Jurnal Teknologi 7 (2) 2020 149-160

JURNAL TEKNOLOGI
http://jurnalftijayabaya.ac.id/index.php/JTek
DOI: https://doi.org/10.31479/jtek.v7i2.48

pISSN 1693-0266
eISSN 2654-8666

Implementasi Algoritma *Convolutional Neural Network* Pada Algoritma *K-Means* Untuk Kategorisasi Data Teks

Rakha Paleva Kawiswara [1,*)], Farid Thalib [2)]

[1]Fakultas Teknologi Industri Universitas Gunadarma
[2]Pusat Studi Sistem Sensor dan Teknik Pengukuran Universitas Gunadarma
Jl. Margonda Raya No. 100, Depok 16424, Jawa Barat

*) *Corresponding author*: kawiswara.r@gmail.com

(*Received: 01-May-2020 • Approved: 09 May-2020 • Accepted: 26-May-2020*)

Abstract

The internet developed from the results of the rapid development of computer technology. In reality, internet users generate much data, especially in the form of text, such as posting on social media and articles. For this reason, it is necessary to categorize the data to group text data that has a particular category. Bag of Words is an algorithm that can represent sentences into vectors. However, Bag of Words often produce very high data dimensions or several features, so they require very high computing power. The approach to overcoming this problem is the creation of models that can represent sentences into vectors. Models that use the Convolutional Neural Network (ConvNet) algorithm can be used to study sentences in text data to represent sentences in vector form. Model training and testing using four text data, namely SMS Spam, Bully Comments, Amazon Alexa Reviews, and Large Movie Reviews. The sentence vector results using ConvNet are more efficient in training time and test time compared to representation using Bag of Words. The results of ConvNet sentence vector testing with Fowlkes-Mallow Index measurement for SMS Spam text data are 0.738, for Bully Comment text data is 0.735, for Amazon Alexa Reviews text data is 0.908 and for Large Movie Reviews text data is 0.680.

Abstrak

Internet merupakan keberlanjutan dari pesatnya perkembangan teknologi komputer. Dalam kenyataannya pengguna internet menghasilkan banyak data, khususnya data berupa teks seperti posting pada social media dan artikel-artikel. Untuk itu, diperlukan kategorisasi data untuk mengelompokkan data teks yang memiliki kategori tertentu. *Bag of Words* merupakan algoritma yang dapat merepresentasikan kalimat menjadi vektor. Namun *Bag of Words* seringkali menghasilkan dimensi data atau jumlah fitur yang sangat tinggi, sehingga memerlukan daya komputasi yang sangat tinggi. Pendekatan untuk mengatasi masalah tersebut adalah pembuatan model yang dapat merepresentasikan kalimat menjadi vektor. Model yang menggunakan algoritma *Convolutional Neural Network* (*ConvNet*) dapat dipakai untuk mempelajari kalimat dalam data teks untuk merepresentasikan kalimat dalam bentuk vektor. Pelatihan dan pengujian model menggunakan empat data teks yaitu *SMS Spam*, Komentar Bully, *Amazon Alexa Reviews*, dan *Large Movie Reviews*. Hasil vektor kalimat menggunakan *ConvNet* lebih efisien dalam waktu latih dan waktu uji dibanding dengan representasi menggunakan *Bag of Words*. Hasil pengujian vektor kalimat *ConvNet* dengan pengukuran *Fowlkes-Mallow Index* untuk data teks *SMS Spam* adalah 0.738, untuk data teks komentar Bully adalah 0.735, untuk data teks *Amazon Alexa Reviews* adalah 0.908 dan untuk data teks *Large Movie Reviews* adalah 0.680.

***Keywords** : Categorization, Convolutional Neural Network, K-Means, Sentence Vector, Text Data*

PENDAHULUAN

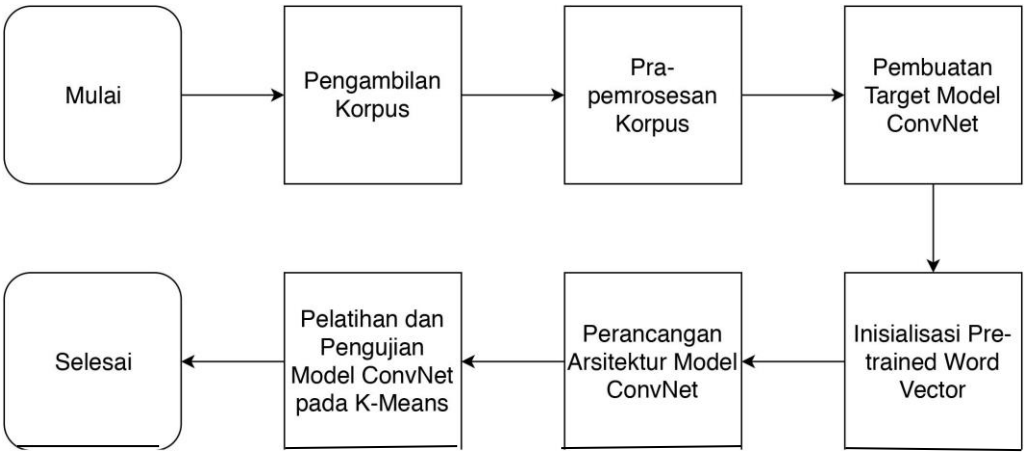
Internet merupakan keberlanjutan dari pesatnya perkembangan teknologi komputer. Dalam kenyataannya pengguna internet menghasilkan banyak data, khususnya data berupa teks seperti *posting* pada sosial media dan artikel-artikel [1]. Untuk itu, diperlukan kategorisasi data untuk mengelompokkan data teks yang memiliki kategori tertentu [1]. *K-Means* merupakan algoritma yang bertujuan untuk kategorisasi data teks menjadi sejumlah kategori yang berhubungan. Untuk kategorisasi data teks menjadi beberapa kategori, setiap kalimat dalam data teks direpresentasikan dalam bentuk vektor. Hal ini dilakukan karena komputer tidak dapat melakukan komputasi pada teks atau *string*.

Salah satu pendekatan untuk merepresentasikan kalimat dalam bentuk vektor adalah dengan pendekatan *Bag of Words*. *Bag of Words* merupakan representasi kalimat yang menggambarkan kemunculan sebuah kata dalam suatu kalimat [2]. Terdapat 2 metode pada *Bag of Words*, metode pertama adalah menghitung kata yang muncul pada kalimat yaitu *Count Vectorizer* dan metode kedua adalah menghitung frekuensi kata yang muncul pada kalimat dikalikan dengan frekuensi dokumen yang terdapat suatu kata yaitu *TF-IDF* [3]. Hasil dari algoritma *Bag of Words* adalah matrikss yang barisnya merupakan kalimat dan kolomnya merupakan kata kunci yang unik yang telah dihitung kemunculannya. Karena prinsip kerjanya yang sederhana, algoritma *Bag of Words* sering digunakan untuk kategorisasi data teks.

Namun terdapat beberapa kelemahan dalam algoritma *Bag of Words* yaitu hasil dari representasi *Bag of Words* menghilangkan urutan suatu kalimat dan menghilangkan makna dari kalimat. Kemudian jika terdapat banyak kata kunci maka ukuran matriks akan sangat besar. Semakin bertambahnya ukuran matriks atau dimensi semakin banyaknya data yang dibutuhkan untuk menggeneralisasi suatu data dengan akurat, hal ini disebut dengan *curse of dimensionality* [4]. Kemudian untuk melakukan proses kategorisasi pada dimensi data yang besar membutuhkan waktu yang lama dikarenakan harus mencari kategori pada setiap data. Oleh karena itu diperlukan hasil representasi vektor kalimat yang memiliki dimensi lebih kecil dan performa yang lebih baik dari hasil representasi *Bag of Words* untuk mengkategorikan data teks menggunakan algoritma *K-Means*.

Penelitian ini akan melakukan implementasi algoritma *Convolutional Neural Network* (*ConvNet*) sebagai algoritma untuk merepresentasikan kalimat dalam bentuk vektor dan *K-Means* sebagai algoritma untuk proses kategorisasi data teks dengan menggunakan data teks yang diambil dari beberapa penelitian yaitu adalah SMS Spam [5], Komentar Bully [6], *Amazon Alexa Reviews* [7] dan *Large Movie Reviews* [8].

METODE PENELITIAN



Gambar 1. Tahapan metodologi penelitian

Metodologi penelitian untuk implementasi algoritma *ConvNet* pada algoritma *K-Means* terdiri dari beberapa tahapan. Pertama tahap pengambilan beberapa data teks yang diambil dari hasil penelitian. Tahapan kedua adalah prapemrosesan data teks untuk membuat data yang sesuai untuk dimasukkan ke dalam model *ConvNet*. Tahap ketiga adalah pembuatan target model *ConvNet*. Tahap keempat adalah inisialisasi pre-trained word vector agar dapat digunakan pada lapisan *embedding* di Model *ConvNet*. Tahap kelima adalah perancangan arsitektur model *ConvNet*. Selanjutnya, tahap terakhir yaitu tahap kelima adalah pelatihan dan pengujian model *ConvNet* pada *K-Means*.

Pengambilan Data

Data teks yang digunakan merupakan hasil penelitian yang telah dilakukan peneliti sebelumnya [5, 6, 7, 8]. Data teks yang digunakan dalam penelitian adalah *SMS Spam* [5], Komentar Bully [6], *Amazon Alexa Reviews* [7] dan *Large Movie Reviews* [8]. Untuk lebih jelasnya, jumlah kalimat dalam data teks dapat dilihat pada Tabel 1.

Tabel 1. Deskripsi jumlah kalimat dalam teks

Data Teks	Jumlah Kalimat
SMS Spam	1143
Komentar Bully	3352
Amazon Alexa Reviews	3071
Large Movie Reviews	25000

Prapemrosesan Data Teks

Prapemrosesan dilakukan untuk mengurangi *noise* pada data teks. Selain itu juga dapat meminimalkan variansi kata yang terdapat pada data teks. Prapemrosesan data teks dibagi dalam beberapa tahapan sebagai berikut:

1. Mengubah karakter dalam kalimat menjadi *lowercase;*
2. Memberikan spasi untuk simbol pada sebelum huruf atau setelah huruf;
3. Menghilangkan spasi yang berlebih dalam kalimat;
4. Melakukan tokenisasi pada kalimat, yang menghasilkan *list* atau vektor yang berisikan token-token kata dan *index* kata;
5. Melakukan *padding* agar setiap vektor memiliki dimensi yang sama. Dimensi vektor ditentukan dengan melihat dimensi vektor dari 90% vektor dalam data teks.
   Setelah prapemrosesan data teks, bentuk data teks menjadi *list* atau vektor yang telah dilakukan *padding*.

Pembuatan Target *ConvNet*

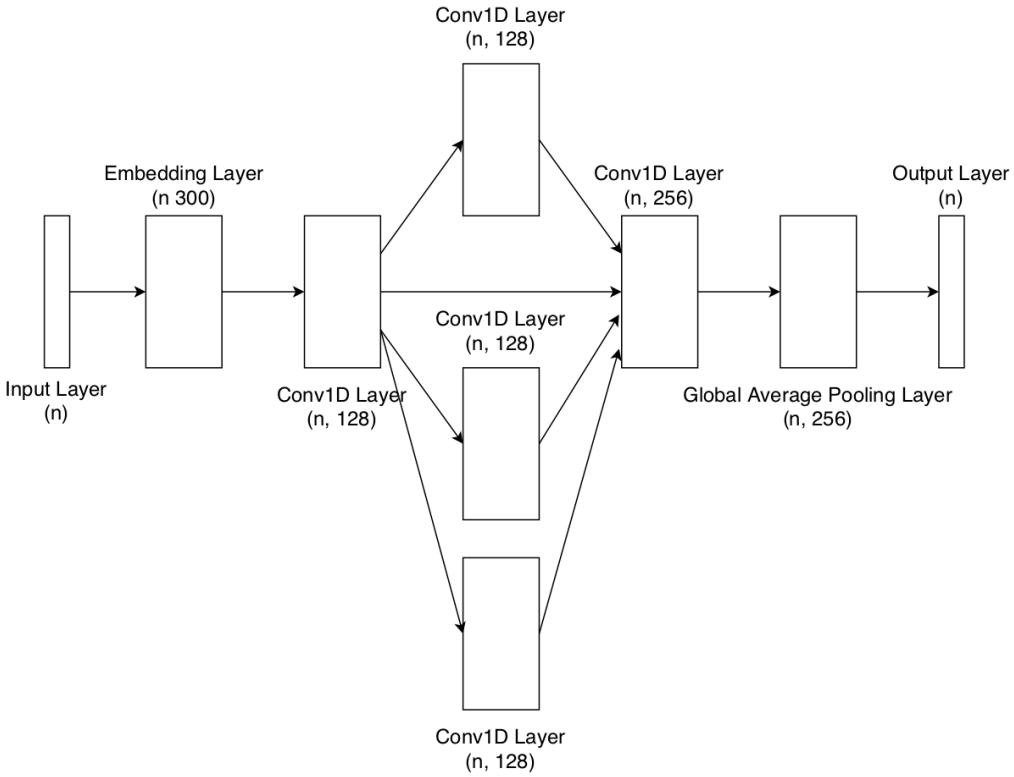
Target dibuat melalui normalisasi nilai pada data teks yang telah dilakukan prapemrosesan. Normalisasi bertujuan untuk mengubah nilai data teks yang semula bernilai dengan skala 0 - ∞ menjadi nilai dengan skala 0 - 1.

Inisialisasi *Pre-trained Word Vector*

Pre-trained word vector menggunakan *pre-trained word vector* dari *Library FastText* [9]. *Pre-trained word vector* yang diambil berdasarkan urutan *index* kata yang didapat dari hasil prapemrosesan. Bentuk dari *pre-trained word vector* merupakan matriks (*n x m*) dengan *n* adalah jumlah *index* kata dan *m* adalah besar *word vector* yang dihasilkan oleh *FastText*. *Pre-trained word vector* digunakan pada lapisan *embedding* di model *ConvNet*.

Perancangan Arsitektur Model *ConvNet*

Tahap ini merupakan perancangan arsitektur model *ConvNet* agar model dapat mempelajari representasi kalimat sedekat mungkin dengan kalimat masukan. Rancangan arsitektur model *ConvNet* dapat dilihat pada Gambar 2.



Gambar 2. Arsitektur Model *ConvNet*

Pada Gambar 2, terdapat 7 lapisan yaitu *Input Layer* sebagai lapisan pertama, *Embedding Layer* sebagai lapisan kedua, *Conv1D Layer* dengan *filter* 128 sebagai lapisan kedua, 3 proses *Conv1D Layer* dengan masing-masing *filter* sejumlah 128 sebagai lapisan ketiga, kemudian *Conv1D Layer* dengan filter 256 sebagai lapisan keempat, *Global Average Pooling Layer* sebagai lapisan kelima dan *Output Layer* sebagai lapisan keenam. Lapisan *input* (masukan) yang menerima kalimat yang sudah ditokenisasi. Setelah lapisan *input*, lapisan *embedding* digunakan untuk pemetaan *pretrained word vector* yang telah diinisialisasi sebelumnya pada setiap nilai pada vektor. Hasil dari lapisan *embedding* berbentuk matriks, dengan baris matriks adalah urutan *token* dan kolom matriksnya adalah *pre-trained word vector*. Selanjutnya dilakukan konvolusi dengan ukuran *kernel* 1 dan jumlah *filter* 128, dengan tujuan untuk mereduksi dimensi matriks dari hasil *embedding* lapisan. Setelah proses reduksi dimensi, dilakukan 3 jenis *konvolusi* dengan ukuran *kernel* 3, 5 dan 7 dengan masing-masing jumlah *filter* 128. Lalu hasil dari 3 jenis *konvolusi* akan digabungkan dengan hasil reduksi dimensi, yang akan menghasilkan *filter* sebesar 512.

Hasil penggabungan konvolusi akan dilakukan reduksi dimensi kembali dengan konvolusi berukuran *kernel* 1 dan jumlah *filter* 256. Aktivasi pada setiap konvolusi menggunakan *ELU* (*Exponential Linear Unit*). Hasil dari reduksi dimensi yang telah dilakukan akan diproses kembali pada lapisan *Global Average Pooling*. *Global Average Pooling* berfungsi untuk menghitung nilai rerata setiap baris matriks pada hasil konvolusi sebelumnya, yang menghasilkan vektor. Hasil *Global Average Pooling* akan diproses pada lapisan terakhir dengan jumlah *neuron* sebesar *input* kalimat dengan menggunakan aktivasi *sigmoid*.

Pelatihan dan Pengujian Model *ConvNet* pada *K-Means*

Pelatihan menggunakan arsitektur model yang telah dirancang. Pengujian model *ConvNet* pada *K-Means* dengan menggunakan hasil representasi kalimat model *ConvNet*. Pelatihan dan Pengujian akan dilakukan dengan metode *K-Fold cross validation* dengan nilai K sebesar 10. Tujuan penggunaan *K-Fold cross validation* adalah untuk melihat reaksi model *ConvNet* dan *K-Means* pada set data yang berbeda-beda. Tahapan metode *K-Fold cross validation* adalah membagi data menjadi beberapa bagian sebanyak nilai K, kemudian melakukan iterasi sebanyak nilai K. Setiap iterasi akan melakukan pengujian pada 1 bagian data dan pelatihan pada 9 bagian data.

Alat Penelitian

Penelitian ini menggunakan *Google Collaboratory* untuk implementasi dan uji coba. Berikut adalah spesifikasi *hardware*-nya,

1. Tesla K80
2. RAM 12GB
3. Intel(R) Xeon(R) CPU @ 2.30GHz

HASIL DAN PEMBAHASAN

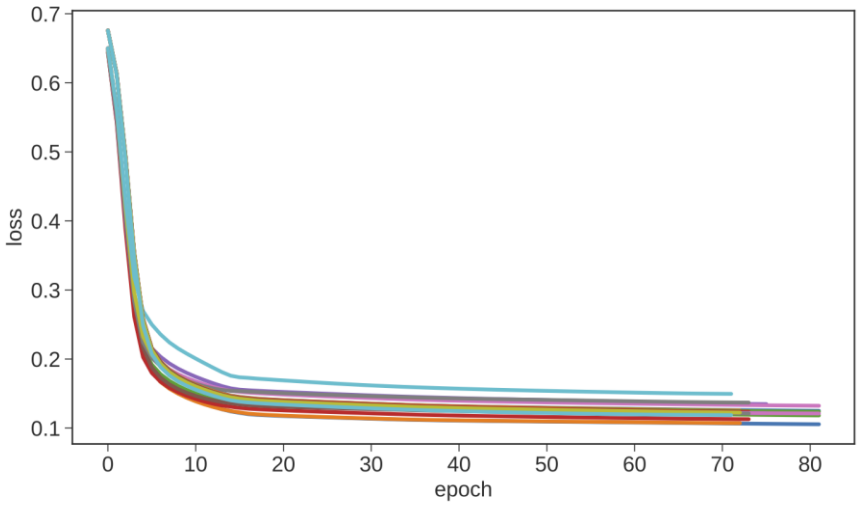
Pelatihan Model *ConvNet*

Model *ConvNet* dilatih dengan menggunakan *Adaptive Momentum Estimation* (*Adam*) sebagai algoritma optimasi, dan menggunakan *Binary Crossentropy* sebagai fungsi *loss*. *Binary Crossentropy* merupakan ukuran model pada *output* yang memiliki probabilitas antara 0 dan 1. Perhitungan *Binary Crossentropy* [10] dapat dinotasikan sebagai berikut,

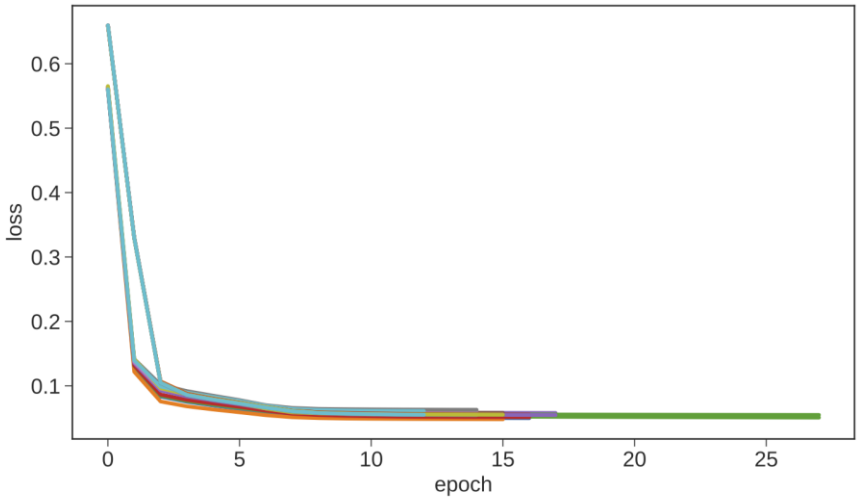
$$L(\hat{y}, y) = -\frac{1}{N}\sum_{i}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \quad (6)$$

dengan:
ŷ, nilai hasil prediksi,
y, nilai label asli,
N, total dokumen pada data teks.

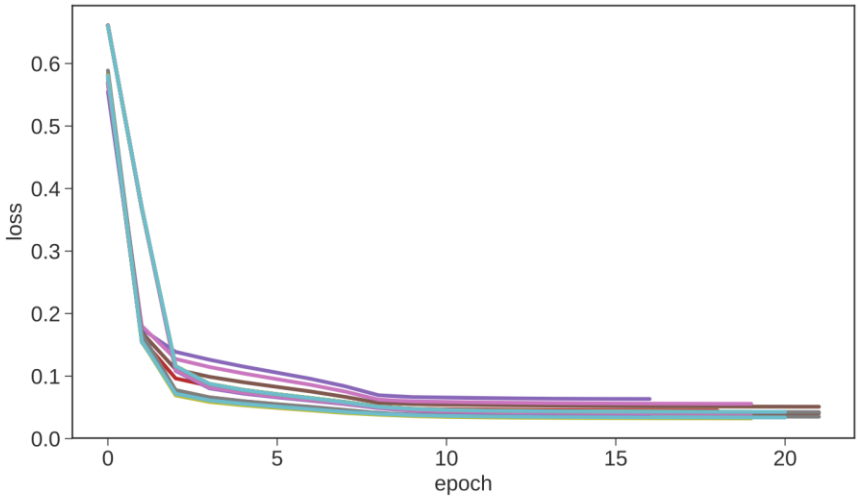
Nilai *Binary Crossentropy* akan besar jika hasil prediksi berbeda dari label asli, dan akan bernilai kecil jika hasil prediksi sama atau hampir dengan label asli. Optimasi pada *Binary Crossentropy* dilakukan untuk mencapai nilai minimum yaitu hasil prediksi sama dengan label asli. Model *ConvNet* dilatih sebanyak 100 kali iterasi dengan besar *batch* sebesar 64, pelatihan akan dihentikan ketika perubahan *loss* hanya sebesar 0.0001. Hasil pelatihan dapat dilihat pada Gambar 3, 4, 5, 6.



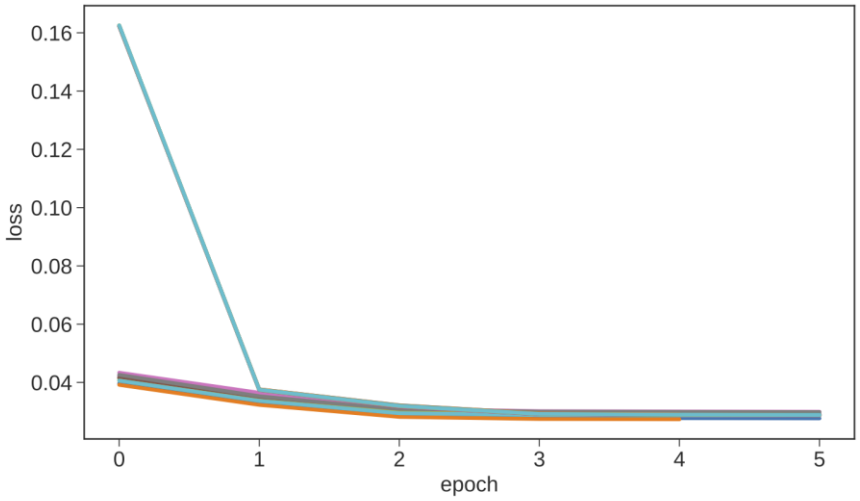
Gambar 3. Grafik nilai *loss* setiap *fold* model pada data teks SMS Spam



Gambar 4. Grafik nilai *loss* setiap *fold* model pada data teks Komentar Bully



Gambar 5. Grafik nilai *loss* setiap *fold* model pada data teks *Amazon Alexa Reviews*

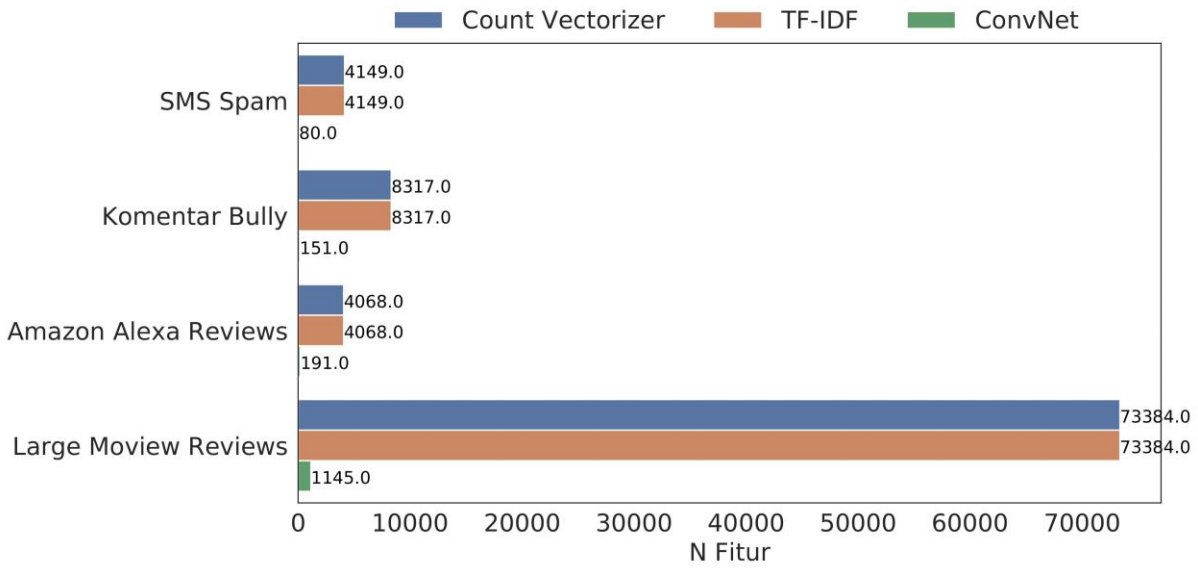


Gambar 6. Grafik nilai *loss* setiap *fold* model pada data teks *Large Movie Reviews*

Pada Gambar 3, 4, 5 dan 6, dapat dilihat hasil pelatihan pada setiap *fold* suatu data tetap memiliki nilai *loss* yang cenderung sama nilainya. Kemudian nilai *loss* paling kecil terdapat pada data *Large Movie Reviews.* Dapat dilihat juga pada Gambar 3, 4, 5 dan 6, terdapat perbedaan iterasi pada proses pelatihan. Model *ConvNet* paling lama dalam melakukan proses pelatiham data teks *SMS Spam* dan paling cepat dalam melakukan proses pelatihan pada data teks *Large Movie Reviews.*

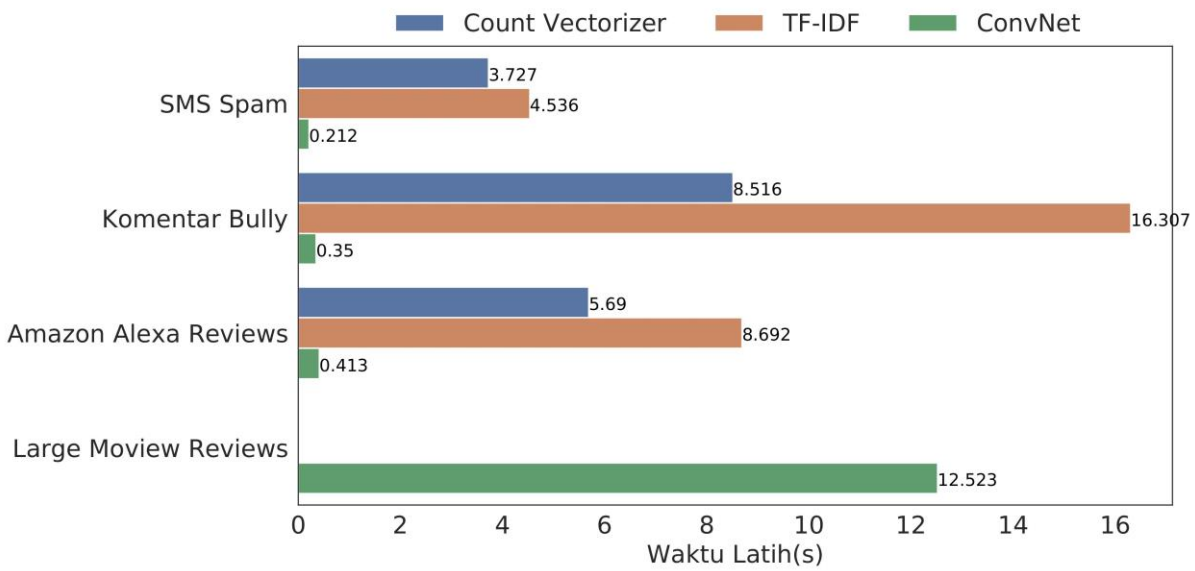
Pengujian Vektor Kalimat *ConvNet* pada *K-Means*

Hasil representasi kalimat dari model *ConvNet* diuji dengan mengukur kinerja *K-Means* dalam pengategorian data teks. Dalam pengujian juga dibandingkan kinerja dengan representasi dua metode *Bag of Words* yaitu *Count Vectorizer* dan *TF-IDF* pada *K-Means*. Ukuran yang digunakan dalam pengujian adalah *Fowlkes-Mallow Index* (*FMI*), yang ditampilkan pada Gambar 7, 8, 9, 10.



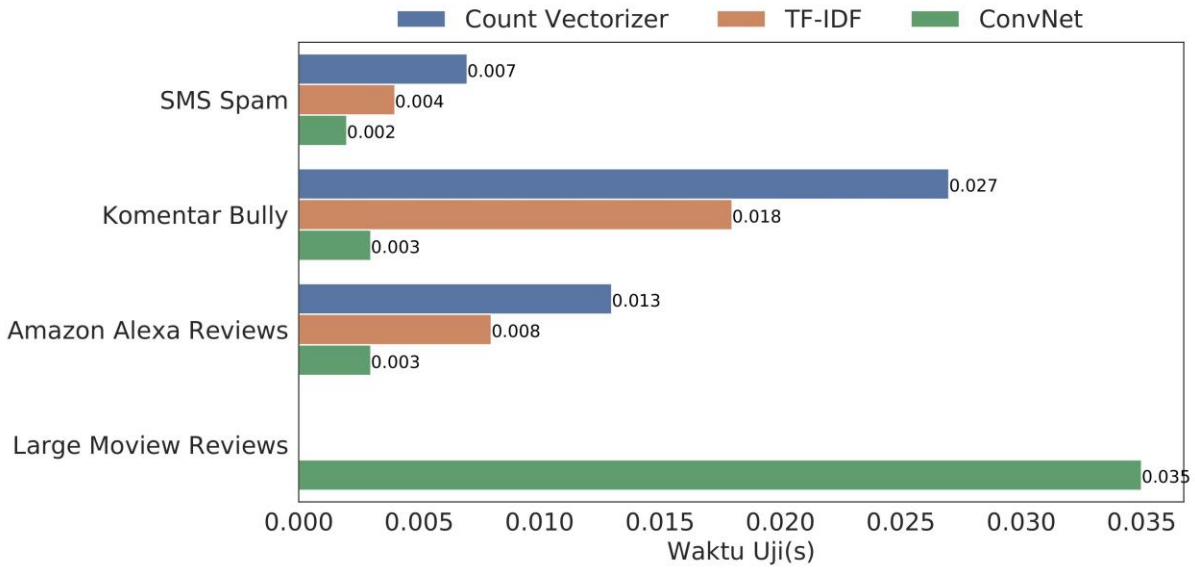
Gambar 7. Grafik rerata hasil pengujian setiap data teks pada variabel N Fitur

Jumlah fitur atau besarnya dimensi vektor dari model *ConvNet* memiliki dimensi paling kecil dibanding *Count Vectorizer* dan *TF-IDF*. Karena *Count Vectorizer* dan *TF-IDF* membuat fitur untuk setiap kata yang terdapat pada data teks, sedangkan model *ConvNet* jumlah fitur atau besar dimensi tergantung dari rata-rata panjang kalimat pada suatu data teks.



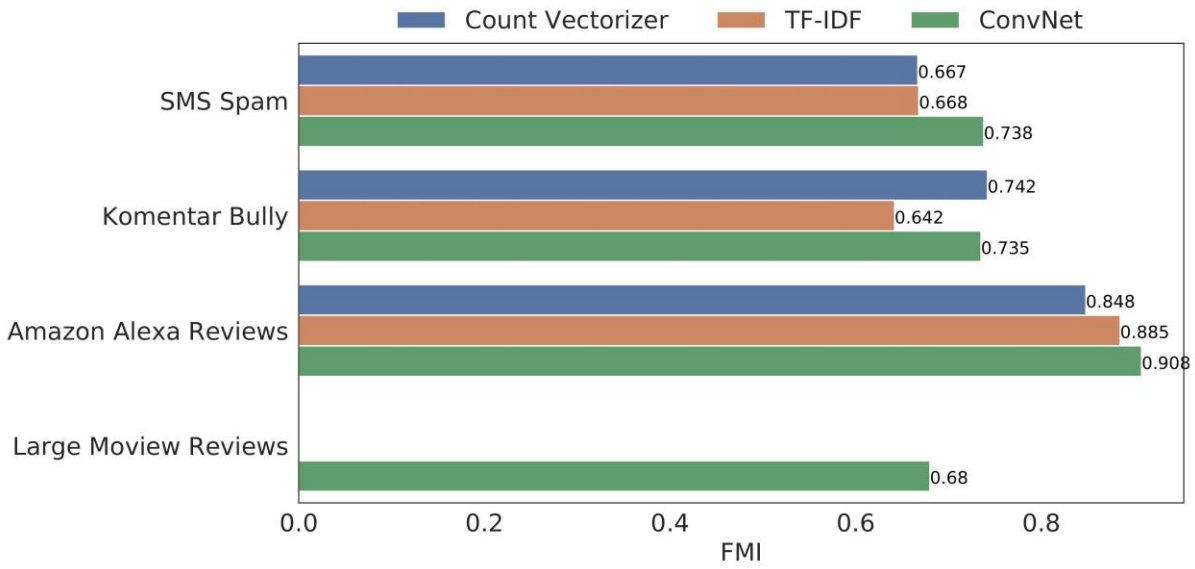
Gambar 8. Grafik rerata hasil pengujian setiap data teks pada variabel Waktu Latih(s)

Hasil vektor kalimat dari model *Convnet* paling cepat diproses pelatihan nya oleh Algoritma *K-Means* karena dimensi vektor yang dihasilkan model *Convnet* lebih kecil dibanding *Count Vectorizer* dan *TF-IDF*. Terdapat kegagalan pelatihan pada data teks *Large Movie Reviews* untuk output *Count Vectorizer* dan *TF-IDF* karena besarnya dimensi yaitu 73384.



Gambar 9. Grafik rerata hasil pengujian setiap data teks pada variabel Waktu Uji(s)

Waktu uji untuk hasil vektor kalimat dari model *Convnet* juga lebih cepat dibanding hasil vektor kalimat dari *Count Vectorizer* dan *TF-IDF* dan terdapat kegagalan dalam pengujian pada data teks *Large Movie Reviews* untuk hasil vektor kalimat *Count Vectorizer* dan *TF-IDF*.



Gambar 10. Grafik rerata hasil pengujian setiap data teks pada variabel FMI

Pada data teks SMS Spam, hasil vektor kalimat dari model *ConvNet* lebih baik. Kemudian pada data teks Komentar Bully, hasil vektor kalimat dari *Count Vectorizer* dan *ConvNet* terdapat perbedaan yang tidak terlalu signifikan. Pada data teks *Amazon Alexa Reviews*, hasil vektor kalimat dari *TF-IDF* dan model *Convnet* memiliki nilai yang sama.

KESIMPULAN

Model *ConvNet* perlu dilatih sebelum merepresentasikan sebuah kalimat dalam bentuk vektor. Hasil vektor kalimat menggunakan model *Convnet* memiliki jumlah fitur atau dimensi yang lebih kecil dibandingkan hasil vektor kalimat *Count Vectorizer* atau *TF-IDF*. Hasil vektor kalimat menggunakan *ConvNet* lebih efisien dalam waktu latih dan waktu uji dibanding dengan vektor kalimat menggunakan *Bag of Words* dikarenakan jumlah fitur dari hasil vektor kalimat dari model *ConvNet* sama dengan jumlah fitur dari hasil tokenisasi. Hasil dari model *ConvNet* juga lebih baik daripada *Bag of Words* karena pola dalam setiap data teks diambil dari hasil pembelajaran model *ConvNet*.

DAFTAR PUSTAKA

[1] Huang, Anna. "Similarity measures for text document clustering." In Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand, vol. 4, pp. 9-56. 2008.

[2] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in Advances in neural information processing systems, pp. 649–657, 2015.

[3] S. C. Eshan and M. S. Hasan, “An application of machine learning to detect abusive bengali text,” in 2017 20th International Conference of Computer and Information Technology (ICCIT), pp. 1–6, IEEE, 2017.

[4] Verleysen, Michel, and Damien François. "The curse of dimensionality in data mining and time series prediction." In International Work-Conference on Artificial Neural Networks, pp. 758-770. Springer, Berlin, Heidelberg, 2005.

[5] F. Rahmi, APLIKASI SMS SPAM FILTERING PADA ANDROID MENGGUNAKAN ALGORITMA NAÏVE BAYES. PhD thesis, Universitas Pendidikan Indonesia, 2016.

[6] H. M. Saputro, “Klasifikasi komentar bully dengan implementasi algoritma random forest,” 2019.

[7] “Amazon alexa reviews corpus.” (https://www.kaggle.com/sid321axn/ amazon-alexa-reviews. [Online; Diakses pada 17-April-2019].

[8] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, (Portland, Oregon, USA), pp. 142–150, Association for Computational Linguistics, June 2011.

[9] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, “Learning word vectors for 157 languages,” in Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[10] Keren, Gil, S.Sabato, and B. Schuller. "Fast single-class classification and the principle of logit separation." In 2018 IEEE International Conference on Data Mining (ICDM), pp. 227-236. IEEE, 2018.