setelah melalui proses tuning. Selain itu, *val accuracy* dan *val micro f1 score* dari model sebelum tuning juga lebih tinggi dibandingkan model setelah tuning. Oleh karena itu, dapat disimpulkan bahwa model sebelum tuning memberikan performa yang lebih baik dibandingkan model tuning. Hal ini dapat terjadi karena kurangnya kombinasi parameter yang dituning. Untuk memperbaiki akurasi model, hal yang dapat dilakukan adalah mengoptimalkan parameter yang digunakan, misalnya dengan menambah kombinasi parameter yang dituning atau memberikan pilihan nilai parameter yang lebih bervariasi.

Berdasarkan hasil evaluasi pada Tabel 1, maka model yang akan digunakan untuk proses *testing* adalah model sebelum tuning karena memberikan performa yang lebih baik. Ketika dimasukkan data testing pada model, maka model mampu menjaga performanya yang dapat dilihat dengan akurasi yang tidak jauh berbeda dengan akurasi ketika dilakukan model validation, yaitu sekitar 70%.

Table 2: Hasil Testing Model

Accuracy	Precision	Recall	Average F1 Score
0.7029	0.8228	0.7764	0.7970

Angka ini menunjukkan bahwa model yang dibangun telah cukup baik dalam melakukan tugasnya untuk memprediksi dan mengklasifikasi. Argumen ini diperkuat dengan hasil dari F1-Score per kategori yang ditunjukkan pada table 3. Hasil dari F1-Score per kategori yang ditunjukkan pada Table 3 menjelaskan bahwa model mampu mengklasifikasikan artikel yang termasuk dalam kategori Computer Science, Physics, Mathematics, Statistics, dan Quantitative Finance dengan baik. Atau dengan kata lain, model memiliki performa yang baik dalam mengenali dan memprediksi artikel atau teks yang berkaitan dengan kelima topik tersebut. Sayangnya, terdapat 1 kategori yang cenderung sulit dikenali oleh model, yaitu Quantitative Biology. Hal ini dapat terlihat dari nilai F1-Score nya yang hanya sebesar 0.5439. Ada beberapa kemungkinan yang dapat menyebabkan hal ini terjadi, misalnya karena jumlah datanya yang sedikit dibandingkan dengan kategori lain (imbalance data), kompleksitas kategori yang lebih tinggi, ataupun karena karakteristik dan atributnya mirip dengan kateogri lain sehingga tingkat kesulitan untuk membedakan artikel pada Quantitative Biology lebih tinggi dibandingkan pada kategori lainnya.

Table 3: F1 Score untuk Setiap Kategori

Kategory	F1 Score
Computer Science	0.8524
Physics	0.8913
Mathematics	0.8257
Statistics	0.8354
Quantitative Biology	0.5439
Quantitative Finance	0.8333

4 Kesimpulan

Multilable Classification Task, khususnya dalam konteks ilmiah, dapat diselesaikan menggunakan *pretrained-model* SciBERT. Dengan menggunakan data dari *Janatahack: Independence Day 2020 ML Hackathon*, model yang dibangun

mampu memberikan akurasi 70% dengan *loss* dibawah 0.2. Setelah dilakukan hyperparameter tuning, performa model menurun, dimana hal ini ditandai dengan menurunnya nilai akurasi dan meningkatnya nilai loss, baik pada training maupun validation. Kurang optimalnya kombinasi nilai-nilai parameter yang digunakan dapat menyebabkan hal ini terjadi. Oleh karena itu, disarankan untuk ke depannya dapat memperbanyak dan mengoptimalisasi kombinasi dan range nilai parameter tuning.

Karena performa model setelah tuning menurun, maka model yang dipakai merupakan model awal yang tidak melalui proses hyperparameter tuning. Dan dari hasil F1-Score per kategori, dapat disimpulkan bahwa model mampu mengenali dan memprediksi teks yang berkaitan dengan *Computer Science, Physics, Mathematics, Statistics,* dan *Quantitative Finance*. Sayangnya, model cenderung sulit untuk mengenali dan memprediksi teks yang berkaitan dengan *Quantitative Biology*. Hal ini bisa terjadi mengingat adanya imbalance data dan kemungkinan bahwa kompleksitas kata yang digunakan pada kategori ini lebih tinggi dibandingkan kategori lain. Untuk meningkatkan kemampuan model, salah satu hal yang dapat dilakukan adalah menambah jumlah data pada kategori ini.

Akhir kata, dari project ini dapat disimpulkan bahwa, tanpa melaui proses hyperparameter tuning, model SciBERT telah memberikan performa yang cukup baik dalam melakukan klasifikasi multilabel pada dataset Janatahack: Independence Day 2020 ML Hackathon dengan akurasi lebih dari 70%. Argumen ini diperkuat dengan hasil F1-Score per kategori yang menyatakan bahwa, secara umum, model mampu mengenali dan memprediksi dengan baik yang berkaitan dengan kategori-kategori yang terdapat dalam dataset.

5 Referensi

Akiba, T., Sano, S., & Yanase, T. (2019). Optuna: A Next-Generation Hyperparameter Optimization Framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2623-2631).

Beltagy, I., Peters, M. E., & Cohan, A. (2019). SciBERT: Pretrained Contextualized Embeddings for Scientific Text. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 4171-4186).

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 13(Feb), 281-305.

Bhatia, P., & Jain, R. (2021). Multilabel Classification. In Text Analytics (pp. 205-228). Springer.

Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.

Brownlee, Jason. (2020). A Gentle Introduction to Multi-Label Classification. Machine Learning Mastery.

Hodgson, A. & Schlager, L. (2017). Closing the pdf gap: Readcube's Experiments in Reader-Focused Design. Learn. Publ. 30, 875–880. https://doi.org/10.1002/leap.1084.

I. Beltagy, K. Lo, A. Cohan. (2019). "SciBERT: A Pretrained Language Model for Scientific Text". https://arxiv.org/abs/1903.10676

J. Devlin, M. Chang, K. Lee, K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". https://arxiv.org/abs/1810.04805

Larsen, P. & Ins, M. V. (2010). The Rate of Growth in Scientific Publication and The Decline in Coverage Provided by Science Citation Index. Scientometrics 84, 575–603. https://doi.org/10.1007/s11192-010-0202-z.

O. Sokolova and G. Lapalme. "A Systematic Analysis of Performance Measures for Classification Tasks." Information Processing & Management, 45(4):427-437, 2009

Tsoumakas, G., & Katakis, I. (2007). Multi-label Classification: An Overview. International Journal of Data Warehousing and Mining, 3(3), 1-13.

Ware, M. & Mabe, M. (2015). The STM Report: An Overview of Scientific and Scholarly Journal Publishing. Int. Assoc. Sci. Techn. Med. Publ. 4, 1175–1356.