Adaptive Sparse Transformer untuk Meningkatkan ROUGE-1 Score pada Text Summarization Scientific Paper

Andrew Firman Saputra, Liliana, Djoni Haryadi Setiabudi
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) - 8417658
E-mail: andrewfirmansap@gmail.com, lilian@petra.ac.id, djonihs@peter.petra.ac.id

ABSTRAK

Perkembangan teknologi dan internet menyebabkan banyaknya informasi yang dapat diakses pada setiap waktu. Artikel jurnal merupakan salah satu dari banyaknya informasi yang tersedia yang membutuhkan waktu untuk dibaca sehingga diperlukan ringkasan secara otomatis. *Automatic Text Summarization* (ATS) pada dasarnya adalah pembuatan kalimat baru yang lebih pendek dari pada kalimat aslinya tanpa menghilangkan makna keseluruhan dari sebuah kalimat. Pembuatan ringkasan secara otomatis tersebut dapat dilakukan secara ektraktif maupun abstratif. Ringkasan yang dibuat secara ektraktif hanya dapat menggunakan kata yang ada pada bagian masukan, sedangkan ringkasan yang dibuat secara abstratif dapat menghasilkan ringkasan yang memiliki kata diluar dari kata yang ada pada bagian masukkan. Penelitian sebelumnya dalam *abstractive summarization* masih belum optimal sehingga perlu ditingkatkan.

Metode yang digunakan dalam penelitian ini adalah *abstractive summarization* dengan *Adaptive Sparse Transformer*. Hal-hal yang akan dilakukan pada penelitian adalah *scraping dataset arxiv machine learning*, pembuatan data, pengolahan *data* dan percobaan terhadap pengaturan *hyperparameter* pada *model* terhadap performa ROUGE-1 *precision score*. Dataset yang digunakan adalah *dataset Arxiv Scientific Paper* dan *Arxiv Scientific Paper+Arxiv Machine Learning*.

Hasil penelitian ini menunjukkan bahwa metode yang digunakan mampu bersaing dengan metode-metode terunggul dengan model nilai *average* R-1 *precision score* 39.4 untuk *Arxiv Scientific Paper+Arxiv Machine Learning* dan 42.5 untuk *Arxiv Scientific Paper*.

Kata Kunci: peringkasan kalimat, *deep learning, transformer, encoder, decoder*

ABSTRACT

Technology advancement and internet causes lots of information that can be accessed at any time. Journal article is one of such many information that's available that requires time to read thereof in need of automatic summary. Automatic Text Summarization (ATS) basically a process of making a new text that's smaller than the original text without removing the meanings from the entire input text. The process of making automatic text summarization can be done in extractive and abstractive way. A summary that was made by an extractive method only able to generate a summary with a word that's included in the original text, whereas summary that was made by an abstractive method can generate a summary that include word that does not exist in the original text. In the previous research in abstractive summarization is found is not optimal thereof need an improvement.

The method used in this research is an abstractive summarization with Adaptive Sparse Transformer. Things that will be done in this research are scraping dataset arxiv machine learning, making the dataset, processing the data and trials on hyperparameter configuration in the model to see ROUGE-1 precision performance. The dataset used is Arxiv Scientific Paper dataset and Arxiv Scientific Paper+Machine Learning dataset.

The results of this research showed that the method used capable to compete with state of the art methods with average R-1 precision score of 39.4 for Arxiv Scientific Paper+MachineLearning and 42.5 for Arxiv Scientific Paper.

***Keywords*:** *text summarization, deep learning, transformer, encoder, decoder*

1. PENDAHULUAN

Pada proses membaca sebuah jurnal ilmiah, terkadang pembaca ingin segera mengetahui intisari dari suatu jurnal dikarenakan pembaca memiliki waktu yang terbatas dalam membaca jurnal sedangkan publikasi-publikasi baru semakin bertambah dengan cepat [4]. Pembaca ingin tahu apakah jurnal yang akan dibaca memiliki topik yang sesuai dan dibutuhkan sehingga diperlukan ringkasan. Ringkasan dapat dilakukan secara manual oleh tenaga manusia tetapi hal tersebut sangatlah tidak efektif dan membutuhkan *resources* yang besar sehingga diperlukan peringkasan secara otomatis [4].

Pada artikel ilmiah atau *scientific paper* terdapat beberapa bagian. Salah satu bagian dari artikel ilmiah tersebut adalah bagian *abstract* dan bagian *introduction*. Bagian *abstract* merupakan bagian dari artikel ilmiah yang merupakan paragraf pendek yang berisikan garis besar dari sebuah artikel ilmiah dan pada bagian *introduction* dijelaskan tentang subjek umum dari penelitian yang dilakukan.

ROUGE atau *Recall-Oriented Understudy for Gisting Evaluation* adalah salah satu cara untuk melihat kualitas dari sebuah ringkasan dengan cara membandingkan hasil ringkasan otomatis yang dibuat oleh komputer terhadap hasil ringkasan ideal yang dibuat oleh manusia. Penilaian tersebut dilakukan dengan cara melihat jumlah *overlapping* dari *n-gram, word sequences* dan *word pairs* pada hasil ringkasan otomatis dan hasil ringkasan yang dibuat oleh manusia.

Pembuatan ringkasan atau *Automatic Text Summarization* (ATS) pada dasarnya adalah pembuatan kalimat baru yang lebih pendek dari pada kalimat aslinya tanpa menghilangkan makna keseluruhan dari sebuah kalimat. ATS merupakan salah satu masalah dari bidang *Natural Language Processing* yang dapat digolongkan menjadi 2 tipe yaitu *Extractive summarization* dan *Abstractive summarization*. *Extractive summarization* adalah peringkasan di

mana kata yang digunakan pada hasil ringkasan berasal dari kata yang ada pada kalimat inti yang akan diringkas, sedangkan *Abstractive summarization* menghasilkan hasil ringkasan di mana kata yang dipakai untuk menyusun ringkasan dapat berasal di luar kalimat inti yang dipakai [11].

Pada penelitian-penelitian terdahulu, *Text summarization* telah dilakukan dengan metode *Reinforcement Ranking* pada *Semantic Link Network* dalam peringkasan jurnal ilmiah secara *Extractive* dengan nilai ROUGE-1 *score* tertinggi 0.63336 [9]. Pendekatan metode *Restricted Boltzmann Machine* dan *Deep Learning* pada peringkasan kalimat Berbahasa Inggris secara *Extractive* dengan nilai *precision* rata-rata 0.7 dan nilai *recall* 0.63 [11]. Pendekatan *Transformer-based model* SciBERT dan PageRANK pada *summarization* jurnal ilmiah Berbahasa Inggris secara *Extractive* dengan nilai ROUGE *precision* 0.807 [6]. Pendekatan kombinasi *Hierarchical Seq2seq Sentence Pointer* dan *Transformer Language Model* untuk melakukan *summarization document* secara *Extractive* dan *Abstractive* dengan nilai ROUGE-1 sebesar 0.607 untuk *Extractive summarization* dan nilai ROUGE-1 sebesar 0.204 untuk *Abstractive summarization* [8]. Pada penelitian-penelitian tersebut terlihat bahwa nilai tertinggi ROUGE untuk *extractive summarization* adalah 0.807 dan *abstractive summarization* adalah 0.204. Pada salah satu NLP task bidang *Machine Translation*, penelitian yang berjudul *Adaptive Sparse Transformer* yang dilakukan oleh Correia et al dengan melakukan modifikasi pada bagian *softmax* menjadi α-*entmax* pada bagian *multi-head layer* dalam arsitektur *Transformer* [3][10].

Pada penelitian ini akan dilakukan penelitian tentang *Text summarization* pada jurnal ilmiah Berbahasa Inggris dengan *model Transformer*. Pengembangan pada penelitian dibandingan penelitian sebelumnya adalah peningkatkan ROUGE-1 *score* hasil ringkasan otomatis untuk *abstractive text summarization*. Pengembangan yang dilakukan adalah dengan menggunakan arsitekur *Adaptive Sparse Transformer* [3] yang sebelumnya digunakan untuk masalah *Translation* atau penerjemahan yang pada penelitian ini digunakan untuk masalah *Text summarization*

2. PENELITIAN SEBELUMNYA

Pada penelitian ini akan digunakan beberapa penelitian terdahulu pada bidang *text summarization sebagai* tinjauan studi. Berikut merupakan penelitian-penelitian pada bidang *text summarization*:

2.1 *Summarization of Scientific Paper Through Reinforcement Ranking on Semantic Link Network*

Sun & Zhuge [9] melakukan penelitian pada bidang NLP dengan tujuan melakukan *summarization* pada *scientific paper* secara *ektraktive* dengan menggunakan *reinforcement learning* dengan cara membuat *is-part-of link* dan *co-occurence link* diantara hubungan antara kata, paragraf dan bagian *section* pada *scientific paper*. Hubungan tersebut akan dibuat menjadi *semantic link* yang kemudian dihitung *top-k ranked* yang akan dijadikan *summary* dari source *paper*.

2.2 *SciSummPip: An Unsupervised Scientific Paper Summarization Pipeline*

Ju et al [6] melakukan penelitian pada bidang NLP dengan penerapan *pretrained model* SciBERT dan PageRANK untuk melakukan *summarization* secara *extractive* pada *scientific paper*. Pada bagian *embedding* digunakan *sentence embedding* dan pada bagian *text generation* diterapkan *spectral clustering* dan *multi-sentence clustering*. Pada penelitian ini ditunjukkan *model* dengan *pretrained SciBERT* mampu meningkatkan performa dan melalui *manual inspection model* dapat mendapatkan informasi penting dari *source text*.

2.3 *Extractive Summarization using Deep Learning*

Verma, S. & Nidhi, V. [11] melakukan penerapan dengan metode *Restricted Boltzmann Machine* dan *Deep Learning* untuk meningkatkan akurasi dari hasil ringkasan yang dibuat dalam *summary generation* akibat masalah utama pada *extractive summarization* 3 masalah utama yaitu *word ranking problem*, yaitu bagaimana kata akan di*rank* untuk mengetahui kata mana yang lebih penting dibandingkan kata lainnya. Masalah kedua adalah masalah *word selection*, yaitu pemilihan kata mana yang akan digunakan dalam melakukan ringkasan. Masalah terakhir adalah *coherence problem*, yaitu perihal apakah kata yang dipilih untuk ringkasan sesuai dengan kalimat asli. Hasil dari penelitian ini adalah metode *Restricted Boltzmann Machine* dan *Deep Learning* bekerja secara efektif dan efisien dalam pembuatan *Extractive summary* terhadap artikel yang bersifat laporan *factual*.

3. *DATASET*

3.1 *Dataset Arxiv Scientific Paper*

Dataset yang digunakan merupakan *dataset Arxiv Scientific Paper* yang diambil dari Cohan et al [2] dan Cachola et al [1] yang telah diolah. Pengolahan pertama adalah *dataset* akan dilakukan *filtering* dan diambil *data* hanya diperlukan yaitu *data introduction text* yang berada pada *list* "article_text" yang akan diberikan *label* "source" dan "abstract_text" yang akan diberikan label "target". *Dataset training* terdiri dari 203,037.00 *pairs source* dan *target* dengan *dataset training_src* memiliki spesifikasi *average*: 5,056.80 *words* dan *max*: 146,832.00 *words*, sedangkan *training_trg* memiliki spesifikasi *average*: 1,615.52 *words* dan *max*: 153,100.00 *words*. *Dataset* validasi terdiri dari 6,436.00 *pairs source* dan *target* dengan *dataset val_src* memiliki spesifikasi *average*: 4,894.36 *words* dan *max*: 46,888.00 *words*, sedangkan *val_trg* memiliki spesifikasi *average*: 957.11 *words* dan *max*: 1,906.00 *words*.

3.2 *Dataset* SciTLDR-A

Pada penelitian ini juga digunakan *dataset* SciTLDR-A yang diambil dari Peters et al[7] akibat *dataset Arxiv Scientific Paper* memiliki ukuran panjang *input* dan *output* yang melebihi kapasitas dari *hardware(Error Out of Memory)*. *Dataset* SciTLDR-A *training* terdiri dari 1,992.00 *pairs source* dan *target* dengan *dataset train_src* memiliki spesifikasi *average*: 1,089.15 *words* dan *max*: 2,957.00 *words*, sedangkan *train_trg* memiliki spesifikasi *average*: 136.60 *words* dan *max*: 668.00 *words*. *Dataset* validasi terdiri dari 619 *pairs source* dan *target* dengan *dataset val_src* memiliki spesifikasi *average*: 1,091.81 *words* dan *max*: 2,379.00 *words*, sedangkan *val_trg* memiliki spesifikasi *average*: 134.02 *words* dan *max*: 249 *words*.

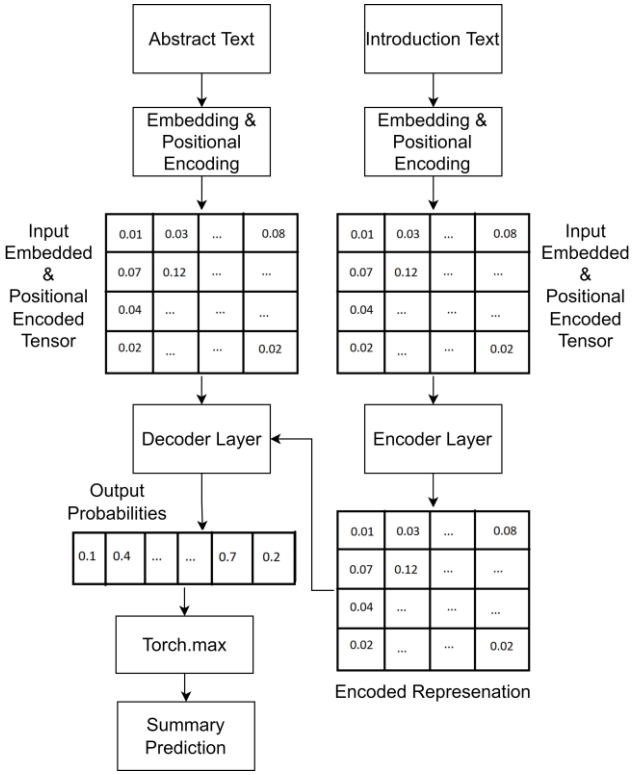
3.3 *Dataset Arxiv* cs.LG

Pada penelitian ini juga digunakan *dataset* buatan sendiri dengan melakukan *scrapping* dari Arxiv pada bidang *machine learning*(cs.LG). *Dataset* Arxiv cs.LG kemudian akan dilakukan *truncate* terhadap setiap *src* dan *trg* pada tanda baca "." dan diambil maksimal 512 *word*. Setelah proses *truncate*, *dataset* akan difilter menggunakan R-1 *precison score* dan diambil bila nilai lebih dari 0.65. *Dataset train* terdapat 4,187.00 pasang *src* dan *trg*, *train_src average:* 3,278.00 *words* dan *max*: 55,382.00 *words*, *train_trg average:* 1,125.45 *words* dan *max:* 1,912.00 *words*.

Dataset validation terdapat 220 pasang *src* dan *trg*, *val_src* average: 3,254.34 *words* dan *max*: 3,715.00 *words*, *val_trg average*: 1,142.25 *words* dan *max:* 1,880.00 *words*.

4. METODE

Metode yang digunakan adalah dengan pengaplikasian *Adaptive Sparse Transformer* [3] yang merupakan sebuah *deep learning model* yang memiliki *encoder* dan *decoder* dengan *attention layer*. pada *abstractive text summarization*. Dalam penelitian ini juga akan dilakukan proses *scraping dataset* yang akan digunakan dalam proses evaluasi.



Gambar 1. Alur Sistem *Text Summarization*

Pada Gambar 1 ditunjukkan alur sistem *text summarization* dimulai dengan pengambilan bagian dari *introduction text* sebagai *input* dan *abstract text* sebagai *target* dan *dievaluasi* menggunakan *metric* ROUGE-1 *precision score*. *Introduction text* akan diberikan *input embedding* melalui *nn.Embedding layer dan positional encoding* dengan dua fungsi sinus dan cosinus. Pada kalimat yang berposisi genap akan menggunakan fungsi sinus (1), sedangkan pada posisi ganjil menggunakan rumus cosinus (2) yang kemudian nilai *positional encoding* tersebut akan ditambahkan ke dalam *embedding* inputan.

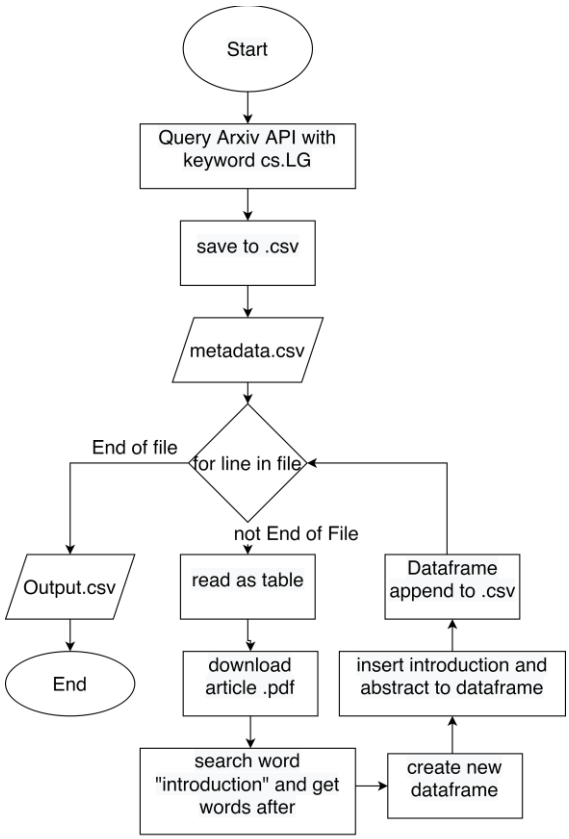
$$PE_{(pos,2i)} = \sin(pos/1000^{2i/d_{model}}) \quad (1)$$

$$PE_{(pos,2i+1)} = \cos(pos/1000^{2i/d_{model}}) \quad (2)$$

Setelah itu *input embedded* dan *positional encoded tensor memasuki encoder layer* menghasilkan *encoded representation. Abstract text* dari *dataset training* akan melalui proses *input embedding* dan *positional encoding* kemudian bersamaan dengan *encoded representation* akan berlanjut ke *layer decoders* yang kemudian akan menghasilkan *output probabilities* yang kemudian diambil *highest next word* yang kemudian akan dibandingan *train target* dengan *loss function* yang kemudian menghasilkan *training loss* yang akan dicatat.

4.1 *Scraping Dataset Arxiv Machine Learning*

Pengambilan *Dataset Arxiv Machine Learning* adalah hasil *scrapping* dari *Arxiv* API. Pertama dengan mengakses *Arxiv* API dengan kategori *Machine Learning* atau kode cs.LG yang kemudian akan didapat *metadata* yang berisikan *metadata* seperti *title*, id, *author*, *abstract* dan *link* pdf dari artikel dan disimpan pada tabel .csv yang kemudian akan dilakukan scrapping. Proses ini akan dijelaskan lebih lanjut dengan Gambar 2 di bawah.



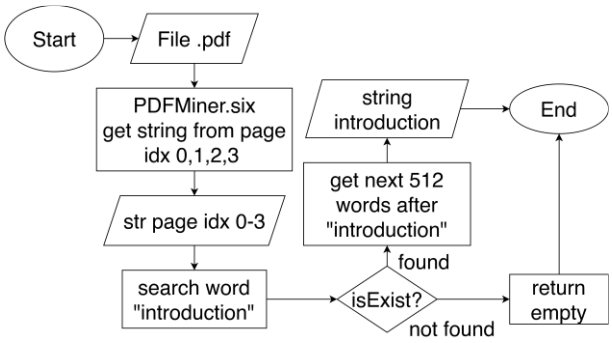
Gambar 2. *Flowchart Scrapping Dataset Arxiv Machine Learning*

Pada Gambar 2 ditunjukkan cara kerja dari proses *scrapping data Arxiv Machine Learning.* Dimulai dengan melakukan *query* pada *Arxiv* API yang akan mendapatkan *metadata* yang akan disimpan pada *file* .csv. *Metadata* tersebut kemudian akan diread per *line* di mana tiap *entry* akan dilakukan *loop* untuk mengunduh *file* .pdf dari artikel, *file* tersebut akan disimpan pada *temporary* yang kemudian akan diambil kalimat *introduction*. *Introduction text* dan *abstract text* tersebut akan disimpan pada sebuah *dataframe* baru dengan kolom *source* dan *target* yang kemudian akan di*append*kan pada sebuah .csv. Proses *loop* akan berlangsung hingga *end of file* dan *program* akan berhenti.

4.2 Pengambilan *Introduction Text* dari *File PDF*

Pada Gambar 23 Pengambilan *Introduction Text* dari *File* PDF dilakukan dengan bantuan *library* PDFMiner.six. PDFMiner.six merupakan *library* pada bahasa pemrograman *python* di mana

library ini digunakan untuk melakukan *parse string text* pada sebuah *file* berektensi .pdf. Proses pengambilan *introduction text* ini akan dijelaskan lebih lanjut pada Gambar 3 di bawah ini.

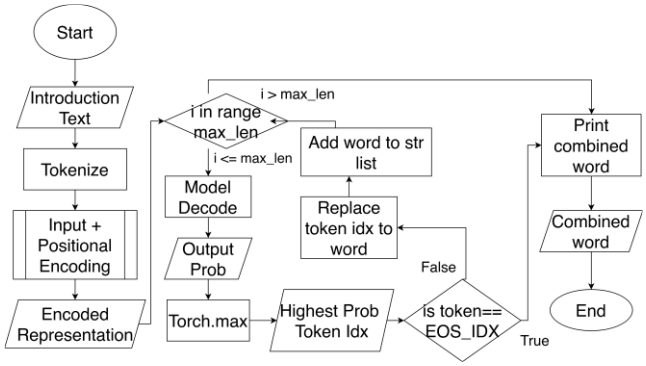


Gambar 3. *Flowchart* Pengambilan *Introduction Text* dari *File* PDF

Pada Gambar 3 ditunjukkan proses *flowchart* pengambilan *introduction text* dari *file* pdf. Pertama *file .pdf* akan di*read* oleh PDFMiner.six untuk mengambil *string text* dari halaman *index* 0 hingga 3 yang akan disimpan sebagai *string*. *String* tersebut kemudian dilakukkan *string find* kata "*introduction*" dan bila *exist* akan diambil 512 kata selanjutnya setelah kata *introduction* dan disimpan sebagai *string*, bila tidak ditemukan maka akan di*return*kan *empty*.

4.3 *Generating Summary* dari *Output Probabilities*

Pembuatan *summary* dari *output probabilities* dilakukan dengan dari *numerical token* setelah melewati *decoder layer* menjadi kalimat. Proses lebih lanjut akan dijelaskan pada Gambar 4 di bawah ini.



Gambar 4. *Flowchart Generating Summary* dari *Output Probabilities*

Pada Gambar 4 ditunjukkan *flowchart* pembuatan *summary*, pertama *introduction text* akan dilakukan *tokenization, input embedding* dan *positional encoding* menjadi *encoded representation*. *Encoded representation* tersebut akan dilakukan *looping* pada *range max_len* untuk melakukan *decode* menghasilkan *output probabilities* yang memiliki ukuran tensor 1 dimensi dengan panjang 32000(panjang *vocabulary size*) yang berisikan *probability*. *Output probabilities* tensor tersebut kemudian dilakukan *torch.max* untuk menghasilkan *index* dengan *probability* tertinggi. *Index* tersebut akan direplace dengan kata sesuai dengan *dictionary vocabulary* dan dimasukkan kepada *string list*. *Looping* akan berlangsung hingga *i* mencapai *max_len* atau *token index* memiliki *index* yang sama dengan *end of sentence token*. *Program* kemudian akan melakukan *print* dari *combined word* dan *program* berhenti.

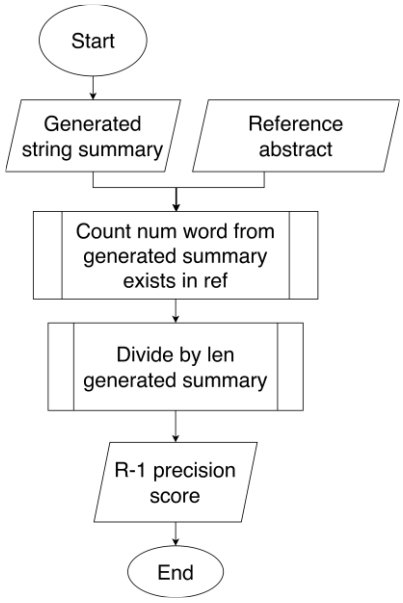
```
###LOOP -3 ###
Probability Tensor size:
torch.Size([1, 32000])
Probability Tensor:
tensor([[82.7636, 85.1664, 82.7899,  ..., 85.5976, 87.7925, 87.4056]],
       device='cuda:0')
Torch max Prob Tensor, output idx of highest probability word idx token:
761
Replace idx token to word:
dynamics

Combine the string: we study the dynamics
```

Gambar 5. *Flowchart Generating Summary* dari *Output Probabilities*

Pada Gambar 5 ditunjukkan *output probability* pada proses *summary*. Tensor berukuran 32000 yang merupakan panjang *vocabulary* dan berisikan *probability* dari tiap *index*. *Probability* tensor tersebut kemudian akan dicari nilai maksimumnya dan mendapatkan *index* nilai tertinggi. Nilai *index* tertinggi itu akan *dicovert* menjadi kata sesuai dengan *dictionary vocabulary*.

4.4 Menghitung ROUGE-1 *Precision Score*

Perhitungan ROUGE-1 *precision score* diperlukan untuk melakukan evaluasi dari hasil *summary*. Proses evaluasi lebih lanjut akan dijelaskan lebih dalam pada Gambar 6 di bawah ini.



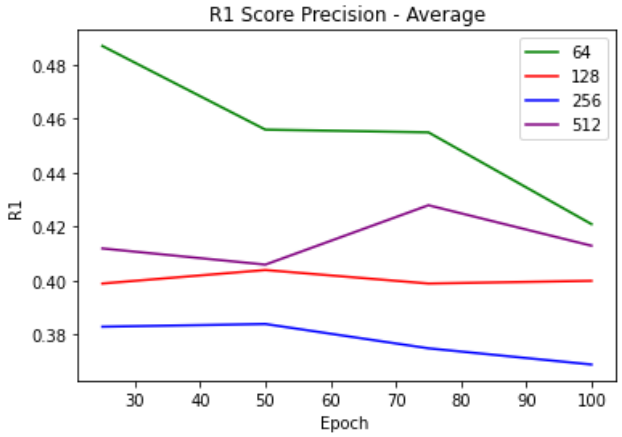
Gambar 6. *Flowchart* Evaluasi ROUGE-1 *Precision Score*

Pada Gambar 6 dijelaskan proses dari *flowchart* evaluasi ROUGE-1 *precision score*. Pertama *generated string summary* dan *reference abstract* akan dihitung jumlah kata yang berada pada *generated string summary* dan *reference abstract* setelah itu akan dibagi dengan jumlah panjang dari *generated string summary* yang akan menghasilkan nilai R-1 *precision score*.

5. PENGUJIAN

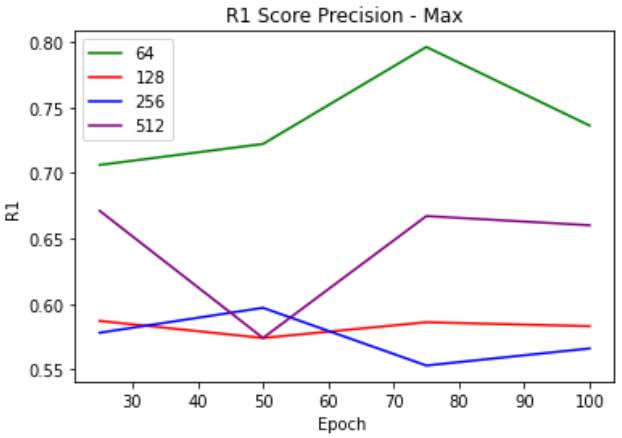
5.1 Hasil Proses *Training* dengan *Dataset Arxiv Machine Learning* dengan Berbagai *Token Length*

Pada proses *training* dengan *Dataset Arxiv Machine Learning* dengan berbagai *token length* dilakukan dengan cara melakukan *training* dengan panjang *token* 64, 128, 256 dan 512 untuk melihat perbedaan *performance* dari panjang *token* pada 1000 *pair training set* dan 100 *pair validation set* dan *training* dilakukan hingga *epoch* ke 100.



Gambar 7. ***Average ROUGE-1 Precision Score dari Berbagai Token Length***

Pada Gambar 7 ditunjukkan nilai *average* dari ROUGE-1 *precision score* dimana dapat dilihat nilai tertinggi adalah *token* dengan panjang 64 kemudian disusul dengan *token* dengan panjang 512, 128 dan yang paling rendah adalah 256. *Token* dengan panjang 64 terlalu pendek bila digunakan untuk *data* berupa *introduction text* dan *abstract text* sehingga pilihan terbaik selanjutnya adalah *token* dengan panjang 512.

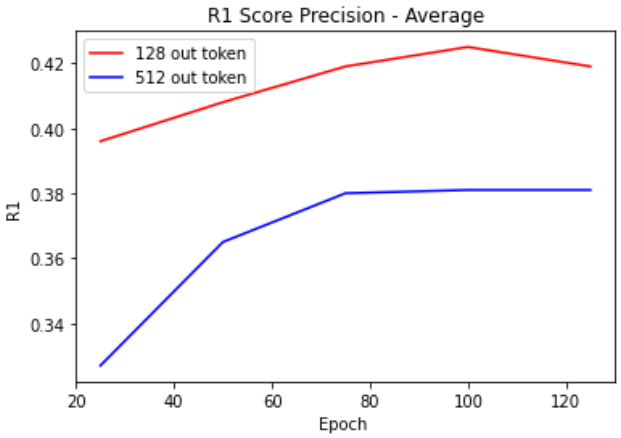


Gambar 8. ***Maximum ROUGE-1 Precision Score dari Berbagai Token Length***

Pada Gambar 8 ditunjukkan nilai *maximum* dari ROUGE-1 *precision score* dimana dapat dilihat nilai tertinggi adalah *token* dengan panjang 64 kemudian disusul dengan *token* dengan panjang 512, 128 dan yang paling rendah adalah 256. *Token* dengan panjang 64 terlalu pendek bila digunakan untuk *data* berupa *introduction text* dan *abstract text* sehingga pilihan terbaik selanjutnya adalah *token* dengan panjang 512.

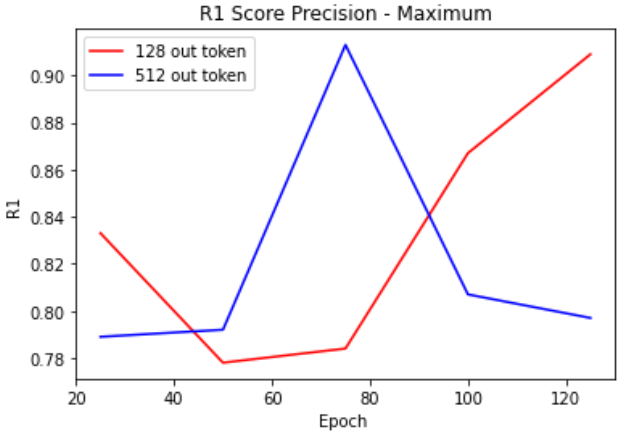
5.2 Hasil Percobaan Proses *Training* dengan Dataset *Arxiv Machine Learning* pada *Output Length*

Pada proses *training* dengan *Dataset Arxiv Machine Learning* dengan berbagai *token length* dilakukan dengan cara melakukan *training* dengan panjang *output length* 128 dan 512 untuk melihat perbedaan *performance* dari panjang *output length* pada 108,882 *pair training set* dan 4058 *pair validation set* dan *training* dilakukan hingga *epoch* ke 125.



Gambar 9. ***Average* ROUGE-1 *Precision Score* pada *Output Length***

Pada Gambar 9 ditunjukkan nilai *average* dari ROUGE-1 *precision score* dari percobaan sebanyak 125 *epoch* dengan 128 *output length* dan 512 *output length*. Dapat dilihat bahwa nilai tertinggi ROUGE-1 *precision average* adalah *output* dengan panjang 128.



Gambar 10. ***Maximum* ROUGE-1 *Precision Score* pada *Output Length***

Pada Gambar 10 ditunjukkan nilai *maximum* dari ROUGE-1 *precision score* dari percobaan sebanyak 125 *epoch* dengan 128 *output length* dan 512 *output length*. Dapat dilihat bahwa *output length* 128 mengalami penurunan pada awal *training* tetapi mengalami peningkatakan setelah *epoch* 100, sedangkan *output length* mengalami peningkatan paling tinggi pada *epoch 75*

kemudian mengalami penurunan. Pada akhir *epoch* 125 dapat dilihat bahwa nilai tertinggi ROUGE-1 *precision maximum* adalah *output* dengan panjang 128.

5.3 Hasil dan Evaluasi *Training Model* dengan Berbagai *Parameter* dengan *Dataset* SciTLDR-A

Pada subbab ini dilakukan *training* dengan *dataset* SciTLDR-A yang diambil dari https://github.com/allenai/scitldr akibat *dataset Arxiv Scientific Paper* memiliki ukuran panjang *input* dan *output* yang melebihi kapasitas dari *hardware*(*Error Out of Memory*).

Tabel 1. *Overview Average* dan *Max* R-1 *Precision Score*

Uji Coba	
Average	
***Average* R-1**	0.154
***Max* R-1**	0.487

Pada Tabel 1 ditunjukkan hasil observasi dari 12 macam percobaan dengan berbagai *parameter* yang berbeda pada *dataset* SciTLDR-A. Dari *metric* R-1 *precision score* menunjukkan bahwa performa *model* sangatlah rendah dan tidak sesuai dengan harapan yang diinginkan. Pada tahapan pengujian selanjutkan akan diuji coba apakah dengan menambahkan jumlah *dataset training* dapat meningkatkan performa dari *model*.

5.4 Hasil *Training* dengan *Dataset Arxiv Scientific Paper* dengan *Blind Truncate*

Pada pengujian ini dilakukan *training* dengan *dataset arxiv scientific paper* dimana *dataset* dilakukan *truncate* sebanyak 512 *words* pertama. Proses *truncate* dilakukan secara *blind-truncate* di mana *dataset* semerta-merta dipotong sebanyak 512 *words* tanpa mempedulikan isi dari *data*. Pada tahap ini penulis memiliki hipotesa bahwa model akan mempelajari *data* dan mampu membedakan mana *data noise* melalui *encoder layer*. *Parameter* pada *model* ini adalah 8 *multi-heads*, 3 *encoder layers*, 3 *decoder layers*, *feed-forward dimension* 512, *embed size* 512, *dropout* 0.1 dan dengan *Vocabulary size* 72248. *Model* ini menggunakan *optimizer* adamw dengan *learning-rate:* 0.0001. Proses *training* dilakukan dengan *dataset training* sebanyak 202,895.00 *src* dan *trg pair* dan dengan *evaluation dataset* sebanyak 6436 *src* dan *trg pair*. Proses *training* dengan GPU Tesla P100 membutuhkan waktu kurang lebih 6010 detik.

Tabel 2. Perbandingan *Training* dengan *Dataset* SciTLDR-A dan *Arxiv Scientific Paper (Truncated)*

	SciTLDR-A	**ArxSciTruncated**	**Diff.**	**Diff.%**
AVG R-1	0.154	0.425	0.271	275.97%
MAX R-1	0.487	0.938	0.451	192.61%

Pada Tabel 2 ditunjukkan hasil peningkatan performa dari perubahan jumlah *dataset* dengan cara merubah *dataset* dari *dataset* SciTLDR-A (1,992.00 *training pairs*) menjadi *dataset Arxiv Scientific Paper(Truncated)* (203,037.00 *training pairs*). Perbedaan performa *model* adalah sebanyak 0.271(275.97%) pada *average* R-1 *precision score* dan 0.451(192.61%) pada *maximal* R-1 *precision score*.

Berdasarkan jurnal "What Have We Achieved on Text Summarization?" [5] performa *model* ini apabila dilihat dari *metric* R-1 *precision score* telah berada pada rata-rata bahkan melampaui beberapa performa *model* lain yang memiliki *task abstractive summarization*.

Pada Tabel 3 ditunjukkan hasil perbandingan R-1 *score* dari berbagai *model* terhadap *abstractive summarization task*. Dapat dilihat bahwa *model* yang di*training* menggunakan *Arxiv Scientific Paper* dengan *Blind Truncate* mampu melampaui beberapa *model* dengan nilai *average* 42.5 namun *model* masih tidak bisa menandingi *model BART*.

Tabel 3. Tabel Perbandingan Hasil Performa R-1 *Score* dari *Model*[5]

	Model (Abstractive Summarization Task)						
	S2S	**PG**	**PG-Coverage**	**Bottom Up**	**BertAbs**	**Our**	**BART**
R-1 Score	31.33	36.44	39.53	41.22	42.13	42.5	44.16
Dataset	CNN	CNN	CNN	CNN	CNN	Arxiv Sci	CNN

5.5 Evaluasi *Model* dengan *Byte Pair Encoding Vocabulary* dan Dataset Arxiv+cs.LG *Cleaning* Menggunakan R-1 *score*

Pada pengujian ini dilakukan percobaan *training model* dengan mengimplementasikan *byte pair encoding* untuk pada bagian *tokenizer* beserta proses *cleaning* menggunakan R-1 *score*. *Dataset* Arxiv Scientific Paper dan cs.LG yang telah dibuat kemudian digabungkan(*merge*) menjadi 108,882.00 pasang *train dataset* dan 4,058.00 pasang *validation dataset*. *Model* memiliki parameter 8 *multi-heads, 3 encoder layers, 3 decoder layers,* 512 *feed-forward dimension* dan 512 *embed size, sedangkan tokenizer* memiliki *vocab size* 32K. Proses *cleaning menggunakan* R-1 *score* diambil dengan cara pertama kita memotong *token* sebanyak 512 kemudian kita mencari simbol titik (.), kemudian kalimat setelah titik terakhir akan dihapus, sehingga semua kalimat berakhiran dengan titik. Kemudian semua *target* akan dibandingkan dengan *source* dengan R-1 *precision score*. Bila R-1 score melebihi 0.65 maka *data* akan digunakan.

Tabel 4. Tabel Perbandingan Hasil Performa R-1 *Score* dari *Model*[5]

	Model							
	S2S	**PG**	**ArxivSci +cs.LG**	**PG-Coverage**	**Bottom Up**	**Bert Abs**	**Our (ArxivSci)**	**BART**
R-1 Score	31.33	36.44	39.4 (AVG)	39.53	41.22	42.13	42.5(AVG)	44.16
Dataset	CNN /DM	CNN /DM	Arxiv Sci+csLG	CNN /DM	CNN /DM	CNN /DM	Arxiv Scientific	CNN /DM

Pada Tabel 4 ditunjukkan hasil pengujian ini adalah performa *model* ini apabila dilihat dari *metric* R-1 *precision score* mengalami penurunan terhadap *model* terdahulu yang hanya dilakukan *training* dengan *dataset Arxiv Scientific Paper* dengan tambahan *dataset* cs.LG(*Machine Learning*), namun telah berada pada rata-

rata bahkan melampaui beberapa performa *model* lain yang memiliki *task abstractive summarization*.

5.6 Evaluasi *Model* dengan *Byte Pair Encoding Vocabulary* dan *Dataset* Arxiv+cs.LG *Cleaning* terhadap Beberapa *Data* dari Ahli Menggunakan R-1 *score*

Pada pengujian ini dilakukan evaluasi dari beberapa data dari ahli. Para ahli yang dimaksud adalah dosen informatika yang telah memberikan data *introduction text* beserta ringkasannya. Berikut merupakan list DOI dari jurnal yang digunakan untuk evaluasi ahli:

1. doi.org/10.1007/s11042-020-10299-5
2. doi.org/10.1007/s11831-020-09464-8
3. doi.org/10.1016/J.COMPBIOMED.2021.104499
4. doi.org/10.1016/j.elerap.2021.101048

Daftar para ahli dan nilai evaluasi akan ditampilkan pada Tabel 5.

Tabel 5. Tabel Evaluasi *Data* dari Para Ahli

File/Judul	R-1
Entry 1	0.4223
Entry 2	0.5
Entry 3	0.3602
Entry 4	0.3875
Average	0.4175

Pada Tabel 5 ditunjukkan hasil dari evaluasi dari *model Arxiv+csLG* dengan *data* yang diberikan oleh para ahli. Observasi dengan *metric* nilai R-1 *precision score* bila dibandingkan pada Tabel 4 memiliki penurunan dari 39.4 menjadi 37.09.

6. KESIMPULAN

Pada penelitina ini ditemukan beberapa kesimpulan:

1. *Parameter model* memiliki pengaruh terhadap performa R-1 *score* tetapi tidak signifikan, performa tertinggi didapatkan dengan 8 *multiheads*, 3 *encoder layers*, 3 *decoder layers*, 512 *feed-forward dimension*, 512 *embed size* dengan *average* R-1 *score* maksimal 42.5(*model Arxiv Scientific Paper*) dan 39.4(*model Arxiv Scientific Paper+cs.LG*).
2. Jumlah *dataset* memiliki pengaruh yang signifikan pada performa R-1 *score*, pada 11 percobaan dengan *dataset SciTLDR-A* A (1,992.00 *training pairs*) memiliki nilai rata-rata R-1 *score* 0.154 dan nilai maksimal 0.487 sedangkan pada percobaan dengan *dataset Arxiv Scientific Paper*(*truncated* 512 *first token*) memiliki nilai rata-rata R-1 *score* 0.425(275.97% *increase*) dan nilai maksimal 0.938(192.61% *increase*). Pada percobaan dengan *dataset Arxiv Scientific Paper(truncated 512 first token)* + *Arxiv* bidang cs.LG(*Machine Learning*) mengalami penurunan nilai rata-rata R-1 *score* menjadi 0.394 dan nilai maksimal 0.833.
3. *Model ArxivScientificPaper* (42.5) mampu mengalahkan model S2S (31.33), PG (36.44), ArxivScientificPaper+csLG (39.4), PG-Coverage (39.53), BottomUp (41.22), BertAbs (42.13), namum tidak bisa mengalahkan model BART(44.16) pada *metric* R-1 *score* pada masalah *abstractive text summarization*.

7. SARAN

Berikut adalah saran yang dapat dilakukan untuk mengembangkan penelitian ini lebih lanjut:

1. Menggunakan *dataset full* tanpa pemotongan/*truncate* sehingga *data* yang digunakan sangatlah sesuai dengan *data asli* pada *real-world usage*.
2. Penggunaan *pre-trained model* seperti BERT, GPT-3 dan melakukan *transfer-learning* untuk meningkatkan performa dari *model*.

8. REFERENSI

[1] Cachola, I., Lo, K., Cohan, A. & Weld, D.S. 2020. TLDR: Extreme Summarization of Scientific Documents. *Allen Institute for AI*. DOI=http://dx.doi.org/10.18653/v1/2020.findings-emnlp.428.

[2] Cohan, A., Dernoncourt, F., Kim, D. S., Bui, T., Kim, S., Chang, W. & Goharian, N. 2018. A discourse-aware attention model for abstractive summarization of long documents. *Association for Computational Linguistics (ACL), 615-621* . DOI=https://doi.org/10.18653/v1/n18-2097.

[3] Correia, G. M., Niculae, V. & Martins, A. F. T. 2020. Adaptively sparse transformers. *Association for Computational Linguistic (ACL)*, 2174–2184. DOI=https://doi.org/10.18653/v1/d19-1223

[4] El-Kassas, W. S., Salama, C. R., Rafea, A. A., & Mohamed, H. K. 2020. Automatic Text Summarization: A Comprehensive Survey. *Expert Systems with Applications*, 165, 113679. DOI=https://doi.org/10.1016/j.eswa.2020.113679.

[5] Huang, D., Cui, L., Yang, S. Bao, G., Wang, K. Xie, J. & Zhang, Y. 2020. What Have We Achieved on Text Summarization?. *School of Engineering, Westlake University*. DOI=http://dx.doi.org/10.18653/v1/2020.emnlp-main.33.

[6] Ju, J., Liu, M., Gao, L. & Pan, S. 2020. SciSummPip: An Unsupervised Scientific Paper Summarization Pipeline. *Association for Computational Linguistics (ACL), 318–327*. DOI=https://doi.org/10.18653/v1/2020.sdp-1.37

[7] Peters, B., Niculate, V. & Martins, A. F. T. 2019. Sparse Sequence-to-Sequence Models. *In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 1504–1519*. DOI=http://dx.doi.org/10.18653/v1/P19-1146.

[8] Pilault, J., Li, R., Subramanian, S. & Pal, C. 2020. On Extractive and Abstractive Neural Document Summarization with Transformer Language Models. *Association for Computational Linguistics (ACL), 9308–9319*. DOI=https://doi.org/10.18653/v1%2F2020.emnlp-main.748.

[9] Sun, X. & Zhuge, H. 2018. Summarization of Scientific Paper Through Reinforcement Ranking on Semantic Link Network. *IEEE Access, 2018-Vol6*, 40611-40625. DOI=https://doi.org/10.1109/ACCESS.2018.2856530.

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones,L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 2017-December, 5999-6009. DOI=https://doi.org/10.48550/arXiv.1706.03762.

[11] Verma, S. & Nidhi, V. 2017. Extractive Summarization using Deep Learning. *Delhi Technological University*. DOI=http://doi.org/10.13053/rcs-147-10-9.