BESKlus : BERT *Extractive Summarization with* K-Means *Clustering in Scientific Paper*

http://dx.doi.org/10.28932/jutisi.v8i1.4474

Riwayat Artikel
Received: 3 Februari 2022 | *Final Revision*: 26 Februari 2022 | *Accepted*: 26 Februari 2022

Feliks Victor Parningotan Samosir [#1], Hapnes Toba [✉*2], Mewati Ayub [#3]

[#] *Magister Ilmu Komputer, Universitas Kristen Maranatha*
Jl. Prof. drg. Suray Sumantri No. 65, Bandung, Indonesia
[1]2079002@maranatha.ac.id
[3]mewati.ayub@maranatha.ac.id

[*] *Magister Ilmu Komputer, Universitas Kristen Maranatha*
Jl. Prof. drg. Surya Sumantri No. 65, Bandung, Indonesia
[2]hapnes.toba@maranatha.ac.id

Abstract — **This study aims to propose methods and models for extractive text summarization with contextual embedding. To build this model, a combination of traditional machine learning algorithms such as K-Means Clustering and the latest BERT-based architectures such as Sentence-BERT (SBERT) is carried out. The contextual embedding process will be carried out at the sentence level by SBERT. Embedded sentences will be clustered and the distance calculated from the centroid. The top sentences from each cluster will be used as summary candidates. The dataset used in this study is a collection of scientific journals from NeurIPS. Performance evaluation carried out with ROUGE-L gave a result of 15.52% and a BERTScore of 85.55%. This result surpasses several previous models such as PyTextRank and BERT Extractive Summarizer. The results of these measurements prove that the use of contextual embedding is very good if applied to extractive text summarization which is generally done at the sentence level.**

Keywords— **contextual embedding; extractive summarization; k-means clustering; scientific paper; sentence-bert.**

I. PENDAHULUAN

Saat ini ada banyak *online journal database sites* yang menyediakan makalah jurnal *online* yang bersifat *free access* dan juga *open access*. arXiv sendiri memiliki sekitar 1,8 juta artikel ilmiah. Beberapa situs konferensi juga menyediakan makalah jurnal ilmiah *periodical* mereka untuk publik, misalnya NeurIPS [1]. Melimpahnya makalah makalah jurnal ilmiah yang mudah diakses memberikan manfaat yang besar terutama pada akademisi (pengajar maupun peserta didik). Namun di sisi yang lain, menimbulkan tantangan. Tantangan tersebut adalah menemukan makalah makalah jurnal yang tepat untuk dapat membantu memberikan *insight* untuk tugas, proyek, ataupun penelitian yang sedang dikerjakan pembaca. Untuk itu, pembaca perlu mengekstrak intisari dari informasi yang disampaikan peneliti dalam makalah jurnalnya.

McCarthy dan Lehnert [2] menjelaskan bahwa tujuan utama dari ekstraksi informasi adalah mengidentifikasi sebuah informasi dalam sekumpulan teks. Mengekstrak informasi merupakan sebuah strategi yang baik namun membutuhkan banyak waktu dan tenaga apalagi jika pembaca memerlukan informasi dari banyak makalah jurnal. Diperlukan sebuah penggalian teks yang dapat dilakukan secara otomatis. Proses ekstraksi teks atau informasi melalui penggalian teks ini merupakan salah satu tugas yang terus dikembangkan dalam bidang *Natural Language Processing*/Pemrosesan Bahasa Alami (PBA). Beberapa teknik penggalian teks dalam pemrosesan bahasa alami adalah *automatic keyword extraction* dan *automatic text summarization* [3].

Peringkasan Teks Otomatis (PTO) kebanyakan dilakukan pada artikel berita. Yasunaga dkk [4] menyebutkan bahwa PTO pada makalah jurnal ilmiah kurang tereksplorasi. Hal ini disebabkan karena secara umum, makalah jurnal ilmiah berukuran panjang dan mengandung konsep yang lebih kompleks dan juga istilah-istilah teknikal. Selain itu, makalah jurnal ilmiah terdiri dari sub-bab dan juga mengandung sitasi.

Dalam konteks makalah jurnal ilmiah, bagian abstrak dianggap sebagai ringkasan dari seluruh isi makalah jurnal ilmiah. Namun Altmami dan Menai [5] menyebutkan ada beberapa alasan mengapa pembuatan model untuk PTO diperlukan walaupun dalam makalah jurnal tersebut sudah terdapat abstrak. Pertama, informasi pada abstrak biasanya tidak mengandung

konten relevan dari keseluruhan teks. Kedua, abstrak menjelaskan pandangan penulis tentang karakteristik unik dengan cara yang *bias* dan tidak lengkap. Ketiga, tidak ada ringkasan tunggal yang cocok untuk semua pembaca. Selain itu, abstrak tidak merefleksikan semua dampak dan kontribusi dari makalah jurnal ilmiah tersebut, melainkan hanya menampilkan hal-hal yang ingin disoroti penulis makalah jurnal ilmiah.

Erera dkk dari IBM Research [6] mewawancara beberapa mahasiswa program doktoral dalam bidang PBA tentang bagaimana mereka mengakses makalah jurnal ilmiah, seberapa sering, bagaimana mengeksplorasi kontennya, dan kendala apa yang mereka temui dengan perangkat pencarian yang ada sekarang. Mayoritas menjawab *information overload* sebagai kendala utama sehingga butuh daya lebih untuk membaca makalah jurnalnya. Mayoritas juga menemukan bahwa abstrak tidak cukup informatif dalam menentukan relevansinya dengan topik yang ingin dipelajari. Karenanya, pentingnya PTO pada makalah jurnal ilmiah adalah membantu peneliti memahami inti dari makalah jurnal tersebut tanpa harus membaca seluruhnya atau bahkan membuka file PDF-nya.

Untuk itulah, model PTO diharapkan dapat mencakup semua bagian utama dan menampilkan informasi paling penting dari makalah jurnal ilmiah. Karena makalah jurnal ilmiah merupakn domain teknikal dengan bahasa yang cukup teratur dan eksplisit [7], maka penelitian ini akan berfokus kepada peringkasan ekstraktif.

Salah satu arsitektur *machine learning* tradisional yang cukup sering digunakan untuk peringkasan ekstraktif secara *unsupervised* adalah K-Means Clustering. Prathima dan Divakar [8] menghasilkan optimal *clustering* dari sebuah dokumen menggunakan algoritma *probabilistic k-representative clustering* dan membuat ringkasan lebih efisien dengan menggunakan *phrase rank*. Manh dkk [9] menggunakan K-Means *Clustering* yang dikombinasikan dengan metode atau pengukuran lainnya seperti metode berbasis *centroid*, pengukuran *Maximal Marginal Relevance (MMR)*, dan *Sentence Position* pada teks aslinya.

Arsitektur *machine learning* lainnya yang saat ini populer digunakan untuk tugas-tugas Pemrosesan Bahasa Alami adalah BERT [10]. BERT dikembangkan pada tahun 2018 oleh beberapa peneliti Google dan merancangnya untuk melatih representasi dua arah secara mendalam dari teks yang tidak berlabel dengan mengkondisikan secara bersama-sama pada konteks kiri dan kanan di semua lapisan [11]. Sehingga membuatnya dapat membaca seluruh urutan kata sekaligus. Hasilnya, model *pretrained* BERT dapat disesuaikan *(fine-tuned)* hanya dengan satu tambahan lapisan keluaran untuk membentuk model yang dapat digunakan pada semua jenis tugas dalam PBA termasuk tugas peringkasan teks otomatis.

Ada berbagai penelitian yang dikembangkan dengan melakukan *fine-tuning* pada BERT, salah satunya SBERT. SBERT atau Sentence-Transformers merupakan salah satu pengembangan dari BERT untuk level kalimat, paragraf, dan gambar [12]. Kemampuannya dalam melakukan *contextual embeddings* pada teks membuatnya mencapai hasil yang sangat baik pada berbagai *tasks* seperti *clustering, cross-encoder, parallel sentence mining, paraphrase mining, semantic search, retrieve & rerank, image search,* dan *text summarization*. Selain SBERT, untuk tugas peringkasan teks otomatis ada peringkasan teks secara ekstraktif (BERT Extractive Summarizer) untuk transkrip kursus *online* [13], dan peringkasan teks secara ekstraktif, abstraktif, dan kombinasi keduanya yang disebut dengan BERTSUM [14].

Penelitian-penelitian yang disebutkan diatas akan digunakan sebagai landasan dalam penelitian ini. Penelitian ini akan berfokus pada peringkasan tipe ekstraktif yang mengolah langsung pada level kalimat untuk melakukan *contextual embeddings*. Dataset yang digunakan adalah makalah jurnal ilmiah berbahasa Inggris dari NeurIPS [1]. Pemilihan makalah jurnal berbahasa Inggris ini karena pada umumnya *tools* yang sudah matang seperti BERT, GPT, T5, dll mayoritas menggunakan korpus berbahasa Inggris pada proses *pre-raining*-nya.

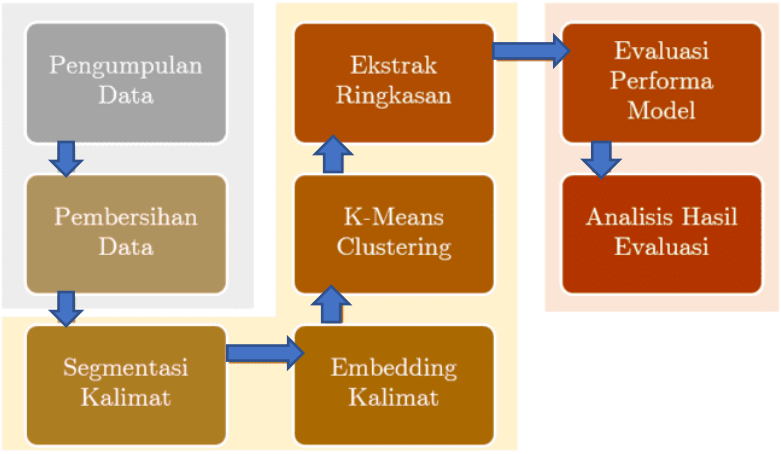
Penelitian akan memaparkan proses apa saja yang dilakukan dalam pengembangan model peringkasan ekstraktif dengan menggunakan contextual embeddings. Selain itu, penelitian ini juga akan menunjukkan hasil peringkasan yang lebih baik dengan menggunakan contextual embeddings dan menganalisis performanya ketika dibandingkan dengan model terdahulu seperti PyTextRank dan BERT Extractive Summarizer.

II. METODE PENELITIAN

Gambar 1 menunjukkan tahapan-tahapan yang dilakukan dalam penelitian ini. Terdapat tiga tahapan utama yaitu:

- Persiapan Dataset: Pengumpulan Data dan Pembersihan Data
- Perancangan Model: Segmentasi Kalimat, Embedding Kalimat, K-Means *Clustering*, dan Ekstrak Ringkasan
- Analisis Performa Model: Evaluasi Performa Model dan Analisis Hasil Evaluasi

p-ISSN : 2443-2210
e-ISSN : 2443-2229



Gambar 1 Alur Tahapan Penelitian

A. Persiapan Dataset

Pengumpulan Data akan dilakukan dengan mengambil data makalah jurnal ilmiah NeurIPS dari Kaggle [1]. Data yang diambil berisikan makalah jurnal dari tahun 1987-2019. Dataset terdiri dari dua file yaitu, author.csv (30237,4) dan papers.csv (9680, 5). File yang digunakan adalah file papers.csv saja. File papers.csv terdiri dari kolom `source_id`, `year`, `title`, `abstract`, dan `full_text`.

Pembersihan Data akan dilakukan dengan menggunakan *library re (regular expression)* dari Python. Dataset yang diambil masih dalam bentuk mentah, masih terdapat karakter-karakter unicode yang perlu ditangani. Selain itu juga masih terdapat *missing values* pada beberapa baris data. Pembersihan pertama adalah dengan membuang baris yang tidak lengkap ini (tidak ada makalah jurnalnya ataupun tidak ada abstraknya). Dataset yang tadinya terdiri dari 9680 baris atau makalah makalah jurnal, sekarang menjadi 6101 makalah makalah jurnal. Pembersihan selanjutnya adalah menghilangkan *unicode whitespace characters* seperti `\t\n\r\f\v`. Pembersihan dilakukan pada kolom `abstract` dan `full_text`. Hasil pembersihan disimpan pada kolom baru: `clean_paper` dan `clean_abstract`. Kolom lainnya akan dibuang dan hanya menyisakan kolom `year`, `clean_paper`, dan `clean_abstract` seperti yang ditunjukkan pada Gambar 2.

	year	clean_paper	clean_abstract
0	1997	A Neural Network Based Head Tracking System D....	We have constructed an inexpensive video based...
1	2000	Algorithms for Non-negative Matrix Factorizat...	Non-negative matrix factorization (NMF) has pr...
2	2007	Compressed Regression Shuheng Zhou∗ John Laffe...	Recent research has studied the role of sparsi...
3	2007	Predictive Matrix-Variate t Models Shenghuo Zh...	It is becoming increasingly important to learn...
4	2007	Loop Series and Bethe Variational Bounds in At...	Variational methods are frequently used to app...

Gambar 2 Hasil Akhir Pembersihan Dataset

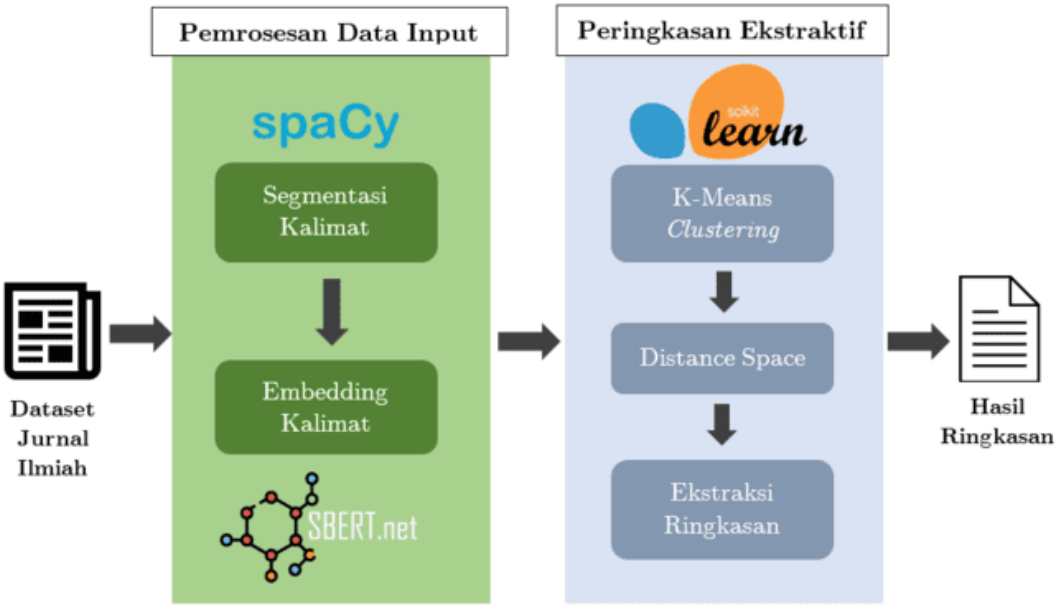
B. Perancangan Model

Gambar 3 menampilkan model peringkasan ekstraktif makalah jurnal ilmiah yang diusulkan terdiri dari dua modul utama, yaitu Pemrosesan Data Input dan Peringkasan Ekstraktif. Pada Modul Pemrosesan Data Input yang pertama dilakukan adalah segmentasi kalimat. Isi dari setiap makalah jurnal dan abstrak dipecah ke dalam bentuk kalimat-kalimat. Segmentasi ini dilakukan dengan library spacy Transformers. Dikutip dari Panduan spaCy untuk *Linguistic Features* [15], iterasi method `.sents` pada seluruh dataset akan menghasilkan Span. Span merupakan potongan-potongan dari sebuah dokumen. Ada empat cara dalam melakukan pemotongan dokumen *(Sentence Segmentation)*, yaitu *Dependency parser*, *Statistical sentence segmenter*, *Rule-based pipeline component*, dan *Custom Function*. Jika sebuah teks merupakan teks dengan tujuan umum seperti berita atau artikel web atau serupa, maka pendekatan *Dependency parser* sangat cocok dan bekerja dengan baik. Namun, untuk media sosial atau teks percakapan lebih cocok menggunakan *Custom Function* atau *Rule-based*. Penelitian ini akan menggunakan *Dependency parser*. Method `.sents` digunakan untuk memisahkan makalah jurnal dan abstrak ke dalam bentuk kalimat-kalimat. Kumpulan kalimat-kalimat ini kemudian dikonversi menjadi *list*.

Jurnal dan abstrak yang sudah tersegmentasi ini selanjutnya akan masuk fase Embedding Kalimat. Proses embedding dilakukan dengan Sentence-Transformers (SBERT). SBERT dibangun diatas BERT. Rogers dkk [10] dalam surveinya

menyebutkan bahwa hanya dalam waktu singkat, BERT sudah menjadi dasar untuk berbagai eksperimen PBA dan menginspirasi banyak penelitian serta mengusulkan berbagai pengembangan. Salah satu alasan utama kesuksesan BERT dalam PBO adalah karena dia adalah *context-based embedding model*, tidak seperti *embedding model* terkenal lainnya seperti Word2Vec yang merupakan *context-free embedding* [16]. *Distributional word representations* seperti Word2Vec [17] dan GloVe [18], mempelajari representasi vektor setiap kata dengan membangun kosakata global tunggal untuk setiap kata dengan mengabaikan konteksnya [19]. BERT menghubungkan setiap kata dalam kalimat dengan semua kata lainnya dalam kalimat untuk memahami makna kontekstual dari setiap kata [16]. Hal ini terjadi karena BERT sendiri dibangun diatas Transformer yang memiliki *self-attention mechanism* (SAM) [20]. Dengan SAM, maka dimungkinkan untuk memodelkan makna semantik sebuah kata pada konteks kata disekitarnya dalam bentuk numerik dan dengan demikian pula operasi matematika dapat dilakukan pada kata tersebut. Dari sini, banyak model yang lebih canggih muncul yang tidak hanya menangkap makna semantik statis tetapi juga makna kontekstual *(contextualized meaning)*.

Proses eksekusi embedding kalimat oleh SBERT ini menggunakan GPU. Namun, karena keterbatasan *resources*, dataset dipecah menjadi enam bagian. Setelah nanti proses ini selesai, keenam bagian tersebut akan disatukan kembali. Jurnal yang sudah tersegmentasi disimpan dalam kolom `text_clean` dan makalah jurnal yang sudah ter-embedding disimpan dalam kolom `text_embedding`. Sementara untuk hasil pengolahan abstrak disimpan dalam kolom `summary_clean`. Gambar 4 menunjukkan hasil salah satu makalah jurnal yang sudah tersegmentasi.



Gambar 3 Desain Model Yang Diusulkan Untuk Peringkasan Teks Ekstraktif Untuk Makalah Jurnal Ilmiah

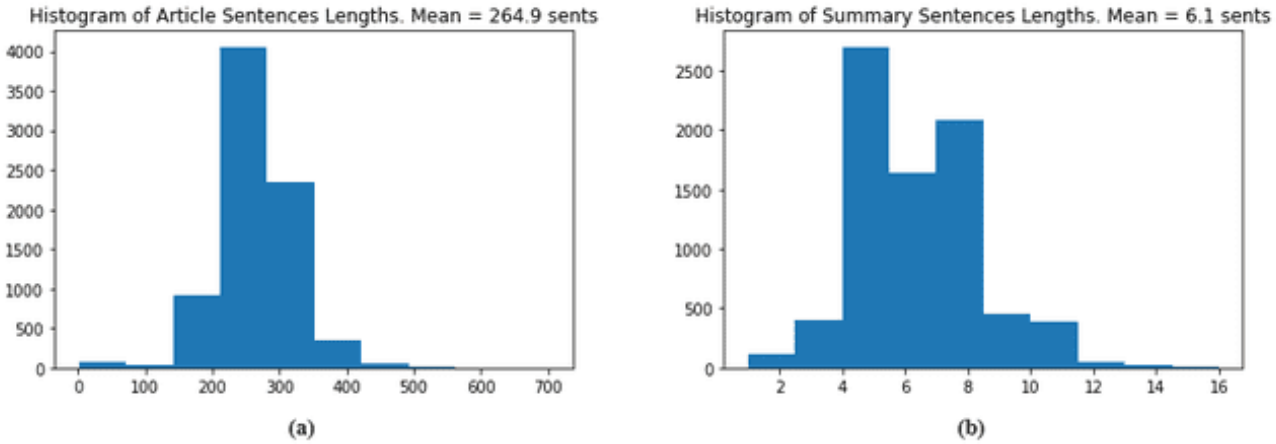
```
df['text_clean'].iloc[2]
```

```
'Compressed Regression Shuheng Zhou*',
'John Lafferty*† Larry Wasserman‡† *Computer Science Department ‡Department of Statistics',
'†Machine Learning Department Carnegie Mellon University Pittsburgh, PA',
'Recent research has studied the role of sparsity in high dimensional regression and signal reconstruction,
'In this paper we study a variant of this problem where the original n input variables are compressed by a
'A primary motivation for this compression procedure is to anonymize the data and preserve privacy by revea
'We characterize the number of random projections that are required for `1-regularized compressed regressio
'In addition, we show that `1-regularized compressed regression asymptotically predicts as well as an or- a
'Finally, we characterize the privacy properties of the compression procedure in information-theoretic term
'Two issues facing the use of statistical learning methods in applications are scale and privacy.',
'Scale is an issue in storing, manipulating and analyzing extremely large, high dimensional data.',
'Privacy is, increasingly, a concern whenever large amounts of confidential data are manipulated within an o
'It is often important to allow researchers to analyze data without compromising the privacy of customers o
'In this paper we show that sparse regression for high dimensional data can be carried out directly on a co
'The approach we develop here compresses the data by a random linear or affine transformation, reducing the
'These compressed data can then be made available for statistical analyses; we focus on the problem of spar
'Informally, our theory ensures that the relevant predictors can be learned from the compressed data as wel
```

Gambar 4 Salah Satu Makalah Jurnal Yang Sudah Tersegmentasi

Modul Peringkasan Ekstraktif dimulai dengan melakukan K-Means *Clustering* dan mendapatkan Distance Space. K-Means *Clustering* merupakan metode partisi dari *Clustering* yang paling sering digunakan dan algoritma yang paling populer untuk metode ini [21] [22] [23]. K-Means berusaha mencari partisi optimal dari data dengan cara meminimalisasi kriteria *sum of squared error* secara iteratif. Fungsi objektif mengoreksi jarak antara setiap data points dengan pusat dari klasternya *(centroid)* masing-masing. Fungsi objektif dioptimasi secara iteratif yaitu dengan memilih nilai awal *(initial values)* lalu secara bergantian memperbarui label/klaster untuk pusat saat ini dan kemudian memperbarui pusat untuk label/klaster saat ini [24]. Itu sebabnya penentuan disimilaritas (*distance*/jarak) antara sebuah data point dengan sebuah prototipe klaster/centroid menjadi hal yang sangat utama pada metode partisi. Hal ini terus dilakukan sampai tidak ada lagi data yang berpindah label/klaster. Sederhananya, K-Means bertujuan untuk mengelompokkan data dengan memaksimalkan kemiripan karakter setiap data dalam klaster dan juga memaksimalkan ketidakmiripannya dengan klaster lainnya.

Clustering dilakukan dengan Scikit-Learn menggunakan method `.fit()`. Sementara untuk menghitung *distance space*-nya menggunakan `.transform()`. Jumlah klaster akan menentukan jumlah kalimat yang akan diekstrak sebagai ringkasan akhir. Untuk itu, maka sebelumnya dilakukan pemeriksaan terkait jumlah rata-rata kalimat dari seluruh abstrak. Gambar 5 bagian a menunjukkan rata-rata kalimat seluruh makalah jurnal (265 kalimat) dan bagian b menunjukkan rata-rata jumlah kalimat abstrak (6 kalimat). Rata-rata jumlah kalimat abstrak ini akan dijadikan sebagai jumlah klaster.



Gambar 5 Rata-Rata Jumlah Kalimat Makalah Jurnal (a) Dan Abstrak (b)

Sebelum melakukan peringkasan ekstraktif, beberapa fitur perlu dibuang terlebih dahulu dan menyisakan fitur-fitur yang paling penting saja seperti `clean_abstract, text_clean, cluster,` dan `distance_space`. Beberapa nama kolom juga diganti agar lebih mudah dipahami (`clean_abstract` menjadi `ref_summary`, dan `text_clean` menjadi `segmentized_paper`). Hasil akhirnya seperti ditunjukkan pada Gambar 6.

year	ref_summary	segmentized_paper	cluster	distance_space
1997	We have constructed an inexpensive video based...	[A Neural Network Based Head Tracking System D...	[3, 3, 1, 2, 2, 4, 2, 1, 5, 5, 5, 5, 2, 5, 5, ...	[[5.908768, 5.9025707, 5.2754364, 5.1869826, 6...
2000	Non-negative matrix factorization (NMF) has pr...	[Algorithms for Non-negative Matrix Factoriza...	[3, 5, 1, 3, 3, 0, 2, 2, 2, 3, 3, 3, 3, 3, 3, ...	[[5.624708, 5.2260203, 5.174033, 4.2155256, 5....
2007	Recent research has studied the role of sparsi...	[Compressed Regression Shuheng Zhou∗, John Laf...	[2, 3, 5, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, ...	[[5.4285326, 5.316185, 4.743139, 5.9228354, 5....
2007	It is becoming increasingly important to learn...	[Predictive Matrix-Variate t Models Shenghuo Z...	[3, 0, 3, 3, 4, 5, 1, 5, 3, 3, 1, 3, 3, 5, 3, ...	[[5.4190125, 4.915897, 5.216683, 4.0999618, 4....
2007	Variational methods are frequently used to app...	[Loop Series and Bethe Variational Bounds in A...	[5, 4, 5, 5, 5, 5, 5, 5, 1, 1, 5, 1, 5, 5, 5, ...	[[5.701174, 4.790844, 5.596019, 5.9524326, 6.3...
...	...	...	...	...

Gambar 6 Hasil Akhir Seleksi Fitur

Pertama, setiap makalah jurnal yang sudah tersegmentasi akan dipecah menjadi dataframe sendiri *(sampling)* yang baris-barisnya terdiri dari kalimat-kalimat. Setiap kalimat kemudian ditetapkan klaster beserta *distance space*-nya masing-masing. Selanjutnya adalah mendapatkan ringkasan (hasil akhir ditunjukkan Gambar 7):

- Setiap kalimat akan dikelompokkan berdasarkan klasternya
- Urutkan anggota setiap klaster (kalimat-kalimat yang ada pada setiap klaster) secara menaik *(ascending)* berdasarkan *distance space*-nya dengan centroid
- Pilih kalimat yang berada pada urutan pertama sebagai kandidat ringkasan dari setiap klaster. Ini karena kalimat yang berada pada urutan pertama merupakan kalimat dengan jarak paling kecil dengan *centroid*
- Tata kembali urutan setiap kalimat terpilih ini berdasarkan posisi indeks mereka pada sekuens aslinya (`segmentized_paper`)
- Hasil ringkasan setiap makalah jurnal disimpan dalam kolom `pred_summary`

year	ref_summary	segmentized_paper	cluster	distance_space	pred_summary
1997	We have constructed an inexpensive video based...	[A Neural Network Based Head Tracking System D...	[3, 3, 1, 2, 2, 4, 2, 1, 5, 5, 5, 5, 2, 5, 5, ...	[[5.908768, 5.9025707, 5.2754364, 5.1869826, 6...	Similarly, an active video conferencing system...
2000	Non-negative matrix factorization (NMF) has pr...	[Algorithms for Non-negative Matrix Factoriza...	[3, 5, 1, 3, 3, 0, 2, 2, 2, 3, 3, 3, 3, 3, ...	[[5.624708, 5.2260203, 5.174033, 4.2155256, 5....	In this submission, we analyze in detail two n...
2007	Recent research has studied the role of sparsi...	[Compressed Regression Shuheng Zhou*, John Laf...	[2, 3, 5, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, ...	[[5.4285326, 5.316185, 4.743139, 5.9228354, 5....	The data are compressed is a random m × p matr...
2007	It is becoming increasingly important to learn...	[Predictive Matrix-Variate t Models Shenghuo Z...	[3, 0, 3, 3, 4, 5, 1, 5, 3, 3, 1, 3, 3, 5, 3, ...	[[5.4190125, 4.915897, 5.216683, 4.0999618, 4....	(a) MVTM.We review three existing models and i...
2007	Variational methods are frequently used to app...	[Loop Series and Bethe Variational Bounds in A...	[5, 4, 5, 5, 5, 5, 5, 5, 1, 1, 5, 1, 5, 5, 5, ...	[[5.701174, 4.790844, 5.596019, 5.9524326, 6.3...	The partition function then equals Z(τ) = 1 +...

Gambar 7 Kondisi Terkini Dataset Dengan Hasil Ringkasan

C. Analisis Hasil Performa Model

Untuk selanjutnya, model peringkasan ekstraktif yang diusulkan dalam penelitian ini akan disebut dengan BESKlus *(BERT Extractive Summarization with K-Means Clustering)*. Tahapan selanjutnya adalah melakukan evaluasi performanya. Metrik yang digunakan untuk mengukur performanya adalah ROUGE dan BERTScore. Perbandingan hasil performa juga dilakukan dengan model terdahulu. Model terdahulu yang akan digunakan adalah PyTextRank dan BERT Extractive Summarizer. Penjelasan lebih detail tentang evaluasi performa dan perbandingannya dengan model lainnya akan dipaparkan pada bagian III. Hasil dan Pembahasan.

III. HASIL DAN PEMBAHASAN

A. Evaluasi Performa Model

NeurIPS dataset digunakan untuk eksperimen pengembangan BESKlus ini. Dataset berjumlah 6101 makalah jurnal ilmiah yang terdiri dari beberapa tahun penerbitan seperti ditunjukkan pada Tabel 1.

TABEL 1
SEBARAN JURNAL ILMIAH PADA DATASET TERKINI

Tahun	Jumlah Jurnal
1997	1
2000	1
2007	95
2008	216
2009	245
2010	281
2011	280
2012	333
2013	353
2014	403
2015	396
2016	551
2017	664
2018	984
2019	1298

Evaluasi performa akan menggunakan ROUGE-L dan BERTScore. Evaluasi akan menilai *Recall*, *Precision* dan *F-Measure*. Ketiga skor ini akan dibutuhkan untuk memprediksi cakupan antara *reference summary* (abstrak) dan *prediction summary* (hasil model). Pada ROUGE-L, *Precision* akan menghitung berapa banyak LCS dari *reference summary* dan *prediction summary* dan membagi nilainya dengan jumlah sekuens *preditcion summary*. *Recall* mirip dengan *Precision* tetapi dibagi dengan jumlah sekuens *reference summary*. F1 merupakan nilai pusat atau rata-rata harmonik antara *Recall* dan *Precision*.

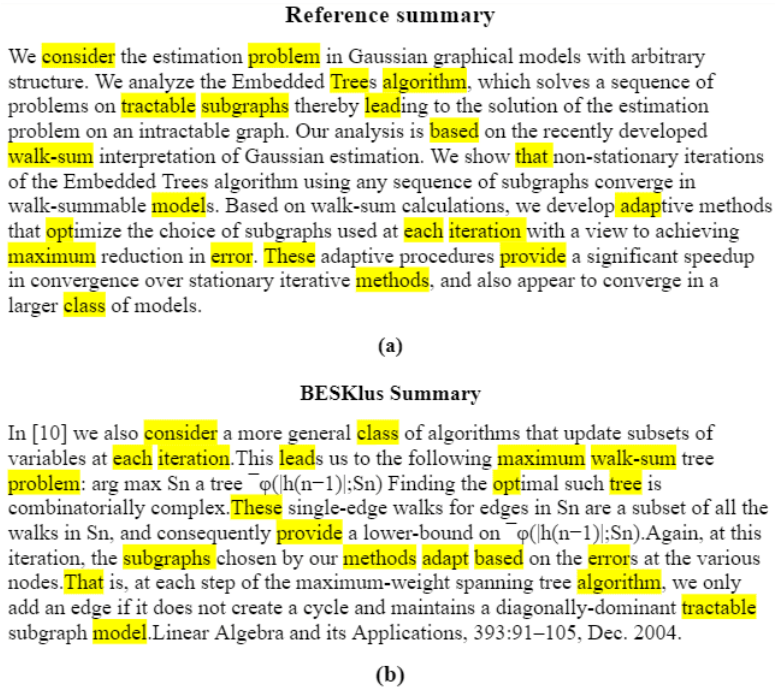
Sementara dalam BERTScore, *Precision* diperoleh dengan cara mencocokkan *(pairwise cosine similarity)* token-token dalam *prediction summary* dengan token-token pada *reference summary*. *Recall* adalah kebalikannya. F1 merupakan kombinasi keduanya. Tabel 2 menunjukkan hasil rata-rata evaluasi ringkasan seluruh makalah jurnal. Tabel 2 menunjukkan hasil rata-rata evaluasi ringkasan seluruh makalah jurnal.

TABEL 2
HASIL EVALUASI PERFORMA BESKLUS

Metrik	Skor F1
ROUGE-L	15.52
BERTScore	85.55

Gambar 8 menunjukkan salah satu makalah jurnal dengan *reference summary/abstract* (a) dan *prediction summary* (b). Jurnal ini berjudul *Adaptive Embedded Subgraph Algorithms using Walk-Sum Analysis* dan dipublikasikan pada tahun 2007 oleh Chandrasekaran dkk. Skor ROUGE-L untuk makalah jurnal ini adalah *Recall* = 17%, *Precision* = 18%, dan F1 = 18%. Dari skor ini, dapat ditelusuri LCS yang dimiliki kedua ringkasan. Panjang (jumlah kata) ringkasan a = 123, dan ringkasan b = 118. Dengan formula Recall, dapat ditemukan bahwa:

$LCS\ (a, b) = \ 0.17\ \times 123 = 20.91$ atau dibulatkan menjadi 21. Teks berwarna kuning dianggap sebagai LCS. Pada saat scoring, parameter `use_stemmer = True`, sehingga kata-kata dikembalikan terlebih dahulu ke dalam bentuk dasar. Itu sebabnya kata sub-urutan seperti *adaptive/adapt*, *optimize/optimal*, *model/models*, dll dianggap sebagai LCS juga.



Reference summary

We consider the estimation problem in Gaussian graphical models with arbitrary structure. We analyze the Embedded Trees algorithm, which solves a sequence of problems on tractable subgraphs thereby leading to the solution of the estimation problem on an intractable graph. Our analysis is based on the recently developed walk-sum interpretation of Gaussian estimation. We show that non-stationary iterations of the Embedded Trees algorithm using any sequence of subgraphs converge in walk-summable models. Based on walk-sum calculations, we develop adaptive methods that optimize the choice of subgraphs used at each iteration with a view to achieving maximum reduction in error. These adaptive procedures provide a significant speedup in convergence over stationary iterative methods, and also appear to converge in a larger class of models.

(a)

BESKlus Summary

In [10] we also consider a more general class of algorithms that update subsets of variables at each iteration.This leads us to the following maximum walk-sum tree problem: arg max Sn a tree ¯φ(|h(n−1)|;Sn) Finding the optimal such tree is combinatorially complex.These single-edge walks for edges in Sn are a subset of all the walks in Sn, and consequently provide a lower-bound on ¯φ(|h(n−1)|;Sn).Again, at this iteration, the subgraphs chosen by our methods adapt based on the errors at the various nodes.That is, at each step of the maximum-weight spanning tree algorithm, we only add an edge if it does not create a cycle and maintains a diagonally-dominant tractable subgraph model.Linear Algebra and its Applications, 393:91–105, Dec. 2004.

(b)

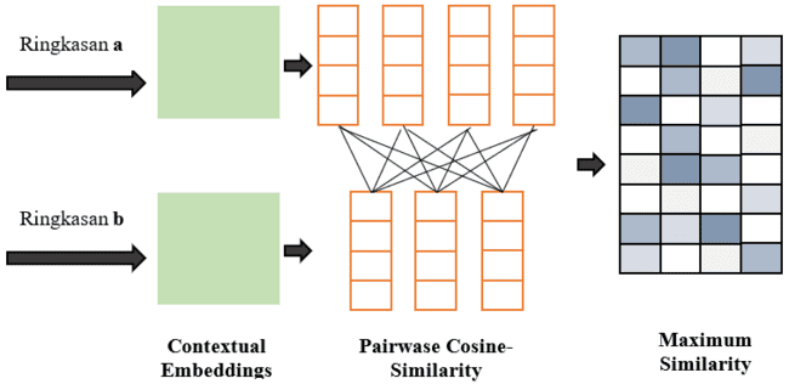
Gambar 8 Hasil Ringkasan Salah Satu Makalah Jurnal Ilmiah

ROUGE-L mampu menangkap struktur sintaktik pada level kalimat, namun tidak pada sisi semantiknya. Selain itu, alternatif LCS lainnya dan sub-urutan yang lebih pendek tidak terhitung pada hasil skor akhir [25]. Misalkan pada kalimat *"he killed the dog"* dibandingkan dengan kalimat *"the dog he killed"*, ROUGE-L hanya akan memilih salah satu antara *the dog* atau *he killed* sebagai LCS. Karenanya, untuk memperoleh hasil pengukuran performa yang mampu menangkap makna

semantik dan konteks dari setiap kalimat, pengukuran selanjutnya akan menggunakan BERTScore. Pada contoh makalah jurnal yang sama, skor yang diperoleh adalah *Precision* = 88.35%, *Recall* = 86.10%, dan F1 = 87.2%.

Jika dibandingkan dengan ROUGE-L hasil evaluasi yang diberikan BERTScore jauh lebih tinggi. BERTScore melakukan pencocokan kata-kata pada dua ringkasan dengan menggunakan *cosine-similarity*. Method `.score` akan mengembalikan nilai *precision, recall* dan F1 dalam bentuk tensor.

Reference summary dan *prediction summary* direpresentasikan menggunakan *contextual embeddings*. *Contextual embeddings* ini dikalkulasikan menggunakan model Longformer [26] yang akan memanfaatkan *self-attention* lokal dan global. Pemilihan model ini juga karena kemampuannya menangani panjang sekuens kalimat lebih dari 512 (max-length berbagai model berbasis transformer lainnya). Panjang sekuens maksimal yang bisa ditangani Longformer adalah 4094.



Gambar 9 Ilustrasi Proses Evaluasi Hasil Ringkasan Sebuah Makalah Jurnal Dengan BERTScore

Gambar 9 menunjukkan ilustrasi dari proses evaluasi makalah jurnal diatas dengan BERTScore. Tahapannya sebagai berikut:

1. Ambil *contextual embedding* dari **Ringkasan a** dan **Ringkasan b**.
2. Hitung *matrix cosine-similarity* setiap kata/token pada **Ringkasan a** dan **Ringkasan b**.
   a. *Precision*: Untuk setiap kata dari **Ringkasan b,** temukan kata yang memiliki similaritas paling tinggi dari **Ringkasan a** dan hitung *average of the maxima* seluruh kata dalam **Ringkasan b**.
   b. *Recall*: Lakukan hal sebaliknya dari *Precision*.
3. Hitung skor F1, yaitu *harmonic average* dari *Precision* dan *Recall*.
4. Visualisasi lengkap dari proses pada bagian Maximum Similarity ditampilkan pada *Lampiran A* dan potongan visualisasinya untuk satu kalimat saja dapat dilihat pada *Lampiran B.*

BESKlus sendiri dibangun diatas arsitektur S-BERT yang melakukan *contextual embeddings* pada setiap sekuens inputnya. Model yang dibangun dengan *contextual embeddings* akan lebih tepat diukur dengan metrik yang menggunakan *contextual embeddings* sebagai dasar pengukurannya. Itu sebabnya pengukuran dengan BERTScore jauh lebih tinggi dari ROUGE-L.

Dari sebagian besar penelitian terkait peringkasan teks otomatis tidak pernah ada yang menyebutkan skor minimum performa sebuah model peringkasan [5]. Semua pengukuran mengacu kepada *gold standard/golden summary*. *Gold standard* ini merupakan *human judgement* atau hasil ringkasan manusia [14]. Pada artikel berita, *gold standard* ini dapat berupa *highlight* dalam bentuk poin-poin atau *headline* [14]. Jika dalam makalah jurnal ilmiah, maka abstraknya yang menjadi *gold standard* [27] [5]. Semakin tinggi skornya, maka dianggap semakin mendekati *gold standard* dan semakin baik pula performa model tersebut dalam menghasilkan ringkasan.

B. Komparasi dengan Model Terdahulu

Penelitian ini juga membandingkan model BESKlus dengan model peringkasan *unsupervised* yang sudah ada sebelumnya seperti PyTextRank dan BERT Extractive Summarizer. PyTextRank [28] merupakan modifikasi yang dilakukan Paco Nathan terhadap TextRank versi Mihalcea dan Tarau [29] sebagai sebuah *pipeline extension* dari spaCy. spaCy merupakan library untuk tugas PBA tingkat lanjut dan menggunakan algoritma-algoritma paling mutakhir untuk fitur-fitur NER, POS *tagging*, *dependency parsing, word vectors*, dll. Pytextrank dikembangkan untuk tugas PBA berbasis *graph* untuk ekstraksi frase dan juga peringkasan ekstraktif dokumen. Pengembangan PyTextRank ini jauh lebih baik dari model yang pertama kali diusulkan [29]. Modifikasi dan peningkatan yang diberikan adalah [28]:

- Penggunaan lematisasi menggantikan stemming

- Integrasi dengan spaCy sebagai komponen *pipeline*
- Peringkasan ekstraktif sederhana berbasis jarak vektor dari *ranked phrases*
- Peningkatan *preprocessing* melalui *noun chunking* dan NER
- Mengikutsertakan *verbs* di dalam *graph*

Sementara BERT Extractive Summarizer merupakan model yang dikembangkan oleh Miller [13]. Model ini dibangun berbasis BERT dan juga menggunakan *clustering* untuk mengelompokkan kalimat kandidat ringkasannya.

Komparasi hanya dilakukan pada satu makalah jurnal saja, yaitu makalah jurnal *Adaptive Embedded Subgraph Algorithms using Walk-Sum Analysis*. Hal ini disebabkan karena keterbatasan sumber daya dan waktu dalam pengolahan. Perangkat yang digunakan adalah Google Colab Pro dengan GPU NVIDIA Tesla P100 (16 GB). Pada PyTextRank, tahapan pengolahan frase untuk satu makalah jurnal saja membutuhkan waktu 1 menit 50 detik. Jika dikalikan dengan 6101 makalah jurnal, maka untuk tahapan pengolahan frase saja membutuhkan waktu 11,185 menit atau 186 jam. Hal ini jauh berbeda ketika menggunakan SBERT untuk melakukan pengolahan makalah jurnal pada tahapan *contextual embedding*. Proses embedding membutuhkan 2.6 detik/makalah jurnal atau 264 menit atau 4.5 jam untuk 6101 makalah jurnal. Hasil komparasi ada pada Tabel 3.

TABEL 3
HASIL KOMPARASI BESKLUS, PYTEXTRANK, DAN BERT EXT. SUMMARIZER

Model	**Metrik**					
	ROUGE-L			BERTScore		
	R	P	F1	R	P	F1
BESKlus	**17.64%**	**18.89%**	**18.25%**	**86.10%**	**88.35%**	**87.21%**
PyTextRank [28]	17.21%	16.53%	16.86%	84.16%	84.66%	84.28%
BERT Ext. Summ [13]	22.34%	31.49%	26.14%	96.70%	78.24%	86.50%

Dari hasil pada Tabel 3 dapat dilihat model yang diusulkan (BESKlus) memiliki skor F1 yang lebih tinggi daripada PyTextRank dan BERT Ext. Summarizer jika dievaluasi menggunakan BERTScore. Pada evaluasi menggunakan ROUGE-L, model BESKlus juga memiliki skor lebih tinggi dibandingkan dari PyTextRank namun lebih rendah dari BERT Ext. Summarizer. Hasil ringkasan semua model dapat dilihat pada *Lampiran C*.

Dari sisi BERTScore, hasil yang dicapai PyTextRank ini cukup mendekati hasil model lainnya yang berbasis BERT (BESKlus dan BERT Ext. Summ). Hasil ini disebabkan adanya peningkatan fitur dari sisi integrasinya dengan spaCy dan juga fitur peringkasan ekstraktifnya yang berbasis jarak vektor. Hal ini berbeda dengan TextRank versi awal [29] yang mengekstraksi kalimat, menjadikannya *graph* dan memberikan *vertex* pada kalimat-kalimat yang memiliki relasi dilihat dari *word overlap*. *Overlap* dari dua kalimat dapat ditentukan sebagai jumlah token umum *(common tokens)* antar representasi leksikal kedua kalimat atau dengan cara menghitung kata-kata dari kategori sintaktik tertentu seperti *open class words*, *noun* dan *verbs*, dll.

Pada PyPyTextRank, setiap kombinasi frase di vektorisasi lalu di-ranking untuk menemukan frase-frase terpenting dari keseluruhan dokumen. Pada contoh makalah jurnal yang digunakan, PyTextRank memperoleh frase-frase tertinggi sebagai berikut: *walks, valid walks, adaptive tree algorithm, Gaussian models, non-walk- summable models*, dan *such models*. Frase dengan tingkat tertinggi tersebut akan dijadikan acuan untuk memeriksa isi setiap kalimat. Kalimat-kalimat yang memiliki frase-frase tertinggi tersebut akan dijadikan sebagai kandidat ringkasan. Hal ini dilakukan dengan melakukan iterasi pada seluruh kalimat pada dokumen dan mengkalkulasikan Jarak Euclidean-nya dengan frase-frase tersebut. Kalimat-kalimat tersebut kemudian di-*ranking* lagi dari jarak paling kecil sampai paling besar. Enam kalimat teratas akan dipilih sebagai ringkasan. Gambar 10 menunjukkan potongan ranking Jarak Euclidean (a) dan kalimat-kalimat terpilih (b).

```
[(55, 0.3651929037230042),
 (58, 0.3651929037230042),
 (66, 0.3651929037230042),
 (93, 0.3651929037230042),
 (165, 0.3651929037230042),
 (121, 0.3683790900548188),
 (199, 0.37321093994539417),
 (33, 0.3776367838899317),
 (203, 0.3776367838899317),
```

(a)

```
ic| sent_id: 55
    sent_text[sent_id]: ('Other algorithms may compute walks according to different orders (rather '
                         'than length-based orders).')
ic| sent_id: 58
    sent_text[sent_id]: ('Based on the absolute convergence condition, walk-summability implies that '
                         'walk-sums over a countable set of walks in G can be computed in any order.')
ic| sent_id: 66
    sent_text[sent_id]: ('Concatenation of walks We briefly describe the concatenation operation for '
                         'walks and walk-sets, which plays a key role in walk-sum analysis.')
ic| sent_id: 93
    sent_text[sent_id]: ('Consider the following recursively defined set of walks for s, t ∈ V : Wn(s → '
                         't) = (cid:20) = Wn−1(s → ∗)')
ic| sent_id: 165
    sent_text[sent_id]: ('Such edges are useful for capturing walks ending at the high-error nodes, '
                         'which contribute to the set of walks in (5).')
ic| sent_id: 121
    sent_text[sent_id]: 'Validity: The walks in Wn are valid walks in G, i.e. Wn(s → t) ⊆'
```

(b)

Gambar 10 Hasil Ranking Kalimat Kandidat Ringkasan Menggunakan PyTextRank

Dari beberapa kali hasil uji coba peringkasan dengan BERT Ext. Summarizer, hasilnya selalu menunjukkan adanya judul, nama penulis, dan kalimat pertama dari abstrak. Gambar 11 menunjukkan hasil peringkasan menggunakan model BERT Ext. Summarizer pada contoh makalah jurnal lainnya. Teks berwarna kuning berisi judul, teks berwarna hijau berisi penulis, departemen, dan instansinya, serta teks berwarna merah berisi kalimat pertama abstrak. Jika dilihat dari struktur kalimat dan tanda bacanya, ketiga teks ini masih dalam satu kalimat. Mengingat segmentasi kalimat yang digunakan BERT Ext. Summarizer mengacu pada tanda titik, maka asumsi sementara adalah BERT Ext. Summarizer selalu mengambil kalimat pertama dari dokumen sebagai kandidat ringkasannya. Pengaturan seperti ini kurang baik jika mau menghasilkan hasil ringkasan yang benar-benar hasil prediksi dari model *machine learning*. Hasil ringkasan seperti ini akan membuat hasil evaluasi menjadi bias karena terdapat sekuens yang sama persis antara *reference summary* dan *prediction summary*.

Reference Summary	Prediction Summary
Recent research has studied the role of sparsity in high dimensional regression and signal reconstruction, establishing theoretical limits for recovering sparse models from sparse data. In this paper we study a variant of this problem where the original n input variables are compressed by a random linear transformation to $m \ll n$ examples in p dimensions, and establish conditions under which a sparse linear model can be successfully recovered from the compressed data. A primary motivation for this compression procedure is to anonymize the data and preserve privacy by revealing little information about the original data. We characterize the number of random projections that are required for ℓ_1-regularized compressed regression to identify the nonzero coefficients in the true model with probability approaching one, a property called ``sparsistence." In addition, we show that $\ell_1$-regularized compressed regression asymptotically predicts as well as an oracle linear model, a property called ``persistence." Finally, we characterize the privacy properties of the compression procedure in information-theoretic terms, establishing upper bounds on the rate of information communicated between the compressed and uncompressed data that decay to zero.	Compressed Regression Shuheng Zhou∗ John Lafferty∗† Larry Wasserman‡† ∗Computer Science Department ‡Department of Statistics †Machine Learning Department Carnegie Mellon University Pittsburgh, PA 15213 Abstract Recent research has studied the role of sparsity in high dimensional regression and signal reconstruction, establishing theoretical limits for recovering sparse models from sparse data. Information Resistence (Propositions 5.1 and 5.2): The rate at which information about X is kβk1≤Ln,m keY − eX βk2 −→ 0, as P revealed by the compressed dataeX satisfies rn,m = sup I (X;eX) supremumisoverdistributionsontheoriginaldata X. np = O(cid:0) m n(cid:1) → 0, where the As summarized by these results, compressed regression is a practical procedure for sparse learning in high dimensional data that has provably good properties. While compressed sensing of X allows a sparse X to be reconstructed from a small number of random measurements, our goal is to reconstruct a sparse function of X. Indeed, from the point of view of privacy, approximately reconstructing X, which compressed sensing shows is possible if X is sparse, should be viewed as undesirable; we return to this point in Section ??. f = 5 f = 10 f = 20 f = 40 f = 80 f = 120 0.0 0.5 1.0 2.0 Control parameter q 1.5 2.5 3.0 1 Figure 1: Plots of the number of samples versus the probability of success for recovering sgn(β∗). Sparse signal detection from incoherent projections. 8] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani.

Gambar 11 Hasil Ringkasan BERT Ext. Summarizer Pada Contoh Makalah Jurnal Lainnya

IV. SIMPULAN

Penelitian ini memperkenalkan model BESKlus, sebuah model peringkasan teks otomatis ekstraktif dengan *contextual embedding* menggunakan kombinasi Sentence-BERT dan K-Means *Clustering*. Dataset yang digunakan adalah kumpulan makalah jurnal ilmiah dari NeurIPS. *Contextual embedding* dilakukan pada level kalimat menggunakan SBERT. Hasil dari *embedding* kemudian dikelompokkan dengan K-Means *Clustering*. Jumlah klaster disesuaikan dengan rata-rata jumlah kalimat pada abstrak seluruh makalah jurnal. Kalimat-kalimat terdekat dengan centroid pada setiap klaster akan diambil sebagai kandidat-kandidat ringkasan. Eksperimen menunjukkan hasil pengujian performa dengan ROUGE-L dan BERTScore memperoleh skor performa yang lebih baik dari model lainnya (PyTextRank dan BERT Ext. Summarizer).

Hasil dari pengukuran tersebut membuktikan bahwa penggunaan *contextual embedding* sangat baik jika diterapkan pada peringkasan teks ekstraktif yang umumnya dilakukan pada level kalimat. Model BESKlus juga membuktikan bahwa arsitektur atau algoritma tradisional (K-Means *Clustering*) dikombinasikan dengan arsitektur terbaru (SBERT) masih dapat menghasilkan performa yang sangat baik.

Kelemahan dari model BESKlus ini adalah masih menangkap persamaan-persamaan atau formula-formula dalam notasi matematika. Hal ini membuat skornya menjadi kurang begitu baik ketika harus dikomparasi dengan *reference summary* yang tidak mengandung formula sama sekali. Selain itu, model juga masih menangkap bagian-bagian dari daftar pustaka. Ini mungkin disebabkan karena umumnya daftar pustaka mengandung judul literatur yang panjang dan model menganggapnya sebagai bagian dari isi makalah jurnal, apalagi jika sekuens dari judul tersebut juga tertuang pada isi makalah jurnal (misalkan ada di *Introduction* atau *Related Works*).

Saran untuk penelitian selanjutnya adalah bisa berfokus pada beberapa hal di antaranya persiapan dataset, alternatif penghitungan *distance*, peringkasan per bagian makalah jurnal, dan penggunaan dataset berbahasa Indonesia. Pada persiapan dataset, perlu mencari cara untuk menangani *formula* dan *reference*. Model perlu dilatih untuk mengenal semua *formula* dan menyimpannya dalam *vocabulary*. Hal ini akan membantu proses embedding memvektorisasi formula dengan tepat (sesuai konteksnya, formula tersebut tentang apa). Tantangan lain dalam menangani formula ini juga adalah beberapa penulis makalah jurnal mencantumkan formula dalam bentuk gambar, bukan diolah *typesetting system* seperti misalnya MathJax atau LaTeX. Dalam menangani *reference* atau rujukan, model bisa dikembangkan agar menggunakan kalimat-kalimat yang mengandung rujukan sebagai bagian dari ringkasan seperti yang dilakukan Yasunaga dkk [4]. Kalimat-kalimat mengandung rujukan ini dapat dipandang sebagai ringkasan singkat makalah jurnal yang dirujuk dari perspektif penulisnya. Hal ini akan semakin memperkaya hasil ringkasan sebuah makalah jurnal karena tidak hanya mengandung pandangan penulis makalah jurnal tersebut tetapi juga pandangan komunitas peneliti lain terhadap topik yang diangkat.

Penggunaan *distance* bisa menggunakan alternatif lainnya seperti *cosine similarity* menggantikan *euclidean distance* saat menghitung jarak pada klaster. Peringkasan per bagian makalah jurnal juga dapat dijadikan pengembangan selanjutnya karena setiap bagian makalah jurnal punya hal utama yang mau disampaikan. Seperti yang dilakukan oleh Erera dkk dari IBM Research [6] yang merancang IBM Science Summarizer untuk meringkas bagian-bagian makalah jurnal secara independen yang memungkinkan pengguna bisa menemukan informasi utama dari setiap bagian.

Perancangan model serupa namun dengan dataset berbahasa Indonesia akan menjadi tantangan tersendiri karena berbagai *pretrained model* yang tersedia saat ini mayoritas masih dilatih menggunakan dataset berbahasa Inggris. SBERT sendiri dalam menghasilkan *sentence embedding* yang mengandung makna kontekstual setiap kata dilatih menggunakan dataset SNLI dan MultiNLI. BERTScore dalam proses evaluasinya juga dilatih menggunakan WMT18 *metric evaluation dataset* yang terdiri 14 pasang bahasa (Inggris ke Ceko, Jerman, Estonia, Finlandia, Rusia, Turki, dan enam bahasa tersebut ke Inggris). Pengembangan model serupa dengan dataset berbahasa Indonesia dinilai akan semakin memperkaya riset Pemrosesan Bahasa Alami di Indonesia khususnya untuk tugas peringkasan teks otomatis.

UCAPAN TERIMA KASIH

Ucapan terima kasih disampaikan kepada Koordinator Mata Kuliah Tesis, Dosen Pembimbing 1 dan 2 atas bimbingan dan arahannya selama proses penelitian dan menyusun laporan. Ucapan terima kasih juga disampaikan kepada semua pihak yang hasil riset dan penelitiannya dikutip untuk membangun dan memperkaya penelitian ini. *I stand on the shoulders of giants*.

DAFTAR PUSTAKA

[1] R. Swami, "Kaggle," 2 Desember 2020. [Online]. Available: https://www.kaggle.com/rowhitswami/nips-papers-1987-2019-updated?select=papers.csv. [Accessed 10 Juni 2021].

[2] J. F. McCarthy and W. G. Lehnert, "Using Decision Trees for Coreference Resolution," in *IJCAI*, 1995.

[3] S. K. Bharti, K. S. Babu and S. K. Jena, "Automatic Keyword Extraction for Text Summarization: A Survey," *arXiv*, 2017.

[4] M. Yasunaga, J. Kasai, R. Zhang, A. R. Fabbri, I. Li, D. Friedman and D. R. Radev, "ScisummNet: A Large Annotated Corpus and Content-Impact Models for Scientific Paper Summarization with Citation Networks," in *Proceedings of Association for the Advancement of Artificial Intelligence 2019*, 2019.

[5] N. I. Altmami and M. E. B. Menai, "Automatic Summarization of Scientific Articles: A Survey," *Journal of King Saud University - Computer and Information Sciences*, 2020.

[6] S. Erera, M. Shmueli-Scheuer, G. Feigenblat, O. P. Nakash, O. Boni, H. Roitman, D. Cohen, B. Weiner, Y. Mass, O. Rivlin, G. Lev, A. Jerbi, J. Herzig, Y. Hou, C. Jochim, M. Gleize, F. Bonin, D. Ganguly and D. Konopnicki, "A Summarization System for Scientific Documents," in *Proceedings of the 2019 EMNLP and the 9th IJCNLP (System Demonstrations)*, Hong Kong, 2019.

[7] E. Collins, I. Augenstein and S. Riedel, "A Supervised Approach to Extractive Summarisation of Scientific Papers," in *Proceedings of the 21st Conference on Computational Natural Language Learning 2017*, Vancouver, 2017.

[8] M. R. Prathima and H. R. Divakar, "Automatic Extractive Text Summarization Using K-Means Clustering," *International Journal of Computer Sciences and Engineering*, vol. VI, no. 6, pp. 782-787, 2018.

[9] H. C. Manh, L. H. Thanh and L. T. Minh, "Extractive Multi-document Summarization using K-means, Centroid-based Method, MMR, and Sentence Position," in *The Tenth International Symposium on Information and Communication Technology*, New York, 2019.

[10] A. Rogers, O. Kovaleva and A. Rumishky, "A Primer in BERTology: What we know about how {BERT} works," *Transactions of the Association for Computational Linguistics*, vol. VIII, pp. 842-866, 2021.

[11] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv*, no. 2, 2019.

[12] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Conference on Empirical Methods in Natural Language Processing*, Hong Kong, 2019.

[13] D. Miller, "Leveraging BERT for Extractive Text Summarization on Lectures," *arXiv*, 2019.

[14] Y. Liu and M. Lapata, "Text Summarization with Pretrained Encoders," *arXiv*, no. 2, 2019.

[15] spaCy, "Guide: Linguistic Features," spaCy, [Online]. Available: https://spacy.io/usage/linguistic-features#sbd. [Accessed 25 January 2022].

[16] S. Ravichandiran, Getting Started with Google BERT, Birmingham: Packt Publishing, 2021.

[17] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *International Conference on Learning Representations*, Scottsdale, 2013.

[18] J. Pennington, R. Socher and C. Manning, "GloVe: Global Vectors for Word Representation," in *Conference on Empirical Methods in Natural Language Processing*, Doha, 2014.

[19] Q. Liu, M. J. Kusner and P. Blunsom, "A Survey on Contextual Embeddings," *Computing Research Repository (CoRR)*, vol. abs/2003.07278, 2020.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez and Ł. Kaiser, "Attention Is All You Need," in *Neural Information Processing Systems*, Long Beach, 2017.

[21] V. Kumar, J. K. Chhabra and D. Kumar, "Performance Evaluation of Distance Metrics in the Clustering Algorithms," *INFOCOMP Journal of Computer Science*, vol. XIII, no. 1, pp. 38-51, 2014.

[22] V. Cohen-Addad, V. Kanade, F. Mallmann-Tren and C. Mathieu, "Hierarchical Clustering: Objective Functions and Algorithms," in *Association for Computing Machinery*, New York, 2017.

[23] K. P. Sinaga and M.-S. Yang, "Unsupervised K-Means Clustering Algorithm," *IEEE Access*, vol. VIII, p. 80716, 2020.

[24] A. Hertzmann, D. Fleet and M. Brubaker, Machine Learning and Data Mining: Lecture Notes, Toronto: University of Toronto Scarborough, 2015.

[25] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Proceedings of Workshop on Text Summarization Branches Out*, Barcelona, 2004.

[26] I. Beltagy, M. E. Peters and A. Cohan, "Longformer: The Long-Document Transformer," *Computing Research Repository (CoRR)*, vol. abs/2004.05150, 2020.

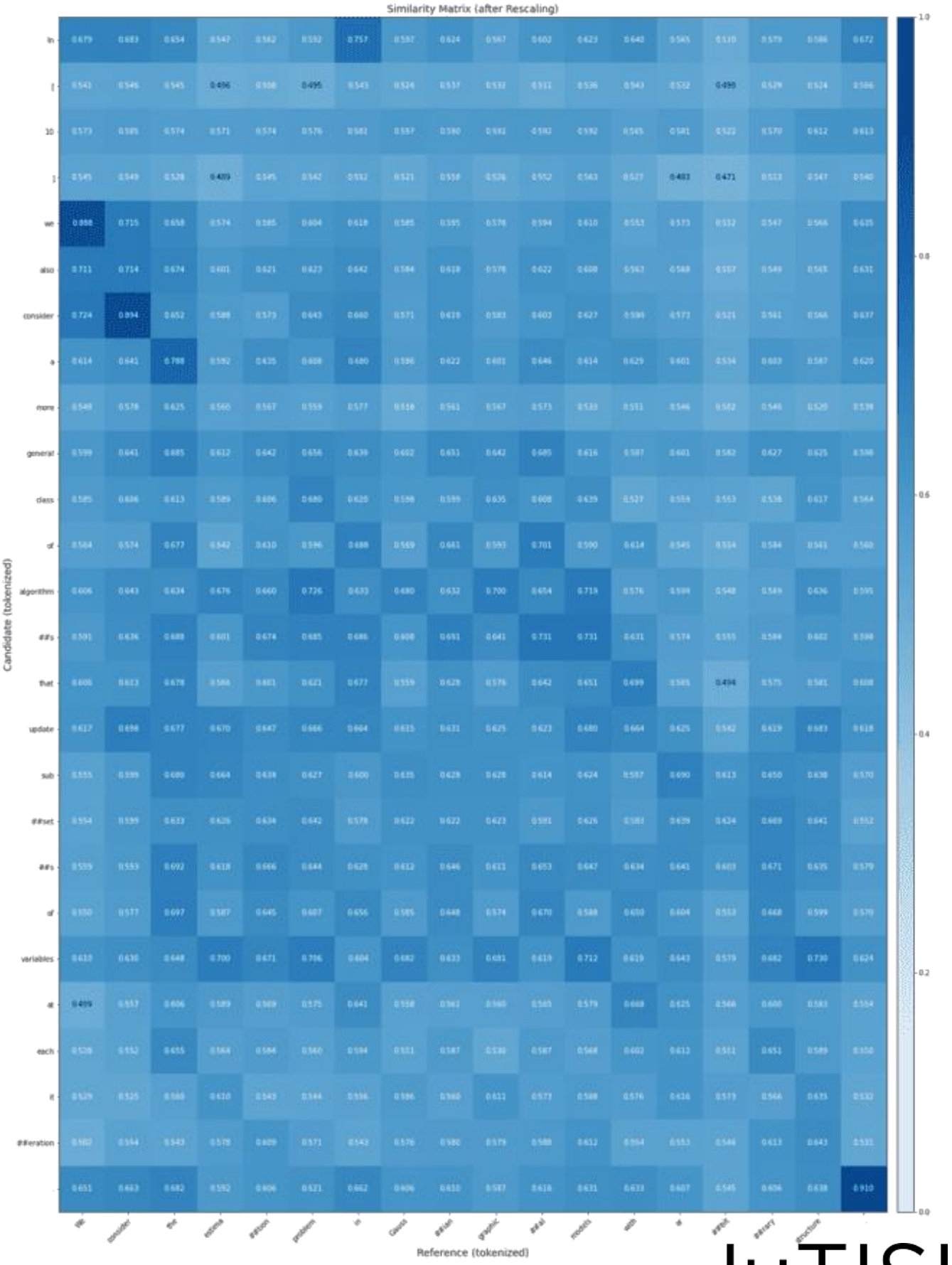
[27] B. Tan, V. Kieuvongngam and Y. Niu, "Automatic Text Summarization of COVID-19 Medical Research Articles using BERT and GPT-2," *arXiv*, 2020.

[28] P. Nathan, "DerwenAI/pytextrank," 29 June 2021. [Online]. Available: https://derwen.ai/docs/ptr/. [Accessed 26 January 2022].

[29] R. Mihalcea and P. Tarau, "TextRank: Bringing Order into Texts," in *Conference on Empirical Methods in Natural Language Processing*, Barcelona, 2004.

p-ISSN : 2443-2210
e-ISSN : 2443-2229

LAMPIRAN A



Potongan visualisasi tersebut menampilkan visualisasi *maximum similarity* untuk kalimat pertama dari masing-masing ringkasan:

Ref. summary:

```
We consider the estimation problem in Gaussian graphical
models with arbitrary structure.
```

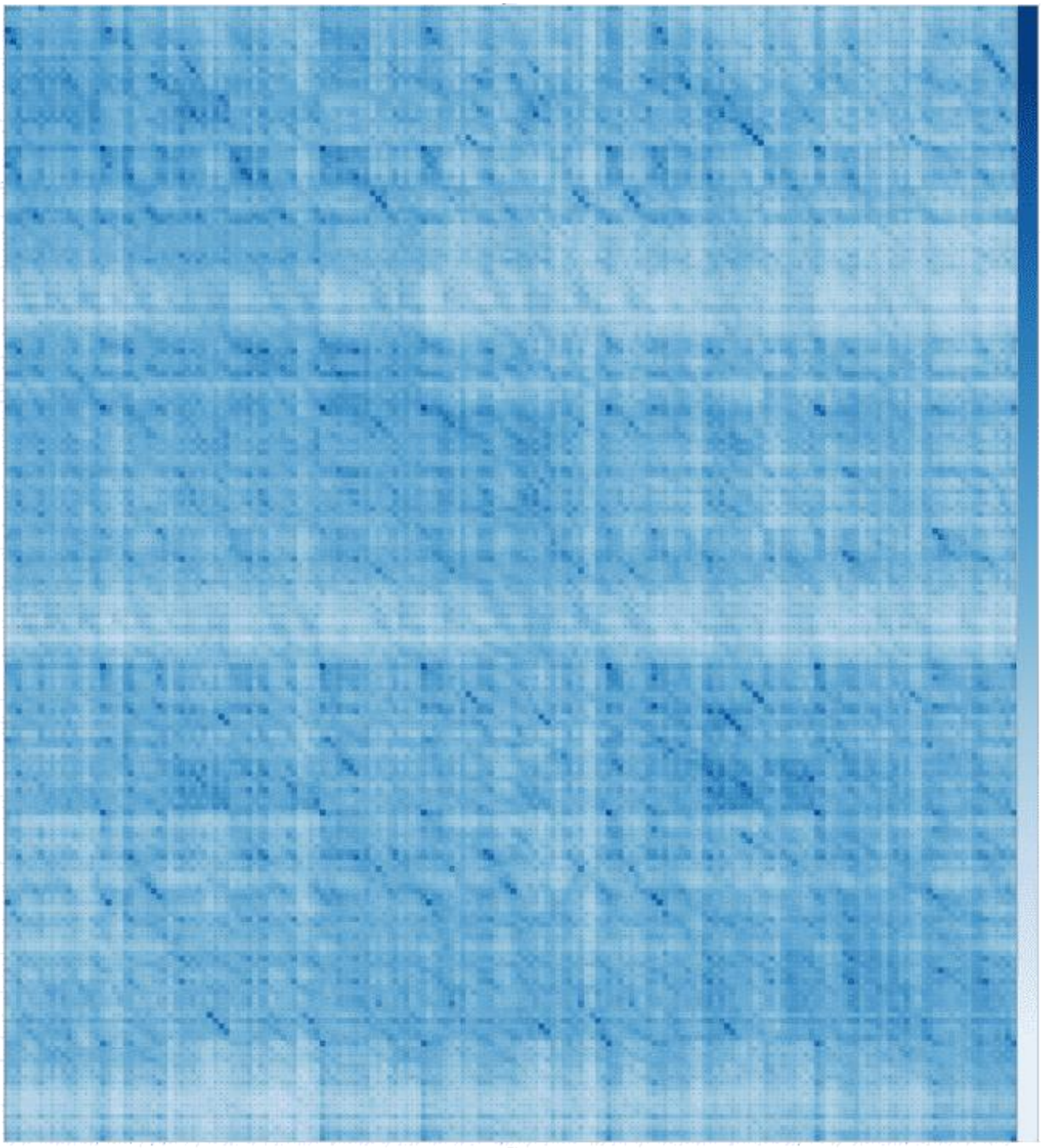
Pred. summary:

```
In [10] we also consider a more general class of algorithms that update subsets of
 variables at each iteration.
```

Kalimat di tokenisasi dalam bentuk *wordpiece* (ditandai dengan ## diawal *wordpiece*-nya). Setiap token pada *Ref. Summary* dicocokkan dengan token pada *Pred. Summary* untuk menghitung *Recall*, dan sebaliknya untuk menghitung *Precision*. *Greedy matching* digunakan untuk memaksimalkan skor similaritas, dimana setiap token dicocokkan dengan token yang paling mirip di kalimat lainnya. Skor *Precision* dan *Recall* dikombinasikan untuk menghasilkan skor F1. Skor F1 ini akan dijadikan skor akhir dari BERTScore untuk mengukur *Pred. Summary*. Visualisasi lengkap *similarity* matriksnya yang menampilkan seluruh kalimat dapat dilihat pada *Lampiran B* dengan skala *similarity* dari 0 (warna terang) sampai dengan 1 (warna gelap).

p-ISSN : 2443-2210
e-ISSN : 2443-2229

LAMPIRAN B



LAMPIRAN C

Reference summary
We consider the estimation problem in Gaussian graphical models with arbitrary structure. We analyze the Embedded Trees algorithm, which solves a sequence of problems on tractable subgraphs thereby leading to the solution of the estimation problem on an intractable graph. Our analysis is based on the recently developed walk-sum interpretation of Gaussian estimation. We show that non-stationary iterations of the Embedded Trees algorithm using any sequence of subgraphs converge in walk-summable models. Based on walk-sum calculations, we develop adaptive methods that optimize the choice of subgraphs used at each iteration with a view to achieving maximum reduction in error. These adaptive procedures provide a significant speedup in convergence over stationary iterative methods, and also appear to converge in a larger class of models.
BESKlus Summary
In [10] we also consider a more general class of algorithms that update subsets of variables at each iteration.This leads us to the following maximum walk-sum tree problem: arg max Sn a tree ¯φ(|h(n−1)|;Sn) Finding the optimal such tree is combinatorially complex.These single-edge walks for edges in Sn are a subset of all the walks in Sn, and consequently provide a lower-bound on ¯φ(|h(n−1)|;Sn).Again, at this iteration, the subgraphs chosen by our methods adapt based on the errors at the various nodes.That is, at each step of the maximum-weight spanning tree algorithm, we only add an edge if it does not create a cycle and maintains a diagonally-dominant tractable subgraph model.Linear Algebra and its Applications, 393:91–105, Dec. 2004.
PyTextRank Summary
Other algorithms may compute walks according to different orders (rather than length-based orders). Based on the absolute convergence condition, walk-summability implies that walk-sums over a countable set of walks in G can be computed in any order. Concatenation of walks We briefly describe the concatenation operation for walks and walk-sets, which plays a key role in walk-sum analysis. Consider the following recursively defined set of walks for s, t ∈ V : Wn(s → t) = (cid:20) = Wn−1(s → ∗) Such edges are useful for capturing walks ending at the high-error nodes, which contribute to the set of walks in (5). Validity: The walks in Wn are valid walks in G, i.e. Wn(s → t) ⊆
BERT Ext. Summarizer Summary
Adaptive Embedded Subgraph Algorithms using Walk-Sum Analysis Venkat Chandrasekaran, Jason K. Johnson, and Alan S. Willsky Department of Electrical Engineering and Computer Science Massachusetts Institute of Technology venkatc@mit.edu, jasonj@mit.edu, willsky@mit.edu Abstract We consider the estimation problem in Gaussian graphical models with arbitrary structure. If the subgraph used changes with each iteration, then we obtain the following non-stationary ET iteration: (6) where {Sn}∞ n=1 is any arbitrary sequence of subgraphs. Given this walk-sum interpretation of the ET algorithm, we can analyze the walk-sets (7) to prove the convergence of ET in walk-summable models by showing that the walk-sets eventually contain all the walks required for the computation of J−1h in (5). kh(n)k2 kh(0)k2 These results show that our adaptive algorithms demonstrate significantly superior convergence properties compared to stationary methods, thus providing a convergent, computationally attrac- tive method for estimation in walk-summable models. Embedded Trees: Estimation of Gaussian processes on graphs with cycles. 5] D. Malioutov, J. Johnson, and A. Willsky.