Klasifikasi Topik dan Analisa Sentimen Terhadap Kuesioner Umpan Balik Universitas Menggunakan Metode *Long Short-Term Memory*

Doenny Auddy Lionovan, Leo Willyanto Santoso, Rolly Intan
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: donny.auddy10@gmail.com, leow@petra.ac.id, rintan@petra.ac.id

ABSTRAK

Dalam melakukan evaluasi dan peningkatan mutu terhadap layanan dan fasilitas, Universitas Kristen Petra (UKP) melakukan pengambilan kuesioner umpan balik yang diberikan pada mahasiswa secara *online*. Pada saat ini, dalam membaca komentar yang ada di kolom saran masih dilakukan secara manual. Sehingga hal tersebut menjadi kurang efektif dan efisien dalam menganalisa sentimen dan mengklasifikasikan topik banyak komentar. Dalam penelitian ini akan menerapkan metode *word2vec* dan *Long Short-Term Memory* untuk membuat sebuah program yang dapat membantu mengklasifikasikan topik dan analisa sentimen.

Word2vec digunakan sebagai metode untuk mengubah suatu kata menjadi vektor beserta dengan pemetaan makna kata yang ada. Dengan parameter yang diuji pada *word2vec* adalah jumlah iterasi dan ukuran *windows*. Sedangkan metode *Long Short-Term Memory* digunakan untuk mengaklasifikasi sentimen dan topik. Dengan parameter yang diuji adalah jumlah *layer*, jumlah *unit*, ukuran *batch*, dan persentase *dropout*.

Hasil dari penelitian ini menunjukan bahwa metode *word2vec* beserta dengan *Long Short-Term Memory* dapat digunakan untuk menganalisa sentimen dan klasifikasi topik. Dengan hasil konfigurasi terbaik didapatkan nilai rata – rata akurasi pada implementasi klasifikasi sentimen sebesar 89,16 % dan untuk implementasi klasifikasi topik sebesar 92,98 %.

Kata Kunci: *Natural Language Processing*, Analisa Sentimen, Klasifikasi teks, *Long Short-Term Memory*, Kuesioner

ABSTRACT

In evaluating and improving the quality of services and facilities, Petra Christian University (PCU) took feedback questionnaires given to students via online. At this time, reading the comments in the suggestion column is still read manually. So that it becomes less effective and efficient in analyzing sentiments and classifying topics for many comments. This study will apply word2vec and Long Short-Term Memory method to create a program that can help classify topics and sentiment analysis.

Word2vec is used as a method to convert a word into a vector along with mapping the meaning of existing words. The parameter tested on word2vec are the number of iterations and windows size. Whereas the Long Short-Term Memory method is used to classify sentiment and topics. The parameter tested are the number of layers, the number of units, the batch size, and the percentage of dropout.

The result of this study indicates that the word2vec method along with Long Short-Term Memory can be used to analyze sentiment and topic classification. The best configuration results obtained average accuracy in the implementation of the sentiment classification is 89,16 % and for implementation of the topic classification is 92,98 %.

Keywords: *Natural Language Processing, Sentiment Analysis, Text Classification, Long Short-Term Memory, Questionnaire*

1. PENDAHULUAN

Dalam melakukan evaluasi dan peningkatan mutu terhadap layanan dan fasilitas yang dimiliki. Universitas Kristen Petra (UKP) sering kali melakukan pengambilan kuesioner umpan balik yang diberikan pada mahasiswa secara *online*. Dari data komentar yang berhasil dikumpulkan akan diolah dan diserahkan untuk segera ditindaklanjuti oleh beberapa lembaga lain yang berkaitan langsung dengan topik yang ada. Saat ini dalam membaca komentar di kolom saran masih dilakukan secara manual yakni membaca saran satu persatu. Selain itu dengan semakin banyak jumlah data yang didapatkan tiap semester. Hal tersebut menjadi kurang efektif dan efisien dalam menganalisa sentimen dan klasifikasi topik pada kolom saran di kuesioner.

Untuk menjawab masalah tersebut, diperlukan program yang dapat digunakan untuk menganalisa sentimen dan klasifikasi topik. Pada penelitian ini akan menggunakan metode *word2vec* untuk mendapatkan nilai vektor pada tiap kata dengan pemetaan makna kata yang serupa. Dan metode *Long Short-Term Memory* (LSTM) untuk melakukan klasifikasi teks baik analisa sentimen atau klasifikasi topik. Metode LSTM dipilih karena dapat mengolah data yang bersifat sekuensial dengan baik. Data untuk penelitian ini berasal dari *survey* dengan total komentar yang berhasil didapatkan sebanyak 1633 komentar. Diharapkan dari penelitian ini dapat membantu Universitas Kristen Petra dalam mengelola data komentar umpan balik dengan lebih cepat dan efisien.

2. DASAR TEORI

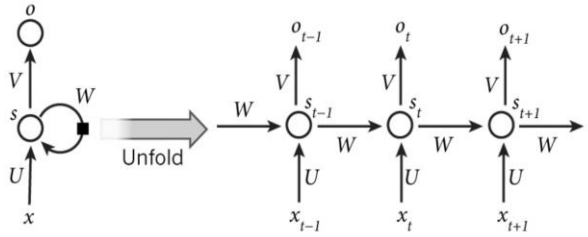
2.1 *Natural Language Processing* (NLP)

NLP merupakan gabungan dari ilmu kecerdasan buatan dan bahasa, digunakan untuk membuat atau memanipulasi komputer agar dapat memahami komentar atau kata – kata yang ditulis dalam bahasa manusia [9]. NLP banyak diterapkan sebagai mesin penerjemah, klasifikasi teks, analisa sentimen, *spam filtering*, peringkasan teks, dan lain sebagainya. Menurut Buntoro (2017) analisa sentimen atau *opinion mining* sendiri merupakan proses

memahami, mengekstrak, dan mengolah data tekstual secara otomatis untuk mendapatkan informasi sentimen yang terkandung dalam suatu kalimat opini [5].

2.2 *Recurrent Neural Network* (RNN)

RNN merupakan salah salah satu metode *neural network* yang didesain untuk dapat memproses data yang memiliki pola sekuensial. Data sekuensial mempunyai karakteristik dimana data yang diproses dengan suatu urutan dan data dalam urutan tersebut mempunyai hubungan erat antara satu dengan yang lainnya [11]. Metode ini memiliki sebuah *hidden state* untuk meneruskan *output* pada setiap tahap yang dilalui. Yang secara berulang *output* dari proses sebelumnya akan kembali menjadi *input* pada proses selanjutnya. Namun, RNN memiliki permasalahan dalam melakukan pengolahan data yakni *vanishing* dan *exploding gradient*. Dimana *vanishing gradient* terjadi saat RNN melakukan *backpropagation* pada data yang memiliki *sequence* yang sangat panjang, nilai *gradient* atau *bobot* yang diupdate akan terus mengecil [4]. Sedangkan *exploding gradient* terjadi ketika perubahan nilai *gradient* menjadi terlalu besar [10]. Bentuk arsitektur RNN dapat dilihat pada Gambar 1.



Gambar 1. Arsitektur RNN [3]

Didalam tiap RNN menerima sebuah *input* yang disimbolkan dengan *x* (x_{t-1}, x_t, dan x_{t+1}). Dan mengeluarkan sebuah *output* dengan simbol *O* (O_{t-1}, O_t, dan o_{t+1}). Dalam setiap proses dari RNN akan menyimpan sebuah hasil dalam *hidden state* yang memiliki simbol *S* (S_{t-1}, S_t, dan S_{t+1}). Persamaan (1) merupakan rumus untuk menghitung *output* dari *hidden state* (S_t).

$$S_t = tanh\ (U \cdot X_t + W \cdot S_{t-1}) \quad (1)$$

Selain *output hidden state* juga terdapat *output* RNN sendiri. Persamaan (2) merupakan rumus dalam menghitung *output* RNN.

$$O_t = softmax\ (V \cdot S_t) \quad (2)$$

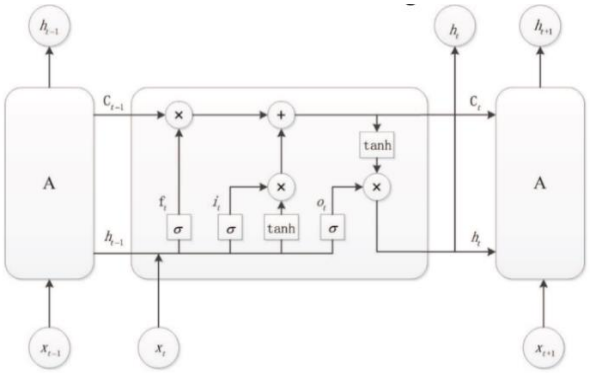
2.3 *Long Short-Term Memory* (LSTM)

LSTM merupakan pengembangan dari model RNN yang juga dapat digunakan untuk mengelola data yang bersifat sekuensial. Pada awalnya dikembangkan oleh Hochreiter & Schmidhuber (1997) untuk mengatasi permasalahan pada model RNN yakni *vanishing gradient* [6].

Arsitektur dari model LSTM memiliki sebuah *cell state* yang dapat menyimpan suatu data dan memberikan ke LSTM berikutnya. *Cell state* pada Gambar 2. terdiri dari *previous cell state* (C_{t-1}) yang menyimpan konteks lama dan *output cell state* (C_t) yang menyimpan konteks baru. Hal tersebut juga digukung dengan tiga *gates*, diantaranya adalah *input gate* (i_t), *forget gate* (f_t), dan *output gate* (O_t) [7]. Di dalam *forget gate* metode LSTM akan mempelajari atau memberikan keputusan suatu konteks di *previous cell state* (C_{t-1}) akan disimpan atau dilupakan. Menggunakan fungsi aktivasi *sigmoid*, *output* dengan nilai mendekati 0 akan dilupakan dan *output* dengan nilai mendekati 1 akan disimpan. Persamaan (3) merupakan persamaan dari *forget gate*.

$$X = \begin{bmatrix} h_t - 1 \\ x_t \end{bmatrix} \quad (3)$$

$$f_t = \sigma\ (W_f \cdot X)$$



Gambar 2. Arsitektur LSTM [1]

Selanjutnya ada *input gate* (i_t) digunakan untuk menentukan informasi baru yang akan disimpan di *cell state* (C_t) menggunakan fungsi aktivasi *sigmoid* dan menghasilkan kandidat konteks baru ($\tilde{C}_t$) menggunakan fungsi aktivasi *tanh*. Persamaan (4) merupakan persamaan *input gate* (i_t) dan kandidat konteks baru ($\tilde{C}_t$).

$$X = \begin{bmatrix} h_t - 1 \\ x_t \end{bmatrix}$$

$$i_t = \sigma\ (W_i \cdot X) \quad (4)$$

$$\tilde{C}_t = \tanh\ (W_c \cdot X)$$

Sebelum masuk ke *output gate*, proses dari LSTM akan melakukan *update* pada *cell state* (C_t) yang ada pada Persamaan (5).

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (5)$$

Proses terakhir masuk pada *output gate* (O_t) yang berfungsi untuk menghasilkan *output hidden state* (h_t). Persamaan (6) merupakan persamaan dalam menghitung hasil dari proses di *output gate*.

$$X = \begin{bmatrix} h_t - 1 \\ x_t \end{bmatrix}$$

$$O_t = \sigma\ (W_o \cdot X) \quad (6)$$

Setelah mendapatkan hasil dari *output gate* (O_t), untuk mendapatkan *output hidden state* (h_t) hasil dari *output gate* dikalikan dengan hasil yang ada di *cell state* yang telah diperbarui (C_t). Persamaan (7) merupakan persamaan untuk menghitung hasil dari *hidden state* (h_t).

$$h_t = O_t \cdot \tanh(C_t) \quad (7)$$

2.4 *Word Embedding*

Word Embedding merupakan salah satu teknik untuk memetakan kata menjadi sebuah vektor agar dapat dipahami oleh metode *neural network*. Dalam hal ini juga dapat memiliki kemampuan untuk menentukan hubungan atau kemiripan makna dari sebuah kata. Oleh karena itu, kata – kata yang bermakna serupa atau sama memiliki representasi nilai vektor yang cenderung dekat. Bentuk hasil dari *word embedding* memiliki dimensi (*v x d*) dimana *v*

merupakan jumlah macam kata (*vocab size*) yang tersimpan di kamus dan *d* merupakan ukuran dimensi pada *word embedding* [8]. Pada penelitian ini akan menggunakan metode *word2vec* dengan model *skip-gram*.

2.5 *Confusion Matrix*

Confusion Matrix merupakan suatu informasi mengenai hasil prediksi dari suatu sistem klasifikasi dengan hasil klasifikasi yang sebenarnya. Isi dari tabel *confusion matrix* ada 4 blok, antara lain:

- *True Positive* (TP) adalah kondisi dimana model data yang terprediksi dengan benar (*True*) dan jawaban aktual juga benar (*True*).
- *True Negative* (TN) adalah kondisi dimana model data yang terprediksi salah (*False*) dan jawaban aktual juga salah (*False*).
- *False Positive* (FP) adalah kondisi dimana model data yang terprediksi benar (*True*) namun jawaban aktual adalah salah (*False*).
- *False Negative* (FN) adalah kondisi dimana model data yang terprediksi salah (*False*) dan jawaban aktual adalah benar (*True*).

Tabel 1. Tabel *confusion matrix* [2]

	Observed True	*Observed False*
Predicted True	*True Positive*	*False Positive*
Predicted False	*False Negative*	*True Negative*

Tabel 1 merupakan ilustrasi *confusion matrix* yang terdiri dari *True Positive, True Negative, False Negative,* dan *False Positive*. Dari tabel *confusion matrix* akan dihitung akurasi, presisi, *f-score*, dan *recall* dari data yang diprediksi. Akurasi merupakan suatu tingkat pengukuran dari rasio prediksi benar (*True*) yang positif dan negatif dengan kesuluruhan data. Akurasi dapat dihitung menggunakan Persamaan (8).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (8)$$

Presisi merupakan jumlah data dengan prediksi positif yang diklasifikasikan dengan benar dibagi dengan keseluruhan hasil yang diprediksi positif. Presisi dapat dihitung menggunakan Persamaan (9).

$$Precision = \frac{TP}{FP + TP} \quad (9)$$

Recall merupakan pengukuran pada data dengan prediksi positif yang benar dibagi dengan keseluruhan data aktual yang positif. *Recall* dapat dihitung menggunakan Persamaan (10).

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

F-Score atau *f-measure* digunakan untuk melakukan perbandingan rata – rata presisi dan *recall*. *F-Score* dapat dihitung menggunakan Persamaan (11).

$$F - score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (11)$$

2.6 *K-Fold Cross Validation*

Cross validation adalah sebuah teknik dalam melakukan pengujian untuk memperoleh perkiraan performa dengan jumlah data terbatas dan mengetahui ketergantungan model terhadap data yang digunakan saat *training* [12]. *Cross validation* dimulai dengan membagi dataset berlabel menjadi partisi sebanyak nilai *k* yang disebut *fold*. Prosesnya adalah dengan melakukan pembagian data sebanyak *k* iterasi. Pada tiap iterasi, *fold* yang berbeda ditetapkan sebagai data *testing* dan sisa data lainnya ditetapkan sebagai data *training*. Sehingga banyak data *training* adalah sebesar (*k-1*) / *k* dan data *testing* sebesar 1/*k*.

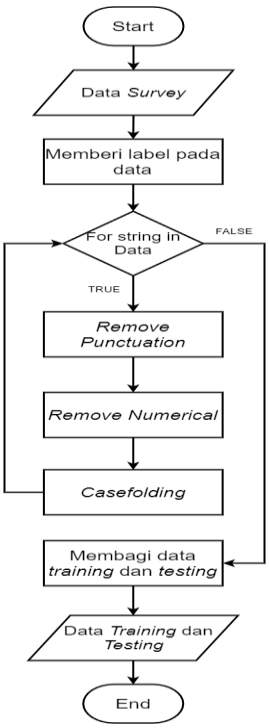
3. DESAIN SISTEM

3.1 Analisis Data

Data yang dikumpulkan berasal data *survey* aspirasi mahasiswa Universitas Kristen Petra yang didapatkan secara *online* pada mahasiswa Universitas Kristen Petra dengan total 1633 komentar. Responden dari *survey* ini merupakan mahasiswa dari beberapa jurusan yang ada di Universitas Kristen Petra. Dari *survey* tersebut data yang digunakan dalam bentuk teks disimpan dalam format CSV (*comma separated values*). Pemberian label sentimen pada penelitian ini berdasarkan dari dua pertanyaan saat *survey*. Antara lain "Mohon berikan komentar positif mengenai layanan dan fasilitas di Universitas Kristen Petra?" dan "Mohon berikan komentar negatif mengenai layanan atau fasilitas di Universitas Kristen Petra?". Sedangkan untuk klasifikasi delapan topik, suatu komentar dapat dikatakan termasuk dalam suatu kategori apabila komentar tersebut memiliki kriteria yang memiliki kata kunci saat digunakan untuk membantu proses pelabelan data.

3.2 Proses *Preprocessing*

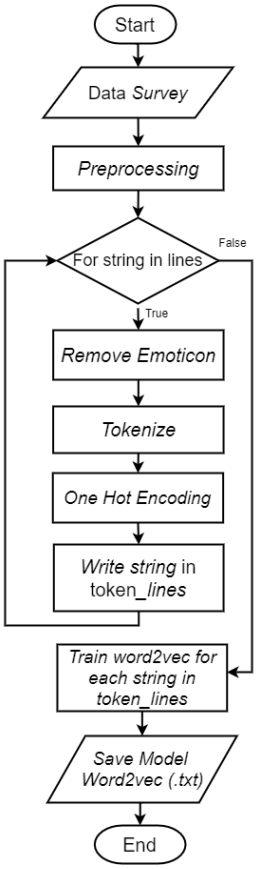
Pada proses *preprocessing* data akan dibersihkan dari tanda baca (*remove punctuation*), menghilangkan angka pada *string* (*remove numerical*), dan mengubah huruf kapital menjadi huruf kecil (*casefolding*). Tahap *preprocessing* juga digunakan untuk memudahkan dalam pembuatan model *word embedding* yang lebih baik. Proses terakhir pada proses *preprocessing* adalah membagi data *training* dan *testing*. Diagram alur proses *preprocessing* dapat dilihat pada Gambar 3.



Gambar 3. Diagram alur *preprocessing*

3.3 Proses *Word Embedding*

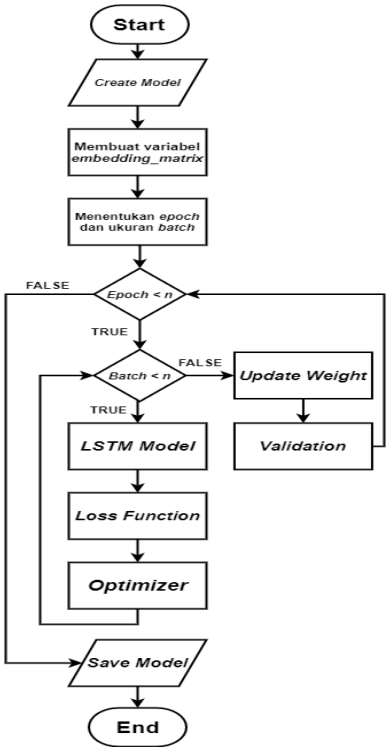
Proses *word embedding* merupakan proses membuat kamus kata untuk merepresentasikan makna kata dengan vektor yang mewakilinya. Proses ini diawali dengan mengambil data *survey* yang telah melalui proses *preprocessing* sebelumnya. Kemudian tiap kalimat atau baris dari data tersebut akan melalui proses *tokenize* untuk memecah suatu kalimat menjadi per kata. Untuk pembersihan *emoticon* juga dilakukan pada proses ini dimana kata yang diproses hanya yang termasuk sebagai alfabet saja. Dari hasil pecahan kata yang ada nanti akan dibuat vektor dalam bentuk *one hot encoding*. Yakni membuat vektor dengan panjang sesuai dengan banyak macam kata. Dimana semua index diberi nilai 0, sedangkan untuk index dari kata tersebut akan diberi nilai 1. Setelah itu akan dilakukan proses *training word2vec* menggunakan *library Gensim*. Hasil akhir dari *training word2vec* akan disimpan dalam format text (.txt). Metode *word2vec* yang digunakan pada penelitian ini adalah *skipgram*. Diagram alur proses *word embedding* dapat dilihat pada Gambar 4.



Gambar 4. Diagram alur proses word embedding

3.4 Proses *Training*

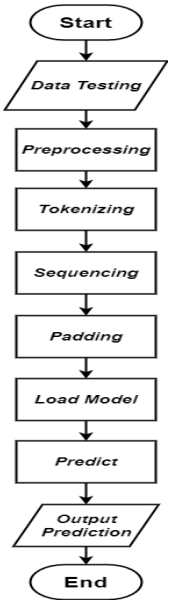
Sebelum masuk dalam proses *training neural network*, perlu terlebih dahulu mengambil bobot dari *word embedding* yang telah dibuat sebelumnya. Proses ini dimulai dengan melakukan proses *tokenizing*, *sequencing*, dan *padding*. Selanjutnya akan disimpan dalam variabel *embedding_matrix* yang menyimpan kata dan nilai vektornya. Setelah membuat variabel *embedding_matrix* yang akan digunakan sebagai bobot awal dalam *embedding layer*. Setelah membuat nilai *embedding_matrix*, akan dilakukan *training* terhadap data yang telah diolah sebelumnya. Diagram alur proses *training* dapat dilihat pada Gambar 5.



Gambar 5. Diagram alur *training*

3.5 Proses *Testing*

Pada proses *testing* diawali dengan mengambil data *testing* sebagai *input* selanjutnya data tersebut dilakukan *preprocessing* seperti menghilangkan tanda baca dan mengubah huruf kapital menjadi huruf kecil. Kemudian dilakukan proses *tokenizing*, *sequencing*, dan *padding* seperti yang dilakukan pada proses *word embedding*. Kemudian perlu untuk *load model* hasil dari proses *training* sebelumnya. Akhir dari proses ini adalah mendapatkan hasil prediksi dari model yang telah melalui proses *training* sebelumnya. Diagram alur proses *testing* dapat dilihat pada Gambar 6.



Gambar 6. Diagram alur *testing*

4. IMPLEMENTASI SISTEM

Implementasi sistem dilakukan pada komputer dengan spesifikasi:

- RAM: 8GB, DDR4
- CPU: Intel Core i7 8750H
- GPU: NVIDIA GeForce GTX 1050
- OS: Windows 10 Home

Implementasi untuk pembuatan program atau aplikasi menggunakan bahasa pemrograman *Python* dengan versi 3.6 64bit. Adapun beberapa *library* yang mendukung sistem ini adalah:

- *Tensorflow*
- *Keras*
- *Gensim*
- *Natural Language Toolkit* (NLTK)
- *Numpy*
- *Pandas*

5. ANALISA DAN PENGUJIAN

5.1 Pengujian *Confusion Matrix*

Pada bagian ini akan dijelaskan mengenai pengujian program yang dilakukan pada data *testing*. Pada pengujian ini dilakukan metode *confusion matrix* sebagai penentu nilai akurasi, *precision*, *recall*, dan *f-score*.

Perincian *confusion matrix* pada klasifikasi sentimen:

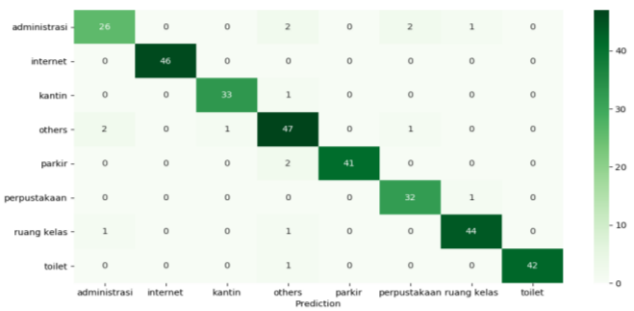
- *True Positive*: 148
- *True Negative:* 143
- *False Positive:* 18
- *False Negative:* 18

Pada pengujian *confusion matrix* menunjukan bahwa dari 166 data sentimen positif, 148 berhasil diprediksi benar dan 18 diantaranya salah prediksi ke sentimen negatif. Sedangkan dari total 161 data negatif, 143 data berhasil diprediksi dengan benar dan 18 diantaranya salah prediksi ke sentimen positif. Untuk hasil *f-score* pada klasifikasi sentimen dapat dilihat pada Tabel 2.

Tabel 2. Hasil *f-score* klasifikasi sentimen

Sentimen	*Precision*	*Recall*	*F1-Score*	Akurasi
Negatif	88,92 %	88,82 %	88,82 %	88,99 %
Positif	89,16 %	89,16 %	89,16 %	88,99 %

Sedangkan untuk perincian *confusion matrix* pada klasifikasi topik dapat dilihat pada Gambar 7.



Gambar 7. *confusion matrix* klasifikasi topik

Berdasarkan dari pengujian *confusion matrix* saat melakukan klasifikasi topik. Pada gambar menunjukan bahwa label "Administrasi" sering salah prediksi ke label lain. Sedangkan banyak komentar yang seharusnya bukan termasuk label "Lain - lain" namun terprediksi ke label "Lain - lain" atau "*Others*".

Tabel 3. Hasil *f-score* klasifikasi topik

Topik	*Precision*	*Recall*	*F1-score*	Akurasi
Administrasi	89,66 %	83,87 %	86,67 %	95,11 %
Internet	100 %	100 %	100 %	95,11 %
Kantin	97,06 %	97,06 %	97,06 %	95,11 %
Lain – lain	87,04 %	92,16 %	89,52 %	95,11 %
Parkir	100 %	95,35 %	97,62 %	95,11 %
Perpustakaan	91,43 %	96,97 %	94,12 %	95,11 %
Ruang Kelas	95,65 %	95,65 %	95,65 %	95,11 %
Toilet	100 %	97,67 %	98,82 %	95,11 %

Pada Tabel 3 menunjukan hasil uji performa *f-score* pada klasifikasi topik. Dari hasil uji performa tersebut menunjukan bahwa komentar dengan label "Administrasi" memiliki persentase *recall* yang paling kecil. Sedangkan untuk nilai *precision* paling rendah ada pada label "Lain - lain".

6. KESIMPULAN

Setelah dilakukan perancangan, implementasi, dan pengujian sistem terhadap program yang telah dibuat, dapat ditarik kesimpulan sebagai berikut:

- Jumlah dan variasi komentar dapat mempengaruhi nilai rata – rata akurasi pada model LSTM.
- Metode *stemming* saat melakukan proses *preprocessing* dapat meningkatkan nilai rata – rata akurasi pada model LSTM.
- *Word2vec* dapat dilakukan untuk implementasi klasifikasi topik dan analisa sentimen bersamaan dengan metode *Long Short-Term Memory* (LSTM). Dengan konfigurasi terbaik menggunakan jumlah iterasi sebanyak 100 kali dan ukuran *windows* sebesar 4. Karena jumlah data teks dan variasi data yang cenderung sedikit diperlukan jumlah iterasi yang sesuai. Karena jika iterasi terlalu sedikit dan juga terlalu banyak hasil rata – rata akurasi menjadi kurang maksimal.
- Konfigurasi metode LSTM untuk implementasi klasifikasi sentimen yang terbaik didapatkan dengan menggunakan 1 *layer*, 16 *unit*, 64 *batch*, 50 % *dropout*, dan 25 *epoch*. Dengan hasil rata – rata akurasi yang terbaik sebesar 89,16 %.
- Konfigurasi metode LSTM untuk implementasi klasifikasi topik yang terbaik didapatkan dengan menggunakan 1 *layer*, 32 *unit*, 64 *batch*, 50 % *dropout*, dan 50 *epoch*. Dengan hasil rata – rata akurasi yang terbaik sebesar 92,98 %.
- Model LSTM untuk analisa sentimen cenderung memiliki nilai *precision* dan *recall* yang lebih rendah pada komentar dengan label negatif daripada komentar dengan label positif. Sedangkan untuk klasifikasi topik cenderung memiliki nilai

precision yang rendah pada komentar dengan label “Lain - lain”.

Saran untuk pengembangan selanjutnya adalah:

- Melakukan penambahan jumlah *dataset* yang lebih jelas dan bervariasi.
- Menggunakan arsitektur RNN atau *neural network* lain yang lebih dalam agar mendapatkan hasil yang lebih maksimal.
- Menggunakan data teks lain dengan jumlah yang lebih besar untuk pembentukan model *word2vec* yang lebih baik.
- Mengintegrasikan penelitian dengan program atau aplikasi Lembaga Penjaminan Mutu (LPM) Universitas Kristen Petra.

7. DAFTAR PUSTAKA

[1] Bai, X. 2018. Text classification based on LSTM and attention. Thirteenth International Conference on Digital Information Management (ICDIM) (pp. 29-32). Berlin: IEEE. doi:10.1109/ICDIM.2018.8847061

[2] Bowes, D., Hall, T., & Gray, D. 2012. Comparing the performance of fault prediction models which report multiple performance measures: recomputing the confusion matrix. Proceedings of the 8th international conference on predictive models in software engineering, (pp. 109-118).

[3] Britz, D. 2015. Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs. Retrieved Maret 31, 2020, from http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[4] Buber, E., & Diri, B. 2019. Web Page Classification Using RNN. 8th International Congress of Information Technology (ICITT). 154, pp. 62-72. Procedia Computer Science. doi:https://doi.org/10.1016/j.procs.2019.06.011

[5] Buntoro, G. A. 2017. Analisis Sentimen Calon Gubernur DKI Jakarta 2017 di Twitter. Integer Journal, 2(1), 32-41.

[6] Hochreiter, S., & Schmidhuber, J. 1997. Long Short Term Memory. In Neural Computation, 9(8). Massachusetts Institute of Technology.

[7] Huang, W., Rao, G., Feng, Z., & Cong, Q. 2018. LSTM with sentence representation for Document-level Sentiment Classification. Neurocomputing, 308, 49-57. doi:https://doi.org/10.1016/j.neucom.2018.04.045

[8] Jozefowicz, R., Zaremba, W., & Sutskever, I. 2015. An Empirical Exploration of Recurrent Neural Network Architecture. International conference on machine learning, (pp. 2342-2350). Lille.

[9] Khurana, D., Koli, A., Khatter, K., & Singh, S. 2017. *Natural Language Processing: State of The Art*, Current Threads, and Challenges. Retrieved from ArXiv: https://arxiv.org/abs/1708.05148

[10] Nicholson, C. 2018. A Beginner's Guide to LSTMs and Recurrent Neural Networks. Retrieved from Pathmind: www.pathmind.com/wiki/lstm

[11] Silvin. 2019. Analisis Sentimen Media Twitter Menggunakan Long Short-Term Memory Recurrent Neural Network. Tangerang: Universitas Media Nusantara.

[12] Wong, T. T., & Yang, N. Y. 2017. Dependency Analysis of Accuracy Estimates in k-fold Cross Validation. IEEE Transactions on Knowledge and Data Engineering, 29(11), 2417-2427.doi:10.1109/TKDE.2017.274092