OPEN ACCESS
Indo-JC
Indonesia Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 4, Issue. 2, Sept 2019. pp. 169-176
doi:10.21108/indojc.2019.4.2.341

Pointer Generator dan *Coverage Weighting* untuk Memperbaiki Peringkasan Abstraktif

Agna Silpi Alpiani [1], Suyanto [2]

School of Computing, Telkom University
Jl. Telekomunikasi Terusan Buah Batu, Bandung, West Java 40257, Indonesia

[1] agnasilpialpiani@student.telkomuniversity.ac.id, [2] suyanto@telkomuniversity.ac.id

Abstract

Long Short-Term Memory (LSTM) sequence-to-sequence model has been widely used to solve some challenges in summarizing text. However, this model has two main problems: the appearance of word outside of vocabulary or often called out-of-vocabulary (OOV) word and the repetition of words. In this paper, a pointer generator and a coverage weighting are proposed to overcome these problems. It starts with a basic sequence-to-sequence model. Then, upgrade with an attention mechanism, which has been added coverage weighting on its calculation to reduce the occurrence of word repetition, and change the encoder to a bi-directional LSTM. After that, the pointer generator is applied in order to point back to the source-word and generate a word if it encounters an OOV word. Using the CNN/Daily Mail news article in English as the dataset and ROUGE score as evaluation metric, the proposed model produces some summaries those are quite similar to the summaries by the experts. It gives some relative improvements from 18 to 34% compared to the standard attention mechanism model. However, it requires a relative higher complexity up to 40%, where it needs more processing time (7 days) than the standard model (only 5 days).

Keywords: abstractive text summarization, coverage weighting, long short-term memory, pointer generator, sequence-to-sequence

Abstrak

Long Short-Term Memory (LSTM) *sequence-to-sequence* telah banyak digunakan untuk menyelesaikan sejumlah tantangan dalam peringkasan teks. Namun, model ini masih memiliki dua masalah utama: kemunculan kata diluar kosakata atau sering disebut *out-of-vocabulary* (OOV) dan perulangan kata. Pada makalah ini, *pointer generator* dan *coverage weighting* diusulkan untuk mengatasi kedua masalah tersebut. Hal ini dimulai dengan model *sequence-to-sequence* dasar. Kemudian, dikembangkan dengan *attention mechanism*, yang telah ditambahkan *coverage weigting* pada perhitungannya untuk mengurangi terjadinya perulangan kata, dan mengganti *encoder* menjadi *bi-directional* LSTM. Setelah itu, *pointer generator* diimplementasikan agar dapat menunjuk kembali ke kata-sumber dan menghasilkan kata jika bertemu dengan kata OOV. Menggunakan artikel berita bahasa Inggris CNN/Daily Mail sebagai *dataset* dan ROUGE *score* sebagai metrik evaluasi, model yang diusulkan menghasilkan sejumlah ringkasan yang lumayan mirip dengan ringkasan yang dibuat para ahli. Model ini memberikan peningkatan relatif dari 18% hingga 34% dari model *standard attention mechanism*. Bagaimanapun, model ini memerlukan kompleksitas relatif lebih tinggi hingga 40%, di mana model usulan perlu lebih banyak waktu proses (selama 7 hari) dibanding model standar (hanya 5 hari).

Kata Kunci: peringkasan teks abstraktif, *coverage weighting*, *long short-term memory, pointer generator, sequence-to-sequence*

Received on August 2019. Accepted on Sept 2019

I. Introduction

TEKNOLOGI peringkasan teks berkembang pesat dalam beberapa tahun terakhir, khususnya sejak ditemukan *deep learning*. Pada tahun 2014, Kågebäck [1] menunjukan bahwa model *neural based continues vector* sangat potensial untuk peringkasan teks. Penemuan ini menandai awal dari meluasnya penggunaan model *neural based* untuk peringkasan teks karena kinerjanya yang unggul dibandingkan dengan teknik tradisional.

Pada dasarnya ada dua metode dalam peringkasan teks, yaitu ekstraktif dan abstraktif [2]. Metode ekstraktif meringkas suatu dokumen dengan memilih sebagian kalimat yang ada dalam dokumen asli. Sedangkan metode abstraktif melakukan peringkasan dengan menginterpretasi teks asal melalui proses transformasi pada kalimat asli tanpa menghilangkan makna sebenarnya. Sejauh ini, pendekatan yang berhasil adalah ekstraktif, dengan mengindentifikasi kata kunci atau frasa dalam teks dan menggabungkannya bersama untuk membentuk sebuah ringkasan. Di sisi lain, peringkasan teks abstraktif umumya lebih disukai pengguna dibanding ekstraktif karena hasil ringkasan hanya berisi sedikit informasi inkoheren sehingga lebih nyaman dibaca.

Dalam pekembangannya, metode peringkasan abstraktif masih memiliki kekurangan, yaitu: kemunculan kata diluar kosakata atau *out-of*-vocabulary (OOV) dan perulangan kata. Akibatnya, kualitas hasil rangkuman tidak mudah dibaca. Untuk meningkatkan performansi metode abstraktif yang sudah ada saat ini, yang berbasis LSTM *sequence-to-sequence* dengan *attention mechanism,* pada penelitian ini diusulkan *pointer generator dan coverage weighting* untuk mengatasi permasalahan tersebut.

Selanjutnya, Bab II akan membahas studi terbaru mengenai peringkasan teks abstraktif. Bab III menjelaskan desain model peringkasan teks abstraktif. Pada Bab IV akan dibahas hasil eksperimen dan analisis hasil model peringkasan teks abstraktif. Terakhir, pada bab V disampaikan kesimpulan dan saran untuk penelitian selanjutnya.

II. Literature Review

Sebelum ditemukan *modern neural network*, peringkasan teks abstraktif tidak mendapatkan banyak perhatian dibandingkan dengan peringkasan teks ekstraktif. Tetapi, Hongyan Jing [3] berhasil mengekplorasi pemotongan bagian kalimat yang tidak penting untuk membuat ringkasan dan Cheung & Penn [4] berhasil mengeksplorasi perpaduan kalimat menggunakan pohon depedensi.

A. Attention mechanism

Rush et al. [5] adalah yang pertama menerapkan *modern neural* network untuk peringkasan teks abstraktif dan berhasil mencapai *state-of-the-art* pada *dataset* DUC 2004 dan *Gigaword,* dataset dengan hasil satu kalimat ringkasan. Pendekatan mereka, *attention mechanism,* telah dikembangkan Chopra et.al [6] dengan *recurrent neural network architecture,* penyederhanaan *framework encoder-decoder* mesin penerjemah oleh Bahdanau et.al [7], dan berhasil melampaui *state-of-the-art* milik Rush et al. dengan *dataset* yang sama.

B. Pointer generator networks

Pointer network [8] adalah sebuah model *sequence-to-sequence* yang memanfaatkan distribusi *soft attention* dari Bahdanau et al. [7] untuk menghasilkan rangkaian output yang terdari dari rangkaian input. *Pointer network* sudah digunakan pada pendekatan *hybrid* untuk *neural machine translation* oleh Gulcehre et al. [9], pemodelan bahasa oleh Merity et al. [10], dan peringkasan teks oleh Gu et al. [11], Miao & Blunsom [12], Nallapati et al [13], dan See et al. [14].

Nallapati et al. [13] memperkenalkan model *pointer generator* yang dapat menentukan apakah akan menggunakan kata dari *vocab* atau menunjuk kembali ke kata tertentu di sumber teks. Namun, modelnya tidak memanfaatkan probabilitas log yang dihasilkan saat menyalin kata dari sumber. See et al. [14] mengatasi masalah ini dengan membuat distribusi tambahan yang menggabungkan probabilitas *vocab* dan probabilitas yang dihasilkan saat menyalin kata dari sumber teks. Model ini diusulkan untuk mengatasi OOV yang muncul dalam *dataset*.

C. Coverage weighting

Pada makalah ini, pendekatan yang diusulkan mirip dengan Tu et al. [15], yang menerapkan *coverage weighting* untuk *neural machine translation*, dan Chen et al. [16], yang juga menerapkan *coverage weighting* yang mereka sebut *"distraction*" untuk pemodelan dokumen.

III. RESEARCH METHOD

A. Dataset

Pada makalah ini digunakan CNN/*Daily Mail Dataset* yang dibuat oleh Hermann et al [17], yang berisi artikel berita berbahasa Inggris dengan dilengkapi ringkasan yang dibuat para ahli (manusia). *Dataset* ini berisi 287.226 artikel dengan rata-rata 29,74 kalimat dan 766 kata, untuk *training* (pelatihan), dan 11.490 artikel dengan rata-rata 3,72 kalimat dan 53 kata, untuk *testing* (pengujian). *Dataset* ini dirilis dalam dua versi. Pada versi pertama, semua entitas nema diganti dengan kode unik, contoh *Telkom University* diganti menjadi @entity2. Sedangkan versi kedua adalah versi orisinal tanpa menggunakan kode unik. Pada makalah ini, digunakan versi orisinal yang lebih sederhana karena tidak memerlukan pra-pemrosesan (*pre-processing*) maupun proses tambahan yang lain pada saat *decoding*.

B. Sequence-to-Sequence Model + Attention Mechanism

Model dasar yang implementasikan mirip dengan Nallapati et al. [13], diilustrasikan pada Gambar 1. Setiap kata w_i dalam artikel dimasukan kedalam *bi-directional* LSTM *encoder* e_i, yang dapat menggabungkan *hidden states* h_i pada setiap tahapan dari kata sebelum dan sesudahnya. Pada setiap tahapan t, *decoder* menerima hasil *word embedding* kata sebelumnya, dan memiliki status s_t.

Perhitungan *attention distribution* sama dengan yang digunakan Bahdanau et al. [7], yaitu menggunakan dua formula

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{attn}) \quad (1)$$

dan

$$a^t = softmax(e^t), \quad (2)$$

di mana bobot matriks W_h, W_s, dan b_{attn} (nilai riil non-negatif) merupakan parameter yang nilainya dipelajari saat pelatihan (*training*). *Attention distribution* dapat diartikan sebagai distribusi probabilitas dari kata di sumber teks, yang memberitahu *decoder* kemana harus menunjuk untuk menghasilkan kata selanjutnya. Selanjutnya, *attention distribution* a^t digunakan untuk menghasilkan penjumlahan *encoder hidden state*, yang dikenal sebagai *context vector* h_t^*, yang dirumuskan sebagai

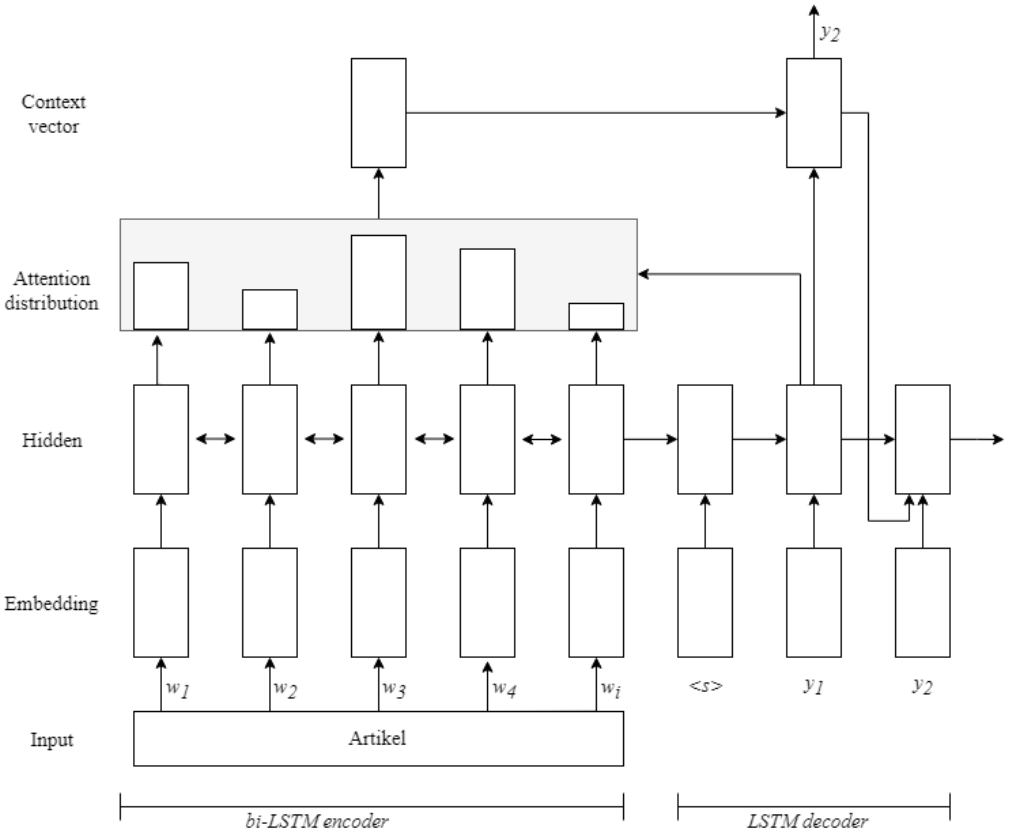
$$h_t^* = \sum_i a_i^t h_i. \quad (3)$$

Context vector h_t^* selanjutnya digabungkan dengan *decoder state* s_t dan dimasukan ke dalam dua *linear layer* untuk menghasilkan distribusi *vocabulary*

$$P_{vocab} = softmax(V'(V[s_t, h_t^*] + \mathrm{b}) + \mathrm{b}'). \quad (4)$$

P_{vocab} adalah distribusi probabilitas seluruh kata yang ada di dalam *vocabulary*, dan menyimpan distribusi final untuk menentukan prediksi kata selanjutnya P_w dengan formula

$$P_w = P_{vocab}(w). \quad (5)$$



Gambar 1. Model *sequence-to-sequence* dengan *attention mechanism*

C. Pointer Generator

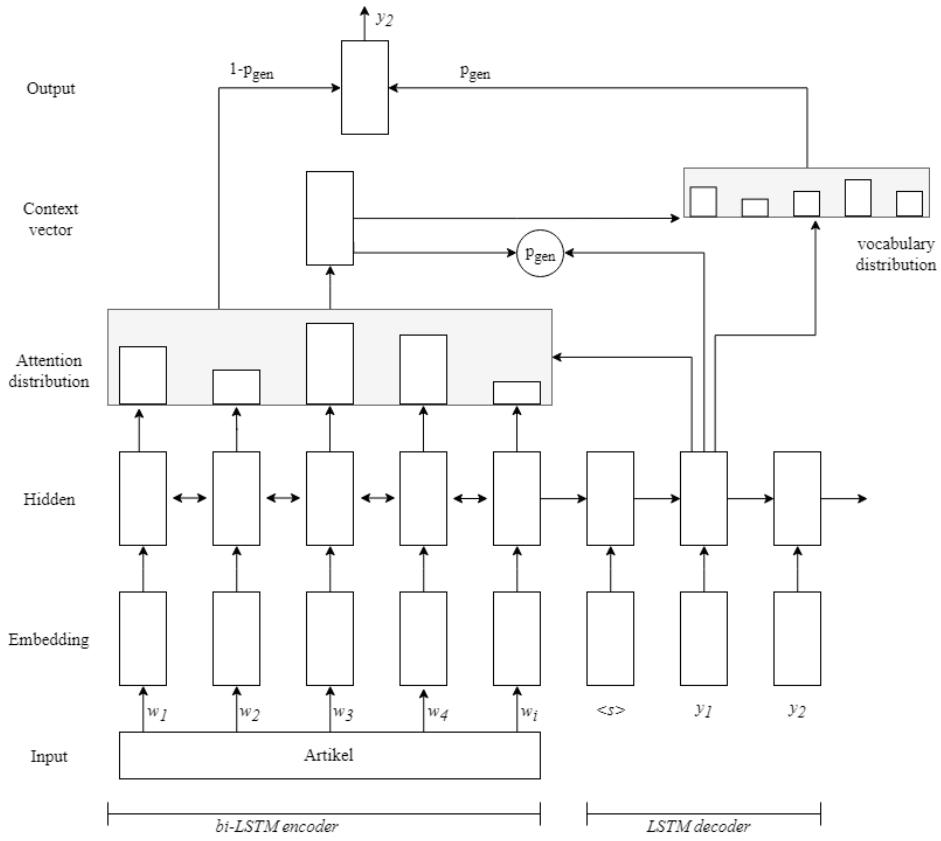
Pada tahap ini akan dilakukan perhitungan *generation probability* p_{gen} dengan memanfaatkan hasil dari *attention distribution* a^t dan *context vector* h_t^* yang dihasilkan pada tahap sebelumnya, Gambar 2. Mengacu pada See et al. [14] dengan memanfaatkan *context vector* h_t^*, status *decoder* s_t, dan *decoder input* x_t akan didapatkan

$$p_{gen} = \sigma\left(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr}\right). \quad (6)$$

Vektor w_h, w_s, w_x dan b_{ptr} (nilai riil non-negatif) merupakan parameter yang nilainya dipelajari saat *training*. Selanjutnya, p_{gen} digunakan untuk menentukan apakah akan menghasilkan kata dari *vocabulary* dengan sampling dari P_{vocab} atau menyalin kata dari *input sequence* dengan sampling dari *attention distribution* a^t berdasarkan probabilitas

$$P(w) = p_{gen} P_{vocab}(w) + \left(1 - p_{gen}\right) \sum_{i:w_i=w} a_i^t. \quad (7)$$

$P(w)$ digunakan untuk menentukan distribusi probabilitas seluruh kata yang ada dalam *vocabulary* dan teks sumber. Model ini memungkinan *decoder* untuk memeriksa *unkown word* untuk digunakan ketika menghasilkan ringkasan.



Gambar 2. Model *pointer generator*

D. Coverage Mechanism

Model yang diusulkan mengadaptasi *coverage* weighting milik Chen et al. [16] untuk mengurangi terjadinya perulangan kata. *Coverage weigting* c^t digunakan sebagai perhitungan tambahan untuk *attention mechanism,* dan mengganti persamaan (1) dengan persamaan

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + w_c c_i^t + b_{attn}). \quad (8)$$

Persamaan (8) dapat memudahkan *attention mechanism* untuk menghindari datang ke lokasi yang sama, dan dengan demikian perulangan teks dapat dihindari.

E. Generating Summaries

Untuk menghasilkan ringkasan, model yang diusulkan menggunakan *sequence-to-sequence* model yang menjadi pendekatan standar untuk *machine translation model*, yaitu dengan formula

$$y^* = \underset{y \epsilon Y}{\text{argmax}} \sum_{i=0}^{N-1} logp(\mathrm{y_i} + 1, x) \, . \quad (9)$$

F. Experiment

Mengacu pada *Google's text summarization,* model yang diusulkan ini menggunakan 128-*dimensional word embedding* dan 256-*dimentional hidden state* untuk sel LSTM. *Word embedding* dilatih secara *scratch* saat *training*. Model yang diusulkan dilatih dengan 50,000 *vocabulary* sebanyak 238,410 iterasi (13 *epochs*). Untuk

optimasi model yang diusulkan, di sini digunakan *Adam: A Method for stochastic optimization* [18] dengan *learning rate* 0,01 dan $b_i = 0.9$, $b_2 = 0.99$. Pada setiap *epoch*, perkembangan *loss* diperhitungkan dan model dengan *loss* minimum yang disimpan. Untuk mencegah *overfitting*, digunakan *dropout rate* sebesar 0,75 di setiap *epoch*.

IV. Results And Discussion

Hasil eksperimen menunjukkan bahwa ringkasan dari model yang hanya menggunakan *attention mechanism* memiliki sejumlah *unknown word* <UNK> dan beberapa perulangan frasa. Sebagai contoh, dari sebuah artikel tentang "*muhhamdu buhari will fight corruption in nigeria*", model ini menghasilkan ringkasan yang memiliki <UNK> pada kalimat pertama dan kedua, dan terjadi perulangan frasa *"<UNK> says his administration"* dan *"nigeria's economy"*. Ringkasan ini juga memiliki kesalahan informasi, di mana frasa *"destabilize nigeria's economy"* adalah hasil parafrase dari *"nigeria's instability"* tetapi konsteknya tidak sesuai dan menghasilkan informasi yang salah. Kata *"nigerians"* berasal dari frasa *"criminals and other contributing to nigeria's instability"* yang diringkas dan mengakibatkan kesalahan semantik. Selain itu, hasil ringkasan pada kalimat ketiga tidak relevan dengan konteks artikel.

Sementara itu, model *pointer generator+coverage weighting* yang diusulkan berhasil mendapatkan kata-kata dengan konsteks yang sesuai, yang sebelumnya tidak didapatkan oleh model *attention mechanism* standar. Dua <UNK> pada kalimat pertama menjadi *"muhammadu buhari"* dan <UNK> pada awal kalimat kedua menjadi *"he"*. Selain itu, model yang diusulkan juga mampu melakukan parafrase *"Buhari told"* menjadi *"Buhari says"* dan kata *"he said"* menjadi *"he says"*.

Bagaimanapun, model yang diusulkan juga memiliki kekurangan. Secara tata bahasa Inggris, hasil ringkasan memiliki kesalahan saat melakukan parafrase. Kata *told* memiliki kesamaan semantik dengan *says*. Tetapi, dalam bahasa Inggris, *says* digunakan untuk kejadian yang sedang terjadi (*present tense*). Selanjutnya, kalimat ketiga yang dihasilkan hanya menyalin kalimat terakhir pada artikel sumber tanpa melakukan parafrase. Hal ini bukan karakteristik peringkasan teks abstraktif, melainkan peringkasan ekstraktif.

Selanjutnya evaluasi hasil ringkasan dilakukan menggunakan ukuran yang disebut *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) *score* [19]. ROUGE *score* adalah sebuah set metode evaluasi yang secara otomatis menentukan kualitas dari hasil sistem peringkasan teks dengan membandingkannya dengan ringkasan yang dibuat manusia. Dalam makalah ini, digunakan nilai F-*measure* dengan $\boldsymbol{b}$ untuk nilai riil non-negatif, *ROUGE*-1, ROUGE-2, dan ROUGE-L sebagai berikut

$$Recall_N = \frac{\sum_r \sum_s match(gram_{refence,proposed})}{\sum_r \sum_s count(gram_{reference,proposed})}, \tag{10}$$

$$Precission_N = \frac{\sum_s \sum_r match(gram_{proposed,reference})}{\sum_s \sum_r count(gram_{proposed,reference})}, \text{dan} \tag{11}$$

$$F\text{-}measure_N = \frac{(1+\boldsymbol{b}^2)R_N P_N}{R_N + \boldsymbol{b}^2 P_N}. \tag{12}$$

ROUGE-1 mengacu pada setiap kata yang sama (*unigram*) antara ringkasan buatan manusia dan ringkasan sistem sedangkan ROUGE-2 mengacu pada setiap dua kata yang sama (*bigram*). ROUGE-L berbasis *Longest Common Subsequence* (LCS) sehingga dapat mengacu pada tingkat struktur kalimat dan mengindentifikasi kesamaan *n*-gram secara otomatis.

Hasil perhitungan ketiga nilai ROUGE *score* diilustrasikan pada Tabel 1. Hasil-hasil ini menunjukkan bahwa model *pointer-generator+coverage weighting* yang diusulkan memberikan nilai yang jauh lebih tinggi untuk ketiga ROUGE *score* dibanding model *attention mechanism* yang standar. Terjadi peningkatan nilai yang sangat signifikan, khususnya untuk ROUGE-2 *score*.

TABEL 1
HASIL EVALUASI ROUGE SCORE

Model	ROUGE *Score*		
	ROUGE-1	ROUGE-2	ROUGE-L
Attention mechanism	31,67	11,61	28,45
Pointer generator + coverage weighting	**37,63**	**15, 66**	**34,21**
Relative improvement	18%	34%	20%

Selanjutnya, dilakukan penghitungan *relative improvement* untuk mengetahui persentase peningkatan relatif antara model yang diusulkan ($x_{proposed}$) dengan model standar ($x_{reference}$), yang dirumuskan sebagai

$$Relative\ improvement_{(x_{proposed},x_{reference})} = \frac{x_{proposed} - x_{reference}}{x_{reference}} 100\%. \quad (13)$$

Dengan rumus tersebut, didapatkan *relative improvement* untuk ROUGE-1 sebesar 18%, ROUGE-2 sebesar 34%, dan ROUGE-L sebesar 20% dibanding model *attention mechanism* standar. Bagaimanapun, model *attention distribution* hanya membutuhkan waktu 5 hari untuk proses *training* sedangkan model *pointer generator+coverage weighting* membutuhkan waktu hingga 7 hari. Artinya, model yang diusulkan memiliki kompleksitas yang relatif lebih tinggi hingga 40% dibanding model standar. Namun, tingginya kompleksitas ini cukup sepadan dengan peningkatan ROUGE *score* yang diperoleh.

V. CONCLUSION

Model peringkasan teks abstraktif menggunakan *pointer generator* dan *coverage weighting* telah sukses diimplementasikan. Pengujian menggunakan CNN/*Daily Mail* yang berbahasa Inggris, dengan metrik evaluasi ROUGE *score*, menunjukkan bahwa secara keseluruhan model yang diusulkan menghasilkan sejumlah ringkasan yang lumayan mirip dengan ringkasan yang dibuat para ahli. Model ini menghasilkan ringkasan yang lebih baik dibanding model standar, dengan *relative improvement* mulai dari 18% hingga hingga 34%. Tetapi, model ini memerlukan kompleksitas yang relatif lebih tinggi hingga 40%, di mana model usulan perlu lebih banyak waktu proses (selama 7 hari) dibanding model standar (hanya 5 hari). Hasil ringkasan juga belum cukup signifikan jika dilihat dari sudut pandang tata bahasa. Hasil ringkasan dari model usulan ini bisa ditingkatkan dengan memperhitungkan informasi lain pada proses *training,* seperti menggabungkan metode ekstraktif dan asbtraktif. Selain itu bisa juga dikembangkan optimasi stokastik saat *training* karena *relative improvement* pada waktu *training* bertambah 40% dari model *attention mechanism* yang standar. Peringkasan teks abstraktif dalam bahasa Indonesia juga sangat potensial untuk dikembangkan, di mana saat ini masih sulit mendapatkan *dataset* bahasa Indonesia yang sudah terverifikasi untuk peringkasan teks.

ACKNOWLEDGMENT

Kami mengucapkan terimakasih kepada orang tua, seluruh kolega di Universitas Telkom, khususnya di Fakultas Informatika, atas segala dukungannya dalam penyelesaian penulisan makalah ini.

REFERENCES

[1] Kågebäck, M., Mogren, O., Tahmasebi, N., & Dubhashi, D. (2014). Extractive summarization using continuous vector space models. In Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC) (pp. 31-39).
[2] Pachantouris, G. (2005). GreekSum–A Greek Text Summarizer. Word Journal of the International Linguistic Association, 1-45.
[3] Jing, H. (2000). Sentence reduction for automatic text summarization. In Sixth Applied Natural Language Processing Conference.
[4] Cheung, J. C. K., & Penn, G. (2014). Unsupervised sentence enhancement for automatic summarization. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 775-786).
[5] Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685.
[6] Chopra, S., Auli, M., & Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 93-98).

[7] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
[8] Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In Advances in Neural Information Processing Systems (pp. 2692-2700).
[9] Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B., & Bengio, Y. (2016). Pointing the unknown words. arXiv preprint arXiv:1603.08148.
[10] Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2016). Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843.
[11] Gu, J., Lu, Z., Li, H., & Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. arXiv preprint arXiv:1603.06393.
[12] Miao, Y., & Blunsom, P. (2016). Language as a latent variable: Discrete generative models for sentence compression. arXiv preprint arXiv:1609.07317.
[13] Nallapati, R., Zhou, B., Gulcehre, C., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv preprint arXiv:1602.06023.
[14] See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368.
[15] Tu, Z., Lu, Z., Liu, Y., Liu, X., & Li, H. (2016). Modeling coverage for neural machine translation. arXiv preprint arXiv:1601.04811.
[16] Chen, Q., Zhu, X. D., Ling, Z. H., Wei, S., & Jiang, H. (2016, July). Distraction-Based Neural Networks for Modeling Document. In IJCAI (pp. 2754-2760).
[17] Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend. In Advances in neural information processing systems (pp. 1693-1701).
[18] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
[19] Lin, C. Y. (2004). Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out.