Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019
masa berlaku mulai Vol. 1 No. 1 tahun 2017 s.d Vol. 5 No. 3 tahun 2021

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 4 No. 2 (2020) 276 - 285 ISSN Media Elektronik: 2580-0760

Klasifikasi Teks Multilabel pada Artikel Berita Menggunakan Long Short-Term Memory dengan Word2Vec

Winda Kurnia Sari[1], Dian Palupi Rini[2*], Reza Firsandaya Malik[3], Iman Saladin B. Azhar[4]
[1] Master of Computer Science, Universitas Sriwijaya, Palembang 30128, Indonesia
[2] Informatics Departement, Universitas Sriwijaya, Palembang 30128, Indonesia
[3] Communication Network and Information Security Research Lab, Palembang 30128, Indonesia
[4] Information System Departement, Universitas Sriwijaya, Palembang 30128, Indonesia
[1]windakurniasari@unsri.ac.id, [2]dprini@unsri.ac.id, [3]rezafm@unsri.ac.id, [4]imansaladin@ilkom.unsri.ac.id

Abstract

Multilabel text classification is a task of categorizing text into one or more categories. Like other machine learning, multilabel classification performance is limited to the small labeled data and leads to the difficulty of capturing semantic relationships. It requires a multilabel text classification technique that can group four labels from news articles. Deep Learning is a proposed method for solving problems in multilabel text classification techniques. Some of the deep learning methods used for text classification include Convolutional Neural Networks, Autoencoders, Deep Belief Networks, and Recurrent Neural Networks (RNN). RNN is one of the most popular architectures used in natural language processing (NLP) because the recurrent structure is appropriate for processing variable-length text. One of the deep learning methods proposed in this study is RNN with the application of the Long Short-Term Memory (LSTM) architecture. The models are trained based on trial and error experiments using LSTM and 300-dimensional words embedding features with Word2Vec. By tuning the parameters and comparing the eight proposed Long Short-Term Memory (LSTM) models with a large-scale dataset, to show that LSTM with features Word2Vec can achieve good performance in text classification. The results show that text classification using LSTM with Word2Vec obtain the highest accuracy is in the fifth model with 95.38, the average of precision, recall, and F1-score is 95. Also, LSTM with the Word2Vec feature gets graphic results that are close to good-fit on seventh and eighth models.

Keywords: recurrent neural network, long short-term memory, multilabel classification, Word2Vec

Abstrak

Klasifikasi teks multilabel adalah tugas mengategorikan teks ke dalam satu atau lebih kategori. Seperti pembelajaran mesin lainnya, kinerja klasifikasi multilabel terbatas ketika ada data kecil berlabel dan mengarah pada kesulitan menangkap hubungan semantik. Dibutuhkan teknik klasifikasi teks multilabel yang dapat mengelompokkan empat label dari artikel berita untuk penelitian ini. *Deep Learning* adalah metode yang diusulkan untuk memecahkan masalah dalam klasifikasi teks multilabel. Beberapa contoh metode deep learning yang digunakan untuk pengklasifikasian teks antara lain *Convolutional Neural Networks*, *Autoencoder*, *Deep Belief Networks*, dan *Recurrent Neural Networks* (RNN). RNN merupakan salah satu arsitektur yang paling popular yang digunakan dalam Pemrosesan Bahasa Alami (PBA) karena struktur *recurrent* cocok untuk proses teks bervariabel panjang. Salah satu metode *deep learning* yang diusulkan pada penelitian ini adalah RNN dengan penerapan arsitektur Long Short-Term Memory (LSTM). Dalam penelitian ini untuk mendapatkan model yang optimal pada klasifikasi teks dilakukan percobaan *trial* dan *error* menggunakan LSTM dengan fitur *word embedding* Word2Vec 300 dimensi. Dengan *tuning hyperparameter* dan membuat perbandingan delapan model LSTM yang diusulkan dengan dataset skala besar, dan untuk menunjukkan bahwa LSTM dengan fitur Word2Vec dapat mencapai kinerja yang baik dalam klasifikasi teks. Hasil penelitian menunjukkan bahwa klasifikasi teks menggunakan LSTM dengan fitur Word2Vec memperoleh akurasi tertinggi pada model kelima dengan 95,38%, sedangkan rata-rata nilai presisi, recall, dan F1-score adalah 95%. Selain itu, LSTM dengan fitur Word2Vec mendapatkan hasil grafik yang dekat dengan good-fit untuk model ketujuh dan kedelapan.

Kata kunci: recurrent neural network, long short-term memory, multilabel classification, Word2Vec.

Diterima Redaksi : 29-01-2020 | Selesai Revisi : 29-02-2020 | Diterbitkan Online : 20-04-2020

1. Pendahuluan

Klasifikasi teks termasuk bagian penting dalam Pemrosesan Bahasa Alami dengan banyak penerapan seperti sentimen analisis, pencarian informasi, perankingan, *indexing* dan klasifikasi dokumen [1][2][3]. Model klasifikasi teks umumnya dibagi menjadi dua kategori: machine learning dan deep learning. Banyak penelitian pada klasifikasi teks telah melibatkan algoritma tradisional machine learning seperti Support Vector Machine, Naive Bayes, k-Nearest Neighbors, Logistic Regression [4][5]. Selain itu, dibandingkan dengan algoritma klasifikasi tradisional machine learning memiliki karakteristik efisiensi dan stabilitas tinggi. Namun, memiliki batasan tertentu dalam hal pelatihan dataset skala besar [6].

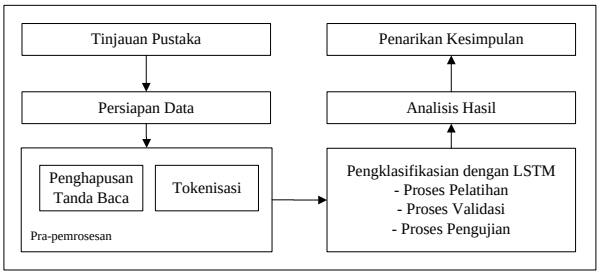
Baru-baru ini, model berbasis jaringan saraf menjadi semakin populer. Model-model ini mencapai kinerja yang sangat baik dalam praktiknya, cenderung relatif lambat baik pada saat pelatihan maupun pengujian, membatasi penggunaannya pada kumpulan data yang sangat besar [7]. Beberapa penelitian terbaru menunjukkan bahwa keberhasilan *deep learning* tentang klasifikasi teks sangat tergantung pada efektivitas *word embedding* [8]. Khususnya, Shen dkk, 2018 secara kuantitatif menunjukkan bahwa tugas klasifikasi teks berbasis *word embedding* dapat memiliki tingkat kesulitan yang sama terlepas dari model yang digunakan, menggunakan konsep dimensi intrinsik. Bagaimanapun, model sederhana lebih dipilih. Sebagai blok bangunan dasar dalam PBA berbasis jaringan syaraf, *word embedding* menangkap kesamaan antara kata-kata [9]. Gagasan ini telah diperluas untuk menghitung *embedding* yang menangkap semantik dari urutan kata seperti Frasa, kalimat, paragraf, dan dokumen.

Salah satu metode *deep learning* yang diusulkan pada penelitian ini adalah RNN dengan penerapan arsitektur *Long Short-Term Memory* (LSTM). RNN dapat menggunakan representasi kata yang terdistribusi dengan terlebih dahulu mengubah token yang terdiri dari setiap teks menjadi vektor, yang membentuk matriks. Sedangkan, LSTM dikembangkan untuk menangani masalah *exploding* dan *vanishing gradient* yang dapat dihadapi saat melatih RNN tradisional [10]. Penelitian Zhang dll, 2015 menerapkan jaringan convolusional (ConvNets) hanya pada karatkter menggunakan *word embedding* bag-of-words, n-grams dan varian TFIDF [16]. Penelitian Conneau dkk, 2017 menggunakan *Deep* CNN dengan 29 lapisan untuk klasifikasi teks [11]. Pada penelitian Wang dkk, 2018 menerapkan penggabungan *embedding* kata-kata dan label dengan mengusulkan *Label Embedding Attentive Model* (LEAM) untuk meningkatkan pengklasifikasian teks [12]. Sedangkan pada penelitian ini menggunakan LSTM dengan *word embedding* dari *deep learning* yaitu Word2vec dengan membuat percobaan delapan model *tuning* LSTM untuk mendapatkan model yang optimal pada pengklasifikasian teks.

2. Metode Penelitian

2.1. Metodologi

Secara umum, langkah-langkah dalam metodologi penelitian yang digunakan untuk membantu dalam persiapan penelitian ini membutuhkan kerangka kerja yang jelas untuk tahapan-tahapannya. Kerangka penelitian yang digunakan seperti pada Gambar.1 yang terdiri dari tinjauan pustaka dengan meninjau penelitian dalam 5 tahun dan 1 tahun terakhir, dilanjutkan dengan persiapan data yang menggunakan dataset AGNews dengan 400.000 lebih sampel data teks, kemudian dilakukan pra-pemrosesan dengan menghapus tanda baca dan melakukan tokenisasi, proses pengklasifikasian dengan salah satu arsitektur RNN yaitu LSTM dimana proses pengklasifikasian dengan LSTM terdiri dari 3 sub proses, yaitu proses pelatihan, validasi, dan pengujian, terakhir menganalisis hasil, dan menarik kesimpulan.



Gambar 1. Metodologi Penelitian

2.2. Ekstraksi Fitur

Ekstraksi fitur adalah bagian penting dari *machine learning* terutama untuk data teks. Dataset teks adalah data yang paling tidak terstruktur yang diperlukan untuk menghasilkan makna dan struktur yang digunakan oleh algoritma *machine learning*. Baru-baru ini, T. Mikolov memperkenalkan teknik yang lebih baik untuk mengekstraksi fitur dari teks menggunakan konsep *embedding*, atau menempatkan kata ke dalam ruang vektor berdasarkan konteks. Pendekatan ini untuk embedding kata, disebut Word2Vec, memecahkan masalah mewakili hubungan kata kontekstual dalam ruang fitur yang dapat dihitung. Pada penelitian ini menggunakan 300 dimensi *embedding* Word2vec untuk menjadi fitur masukan dalam LSTM.

2.3. *Recurrent Neural Network*

RNN adalah jenis jaringan saraf dengan status memori untuk memproses masukan urutan. RNN tradisional memiliki masalah yang disebut *gradient vanishing* dan *exploding* selama pelatihan. RNN masuk dalam kategori *deep learning* karena data diproses secara otomatis dan tanpa pendefinisian fitur. RNN dapat menggunakan status internal (memori) untuk memproses urutan masukan. Ini membuatnya dapat diterapkan untuk tugas-tugas seperti pemrosesan bahasa

alami (PBA), pengenalan suara, sintesa musik, pemrosesan data finansial seri waktu [13][14].

Persamaan Dasar RNN:

$$s_t = \tanh(U_{x_t} + W_{s_{t-1}}) \quad (1)$$

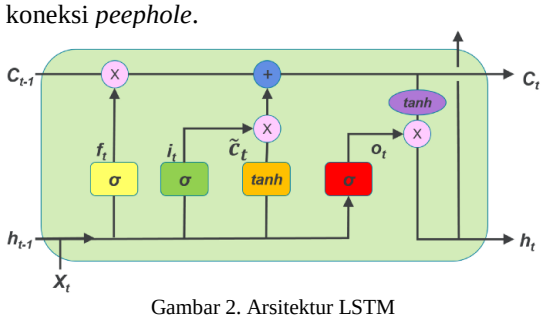
$$\hat{y}_t = softmax(V_{s_t}) \quad (2)$$

s_t merupakan status internal yang diberikan satu langkah waktu ke langkah waktu berikutnya dinamakan memori dari RN. $\hat{y}_t$ adalah keluaran kata demi kata yang diberikan oleh RNN. x_t adalah proses masukan kata demi kata. U, V, dan W merrupakan parameter-parameter yang dimiliki RNN. Sedangkan t adalah waktu. Tanh adalah fungsi aktivasi di lapisan tersembunyi dan Softmax di lapisan keluaran.

2.4. *Long Short-Term Memory*

LSTM bertujuan untuk memecahkan masalah RNN yaitu *gradient vanishing* dan *exploding* LSTM menggantikan vektor tersembunyi RNN dengan blok memori yang dilengkapi dengan gerbang. Ini dapat mempertahankan memori jangka panjang pada prinsipnya dengan melatih bobot *gating* yang tepat dan telah terbukti sangat berguna dalam mencapai *state-of-the-art* untuk berbagai masalah, termasuk pengenalan ucapan [15]. LSTM diusulkan oleh Hochreiter dan Schmidhuber, 1997 untuk secara khusus mengatasi masalah ini belajar ketergantungan jangka panjang. LSTM menyimpan sel memori terpisah di dalamnya yang dapat memperbarui dan memaparkan isinya hanya jika dianggap perlu. Mekanisme gerbang LSTM mengimplementasikan tiga lapisan; gerbang (1) masukan, (2) *forget*, dan (3) keluaran [16].

Setiap unit LSTM (Gambar 2) memiliki sel memori, dan status pada waktu t direpresentasikan sebagai c_t. Membaca dan memodifikasi dikendalikan oleh gerbang sigmoid dan berpengaruh pada gerbang masukan i_t, gerbang *forget* f_t dan gerbang keluaran o_t. LSTM dihitung sebagai berikut: Pada saat momen tth, model menerima masukan dari dua sumber eksternal (h_{t-1} dan x_t). Status tersembunyi ht dihitung oleh vektor masukan xt yang diterima jaringan pada waktu t dan status tersembunyi sebelumnya h_{t-1}. Pada saat menghitung status simpul lapisan tersembunyi, gerbang masukan, keluaran, *forget* dan x_t akan secara bersamaan mempengaruhi keadaan *node*. Selain itu, setiap gerbang memiliki sumber internal, yaitu, status sel c_{t-1} dari blok selnya. Tautan antara cell dan gerbang sendiri dirujuk ke koneksi *peephole*.



Gambar 2. Arsitektur LSTM

Di bawah ini merupakan langkah-langkah dari LSTM dan gerbang nya:

Gerbang Masukan:
Gerbang ini memutuskan nilai mana yang akan diperbarui dengan transformasi nilai antara 0 dan 1. Gerbang ini memiliki dua bagian.Pertama, lapisan sigmoid yang disebut lapisan gerbang masukan memutuskan nilai mana yang harus diperbarui (Persamaan 3). Selanjutnya, lapisan tanh membuat vektor dari kandidat baru $\check{C}_t$ yang dapat ditambah kedalam status (Persamaan 4).

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (3)$$

$$\check{C}_t = \tanh(W_C.[h_{t-1}, x_t] + b_C \quad (4)$$

Gerbang *Forget*:
Pada gerbang ini memutuskan informasi mana yang harus dibuang dan mana yang harus disimpan. Keputusan dibuat oleh lapisan sigmoid yang disebut lapisan *forget* gerbang (Persamaan 5) dimana keluaran angka antara 0 dan 1.

$$f_t = \sigma\big(W_f.[h_{t-1}, x_t + b_f\big) \quad (5)$$

Status Memori:
Pada gerbang ini memungkinkan untuk menjatuhkan nilai di cell state jika dikalikan dengan nilai mendekati 0 (Persamaan 6).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (6)$$

Gerbang Keluaran:
Pada gerbang ini membuat keputusan apa yang harus dilakukan hidden state selanjutnya. Mengingat bahwa status tersembunyi terdiri dari informasi pada masukan sebelumnya. Pertama, menjalankan lapisan sigmoid yang memutuskan bagian mana dari cell state yang akan menjadi keluaran (Persamaan 7). Kemudian, menempatkan cell state melalui tanh (Persamaan 8).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = o_t * \tanh(C_t) \quad (8)$$

2.5 *Optimizers*

Ada beberapa jenis *optimizers* untuk model *deep learning* seperti SGD, Adam, RMSProp, dan banyak lagi. Penelitian ini menerapkan Adam dan RMSProp untuk melatih data. *Optimizer* Adam dapat mengontrol masalah *sparse gradient*. Ini adalah perluasan untuk keturunan gradien stokastik yang saat ini telah melihat adopsi yang lebih luas untuk mengaplikasikan *deep learning* seperti Pemrosesan Bahasa Alami. di mana m dan v merujuk rata-rata dua momen pertama dari gradien, g menunjukkan gradien pada mini-batch saat ini.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (9)$$

$$v_t = \beta_2 v_{t-1} + (1 + \beta_2) g_t^2 \quad (10)$$

RMSProp mampu mengadaptasi tingkat pembelajaran untuk setiap parameter. Ini bertujuan untuk membagi tingkat pembelajaran untuk berat dengan rata-rata

berjalan dari besarnya gradien terbaru untuk bobot tersebut [16].

$$v(w,t) \coloneqq \gamma v(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2 \quad (11)$$

Di mana γ adalah faktor *forgetting*.
Dan parameternya adalah pembaruan seperti,

$$w \coloneqq w - \frac{n}{\sqrt{v(w,t)}} \nabla Q_i(w) \quad (12)$$

2.6. Pra-Pemrosesan

2.6.1 Tokenisasi dan Penghapusan Tanda Baca

Tokenisasi adalah proses memecah aliran teks menjadi kata, frasa, simbol, atau elemen bermakna lainnya yang disebut token. Tokenisasi berarti memecah teks menjadi unit-unit yang memiliki makna minimal. Ini adalah langkah wajib sebelum semua jenis pemrosesan. Proses ini akan membagi teks menjadi kalimat dan kalimat menjadi token tipografi. Itu berarti memisahkan tanda baca. Fitur yang dihasilkan dari tokenisasi adalah data pelatihan. Dalam proses ini *padding* juga dilakukan untuk mengidentifikasi akhir kalimat karena decoder dilatih kalimat demi kalimat.

2.6.2 One-Hot Encoding

Pra-pemrosesan pertama dalam penelitian ini adalah One-hot encoding. One-hot encoding adalah mengubah data teks (kategorikal) menjadi angka. Algoritma machine learning tidak dapat bekerja dengan data kategorikal secara langsung. Data kategorikal harus dikonversi menjadi angka. Ini berlaku karena penelitian bekerja dengan jenis klasifikasi urutan dan menggunakan metode pembelajaran mendalam seperti Long Short-Term Memory Recurrent Neural Networks.

3. Hasil dan Pembahasan

3.1. Dataset
Penelitian sebelumnya pada Zhang, dkk 2015 dan Wang, dkk 2018 telah menunjukkan hasil yang baik dengan dataset skala besar. Dari delapan dataset skala besar, dataset AGNEWS diambil untuk pelatihan. AGNews adalah klasifikasi topik dalam empat kategori artikel berita Internet yang terdiri dari judul dan deskripsi yang diklasifikasikan ke dalam empat kelas: Dunia, Hiburan, Olahraga, dan Bisnis [6],[12]. *Dataset* ditunjukkan pada Tabel 1, dengan spesifikasi konten berikut:

Table 1. Dataset Specification

Dataset	Kelas	Data
AGNews	4	496,835

3.2. Proses Pelatihan

Dataset AGNews[1] dibagi menjadi masing-masing 80% untuk pelatihan dan 20% untuk pengujian. *Dataset* pelatihan yang digunakan tidak digunakan untuk pengujian LSTM, dan sebaliknya. Dari 80% data pelatihan, 10% digunakan untuk proses validasi data. Jumlah setiap dataset dibagi secara acak, dengan pemisahan data otomatis.

3.3. Model Pelatihan

Hyper-parameter yang digunakan adalah fungsi aktivasi Relu dan Tanh, *optimizer* Adam dan RMSProp akan divalidasi dengan *learning rate* (Lr) 0,001 dan 0,0001 untuk meminimalkan error. Dimensi *word embedding* adalah 300. Dengan fungsi *Loss Categorical Cross Entropy*. Struktur dan hyper-parameter yang digunakan dalam validasi LSTM dengan fitur Word2Vec dapat ditunjukkan pada Tabel 2.

Tabel 2. Model Pelatihan LSTM dengan Word2Vec

Model	Lr	Opt	Fungsi Aktivasi		Dim
			Hidden	Keluaran	
1	0,001	Adam	Relu	Softmax	300
2	0,001	Adam	Tanh	Softmax	300
3	0,001	RMSProp	Relu	Softmax	300
4	0,001	RMSProp	Tanh	Softmax	300
5	0,0001	Adam	Relu	Softmax	300
6	0,0001	Adam	Tanh	Softmax	300
7	0,0001	RMSProp	Relu	Softmax	300
8	0,0001	RMSProp	Tanh	Softmax	300

3.4. Model LSTM

Proses pelatihan klasifikasi sekuensial LSTM menggunakan fitur *word embedding* Word2Vec 300-dimensional dilatih dengan matriks penyisipan hyper-parameter yang diperoleh dari pra-pemrosesan fitur Word2Vec pada masukan, aktivasi Relu dan Tanh pada hidden gerbang, aktivasi softmax pada keluaran gerbang, *optimizer* Adam dan RMSprop, dengan dropout 0,5 dan epoch 50, telah dilatih di masing-masing 8 model dengan tuning *learning rate* 0,001 dan 0,0001. *Learning rate Hyperparameter* mengontrol laju atau kecepatan di mana model pembelajaran. Secara khusus, ini mengontrol jumlah kesalahan terbagi yang bobot modelnya diperbarui setiap kali diperbarui, seperti pada akhir setiap kumpulan contoh pelatihan. *Learning rate* mungkin merupakan hiperparameter yang paling penting.

3.4.1. Model 1

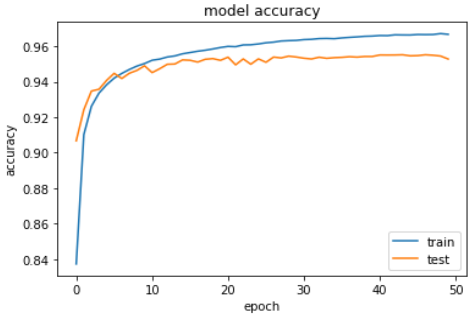
Pada Tabel 3 menunjukkan hasil kinerja evaluasi proses pelatihan LSTM dan *Confusion Matrix* yang dilatih menggunakan aktivasi Relu, *optimizer* Adam dengan *learning rate* 0,001. Sedangkan, hasil kinerja evaluasi proses pengujian dengan nilai rata-rata presisi, recall, dan f1-score pada Tabel 4. Akurasi yang didapat pada proses pelatihan sebesar 96,67 sedangkan rata-rata presisi, recall, dan f1-score adalah 95. Pada model 1 terjadi *overfitting* selama pelatihan yang mengakibatkan *loss* meningkat dan akurasi validasi menurun. Dapat dilihat pada Gambar 3 perbadingan pelatihan dan pengujian untuk kurva model akurasi dan Gambar 4 kurva model *loss*.

Tabel 3. Hasil Kinerja Evaluasi Proses Pelatihan dan *Confusion Matrix*

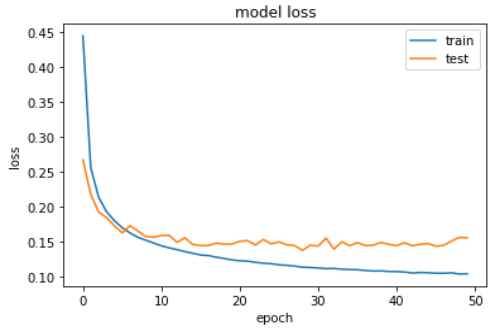
Akurasi		96,67			
Confusion Matrix	Label	0	1	2	3
	0	29563	431	236	276
	1	179	21998	819	290
	2	174	1074	20014	145
	3	72	250	54	8909

Tabel 4. Hasil Kinerja Evaluasi Proses Pengujian

Label	Presisi	Recall	F1-score	Data
0	99	97	98	30506
1	93	94	94	23286
2	95	93	94	21407
3	93	96	94	9285
Avg	95	95	95	84484



Gambar 3. Kurva Perbandingan Akurasi Pelatihan dan Pengujian 50 epoch



Gambar 4. Kurva Perbandingan *Loss* Pelatihan dan Pengujian 50 epoch

3.4.2. Model 2

Pada model 2 dilakukan pelatihan dengan aktivasi Tanh, *optimizer* Adam, dan *learning rate* 0,001. Tabel 5 menunjukkan hasil kinerja proses evaluasi pelatihan dan *Confusion Matrix* dari keempat label. Tabel 6 menunjukkan kinerja proses evaluasi pengujian dengan rata-rata nilai presisi, recall, dan f1-score. Model 2 dengan fitur Word2vec menghasilkan akurasi tertinggi dari proses pelatihan sebesar 98,76 dibanding semua model yang telah dilatih. Namun, akurasi pelatihan yang tinggi tidak menentukan hasil yang sama untuk akurasi pengujian. Hal ini dikarenakan terjadinya *overfitting* yang membuat *loss* meningkat dan akurasi validasi menurun. Model 2 masih terjadi overfitting yang signifikan dapat dilihat pada Gambar 5 kurva perbandingan pelatihan dan pengujian model akurasi dan model *loss* pada Gambar 6.

Tabel 5. Hasil Kinerja Evaluasi Proses Pelatihan dan *Confusion Matrix*

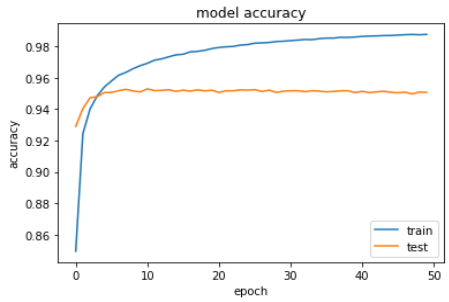
Akurasi		98,76			
Confusion Matrix	Label	0	1	2	3
	0	29617	245	208	144
	1	378	21803	991	211
	2	243	1047	20385	112
	3	158	226	98	8618

Tabel 6. Hasil Kinerja Evaluasi Proses Pengujian

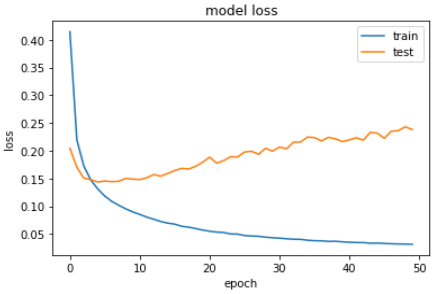
Label	Presisi	Recall	F1-score	Data
0	97	98	98	30214
1	93	93	93	23383
2	94	94	94	21787
3	95	95	95	9100
Avg	95	95	95	84484

3.4.3. Model 3

Model 3 dilatih dengan aktivasi Relu, *optimizer* RMSprop, dan *learning rate* 0,001. Hasil kinerja evaluasi pelatihan dan *Confusion Matrix* dapat dilihat pada Tabel 7 dan hasil kinerja evaluasi pengujian dengan nilai rata-rata presisi, recall, dan f1-score pada Tabel 8. Sama halnya dengan model 1 dan 2 dengan fitur Word2vec diatas, model 3 terjadi *overfitting* yang mengakibatkan *loss* terus meningkat dan akurasi menurun. Nilai akurasi pelatihan yang didapat pada model ini sebesar 96,67 dan rata-rata presisi, recall, dan f1-score sebesar 95. Terjadinya *overfitting* saat proses pelatihan dapat dilihat pada Gambar 7 untuk kurva model akurasi dimana akurasi terus menurun dikarenakan *loss* meningkat yang ditujukan pada Gambar 8.



Gambar 5. Kurva Perbandingan Akurasi Pelatihan dan Pengujian 50 epoch



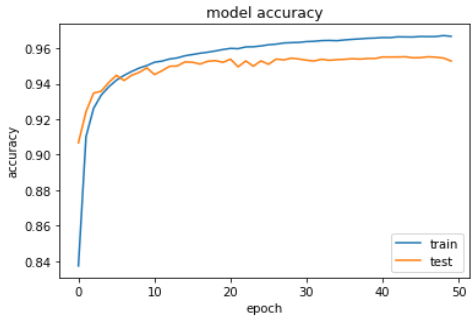
Gambar 6. Kurva Perbandingan *Loss* Pelatihan dan Pengujian 50 epoch

Tabel 7. Hasil Kinerja Evaluasi Proses Pelatihan dan *Confusion Matrix*

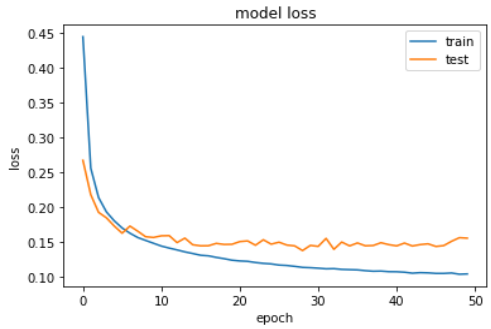
Akurasi		96,67			
Confusion Matrix	Label	0	1	2	3
	0	29563	431	236	276
	1	179	21998	819	290
	2	174	1074	20014	145
	3	72	250	54	8909

Tabel 8. Hasil Kinerja Evaluasi Proses Pengujian

Label	Presisi	Recall	F1-score	Data
0	99	97	98	30506
1	93	94	94	23286
2	95	93	94	21407
3	93	96	94	9285
Avg	95	95	95	84484



Gambar 7. Kurva Perbandingan Akurasi Pelatihan dan Pengujian 50 epoch



Gambar 8. Kurva Perbandingan *Loss* Pelatihan dan Pengujian 50 epoch

3.4.4. Model 4

Model 4 dilakukan pelatihan dengan parameter aktivasi Tanh, *optimizer* RMSprop, dan *learning rate* 0,001. Hasil kinerja evaluasi pelatihan dan *Confusion Matrix* dapat dilihat pada Tabel 9 dengan nilai akurasi pelatihan 96,74. Nilai rata-rata presisi, recall, dan f1-score terdapat pada Tabel 10 sebagai hasil kinerja evaluasi pengujian sebesar 95. Kurva perbandingan pelatihan dan pengujian dalam 50 epoch untuk model akurasi masih terjadi *overfitting* dimana nilai akurasi pengujian lebih rendah dari akurasi pelatihan dapat dilihat pada Gambar 9. Ini disebabkan oleh *loss* validasi terus meningkat sedangkan *loss* pelatihan terus menurun pada Gambar 10 untuk kurva model *loss*.

3.4.5. Model 5

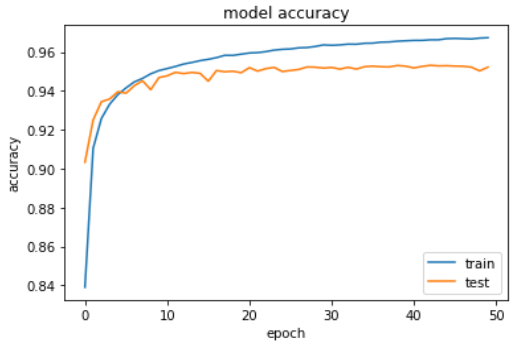
Model 5 dilakukan pelatihan dengan parameter aktivasi Relu, *optimizer* Adam, dan *learning rate* 0,0001. Hasil kinerja evaluasi pelatihan dan *Confusion Matrix* dapat dilihat pada Tabel 11 dengan nilai akurasi pelatihan 96,38. Nilai rata-rata presisi, recall, dan f1-score terdapat pada Tabel 12 sebagai hasil kinerja evaluasi pengujian sebesar 95. Meskipun akurasi pelatihan dan pengujian pada model ini tinggi, pada model 5 ini masih sedikit terjadi *overfitting* dimana *loss* pelatihan terus menurun tidak dengan *loss* validasi yang tetap datar mulai dari epoch ke 25.

Tabel 9. Hasil Kinerja Evaluasi Proses Pelatihan dan *Confusion Matrix*

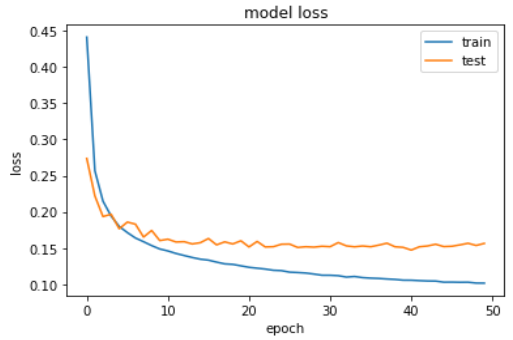
Akurasi		96,74			
Confusion Matrix	Label	0	1	2	3
	0	29783	264	177	135
	1	295	21920	836	306
	2	294	999	20254	150
	3	163	232	95	8581

Tabel 10. Hasil Kinerja Evaluasi Proses Pengujian

Label	Presisi	Recall	F1-score	Data
0	98	98	98	30359
1	94	94	94	23357
2	95	93	94	21697
3	94	95	94	9071
Avg	95	95	95	84484



Gambar 9. Kurva Perbandingan Akurasi Pelatihan dan Pengujian 50 epoch



Gambar 10. Kurva Perbandingan *Loss* Pelatihan dan Pengujian 50 epoch

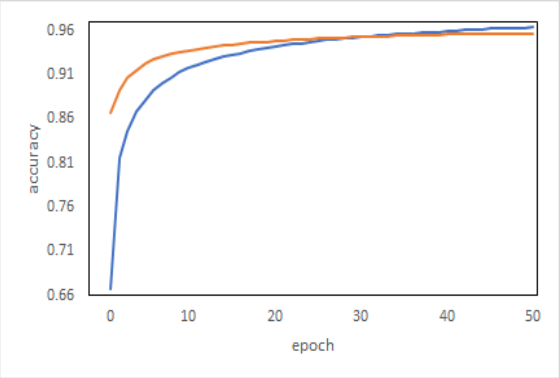
Tabel 11. Hasil Kinerja Evaluasi Proses Pelatihan dan *Confusion Matrix*

Akurasi		96,38			
Confusion Matrix	Label	0	1	2	3
	0	29880	259	216	132
	1	284	21670	949	220
	2	275	1036	20512	85
	3	152	197	92	8525

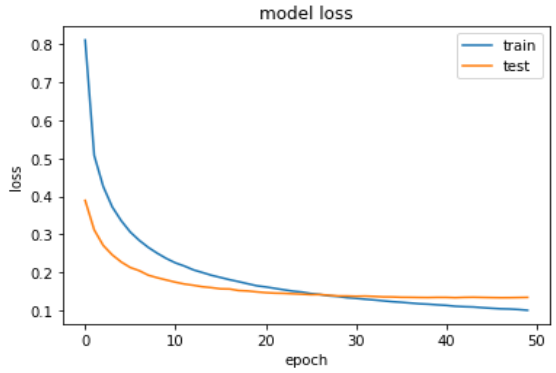
Tabel 12. Hasil Kinerja Evaluasi Proses Pengujian

Label	Presisi	Recall	F1-score	Data
0	98	98	98	30487
1	94	94	94	23123
2	94	94	94	21908
3	94	95	95	8966
Avg	95	95	95	84484

Kurva model akurasi yang menunjukkan masih terjadinya *overfitting* dimana akurasi pelatihan terus meningkat namun akurasi pengujian masih pada nilai yang sama mulai dari epoch ke 25 dapat dilihat pada Gambar 11 dan kurva model *loss* pada Gambar 12.



Gambar 11. Kurva Perbandingan Akurasi Pelatihan dan Pengujian 50 epoch



Gambar 12. Kurva Perbandingan *Loss* Pelatihan dan Pengujian 50 epoch

3.4.6. Model 6

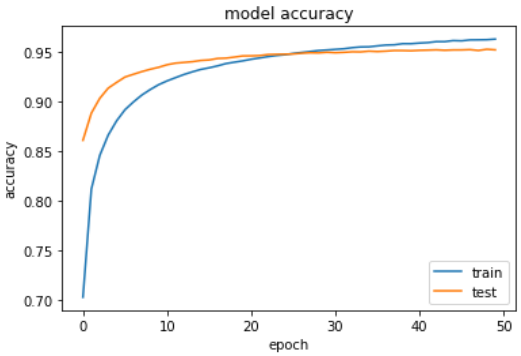
Parameter yang dilatih dengan menggunakan aktivasi Tanh, *optimizer* Adam, dan *learning rate* 0,0001. Hasil kinerja evaluasi pelatihan dan *Confusion Matrix* dapat dilihat pada Tabel 13 dengan nilai akurasi pelatihan 96,33. Hasil kinerja evaluasi pengujian dengan nilai rata-rata presisi, recall, dan f1-score dapat dilihat pada Tabel 14. Pada Gambar 13 adalah kurva model akurasi dimana akurasi pengujian yang dihasilkan meningkat tetapi tidak diatas akurasi pelatihan. Ini disebabkan oleh nilai *loss* validasi yang tidak melakukan pergerakan penurunan yang signifikan dari *loss* pelatihan dapat dilihat pada Gambar 14, kedua kurva mendekati *fit* diantara epoch ke 25.

Tabel 13. Hasil Kinerja Evaluasi Proses Pelatihan dan *Confusion Matrix*

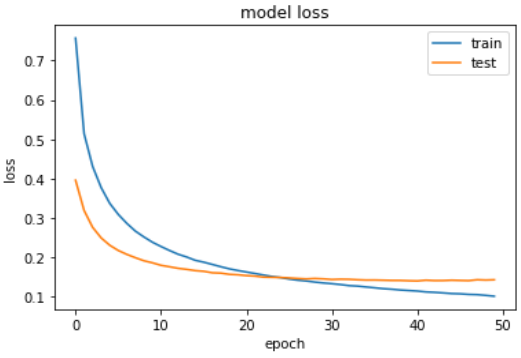
Akurasi		96,33			
Confusion Matrix	Label	0	1	2	3
	0	29904	257	225	92
	1	292	21874	895	177
	2	316	1101	20178	85
	3	146	234	98	8610

Tabel 14. Hasil Kinerja Evaluasi Proses Pengujian

Label	Presisi	Recall	F1-score	Data
0	98	98	98	30478
1	93	94	94	23238
2	94	93	94	21680
3	96	95	95	9088
Avg	95	95	95	84484



Gambar 13. Kurva Perbandingan Akurasi Pelatihan dan Pengujian 50 epoch



Gambar 14. Kurva Perbandingan *Loss* Pelatihan dan Pengujian 50 epoch

3.4.7. Model 7

Model 7 dilatih dengan parameter aktivasi Relu, *optimizer* RMSprop, dan *learning rate* 0,0001. Tabel 15 menunjukkan hasil kinerja evaluasi pelatihan dan multilabel *Confusion Matrix* dengan nilai akurasi pelatihan 94,49. Hasil kinerja evaluasi pengujian berupa nilai rata-rata presisi, recall, dan f1-score dapat

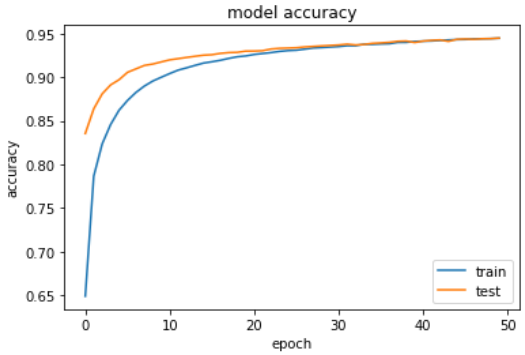
dilihat pada Tabel 16. Dari pelatihan yang dilakukan, model 7 dengan *optimizer* RMSprop dan *learning rate* 0,0001 menghasilkan kurva model akurasi mendekati *good-fit* sampai epoch terakhir yang ditunjukkan pada Gambar 15 dan kurva model *loss* pada Gambar 16.

Tabel 15. Hasil Kinerja Evaluasi Proses Pelatihan dan *Confusion Matrix*

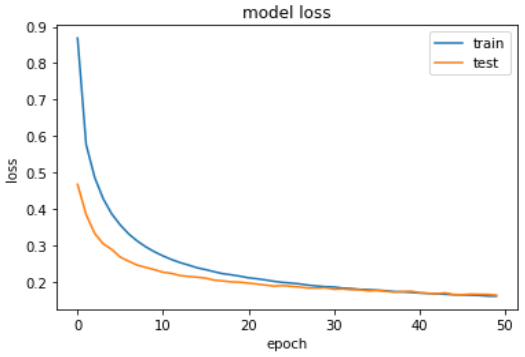
Akurasi		94,49			
Confusion Matrix	Label	0	1	2	3
	0	29756	325	215	112
	1	405	21390	1140	238
	2	397	1019	20384	132
	3	224	303	113	8331

Tabel 16. Hasil Kinerja Evaluasi Proses Pengujian

Label	Presisi	Recall	F1-score	Data
0	97	98	97	30408
1	93	92	93	23173
2	93	93	93	21932
3	95	93	94	8971
Avg	95	95	95	84484



Gambar 15. Kurva Perbandingan Akurasi Pelatihan dan Pengujian 50 epoch



Gambar 16. Kurva Perbandingan *Loss* Pelatihan dan Pengujian 50 epoch

3.4.8. Model 8

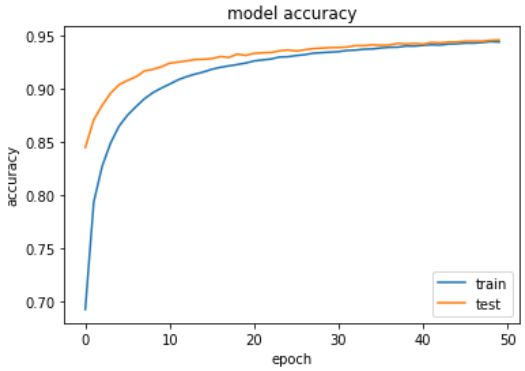
Model 8 dilatih menggunakan *hyperparameter* yang sama, aktivasi Tanh dengan *optimizer* RMSprop dan *learning rate* 0,0001. Hasil kinerja evaluasi pelatihan dapat dilihat pada Tabel 17 dengan akurasi pelatihan sebesar 94,37. Hasil kinerja evaluasi pengujian dengan nilai rata-rata presisi, recall, dan f1-score dapat dilihat pada Tabel 18. Setelah dilakukan pelatihan kedelapan model LSTM dengan fitur *word embedding* Word2vec, model 8 adalah model terbaik dengan kurva mendekati *good-fit* dimana akurasi pelatihan dan pengujian sama, begitu juga dengan nilai presisi, recall, dan f1-score.

Tabel 17. Hasil Kinerja Evaluasi Proses Pelatihan dan *Confusion Matrix*

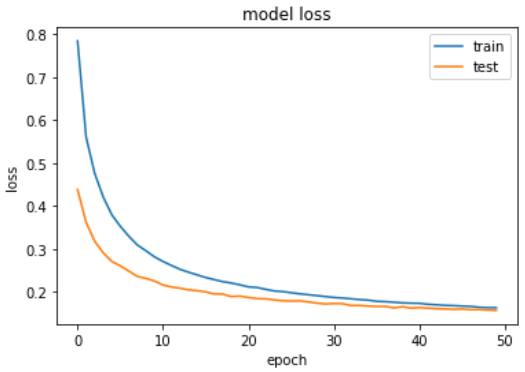
Akurasi		94,37			
Confusion Matrix	Label	0	1	2	3
	0	29937	294	239	143
	1	418	21222	1265	285
	2	413	960	20059	118
	3	217	256	126	8532

Tabel 18. Hasil Kinerja Evaluasi Proses Pengujian

Label	Presisi	Recall	F1-score	Data
0	97	98	97	30613
1	93	92	92	23190
2	92	93	93	21550
3	94	93	94	9131
Avg	94	94	94	84484



Gambar 17. Kurva Perbandingan Akurasi Pelatihan dan Pengujian 50 epoch



Gambar 18. Kurva Perbandingan *Loss* Pelatihan dan Pengujian 50 epoch

Kurva model akurasi dapat dilihat pada Gambar 17 dan kurva model *loss* dapat dilihat pada Gambar 18. Pada Tabel 19 menunjukkan perbandingan akurasi pengujian dari kedelapan model LSTM menggunakan fitur *word*

embedding Word2vec. Tabel 20 menampilkan perbandingan hasil dari penelitian sebelumnya.

Tabel 19. Akurasi pengujian dari kedelapan model LSTM Menggunakan Fitur Word2vec

Model	Neuron	Lr	Optimizer	Fungsi Aktivasi	Akurasi
1	128	0,001	Adam	Relu	95,26
2	128	0,001	Adam	Tanh	95,19
3	128	0,001	RMSProp	Relu	95,26
4	128	0,001	RMSProp	Tanh	95,32
5	128	0,0001	Adam	Relu	95,38
6	128	0,0001	Adam	Tanh	95,36
7	128	0,0001	RMSProp	Relu	94,52
8	128	0,0001	RMSProp	Tanh	94,39

Tabel 20. Perbadingan Hasil Penelitian Sebelumnya

Model	AGNews
Bag-of-words (Zhang et al.,2015)	88,8
Small word CNN (Zhang et al.,2015)	89,13
Large word CNN (Zhang et al.,2015)	91,45
LSTM (Zhang et al.,2015)	86,06
Deep CNN (29 layer) (Conneau et al.,2017)	91,27
SWEM (Shen et al.,2018)	92,24
fastText (Joulin et al.,2016)	92,5
LEAM (Wang et al., 2018)	92,45
LEAM (linear) (Wang et al., 2018)	91,75
Word2vec + LSTM	95,38

4. Kesimpulan

Klasifikasi teks menggunakan LSTM dilakukan dengan melakukan percobaan trial dan error. Klasifikasi teks menggunakan LSTM dengan fitur Word2Vec melakukan tuning hyper-parameter untuk mendapatkan model yang optimal. Sedangkan, LSTM dan struktur hyperparameter yang digunakan dari hasil pengujian menggunakan embedding fitur Word2Vec dan Word2vec sebagai masukan, fungsi aktivasi softmax dalam keluaran, fungsi aktivasi Relu dan Tanh, fungsi loss categorical crossentropy, *learning rate* 0,001 dan 0,0001, dengan jumlah epoch 50.

Akurasi tertinggi dengan fitur Word2Vec adalah pada model keenam 95,17 dengan rata-rata presisi, recall, dan F1-score yaitu 95. Untuk akurasi tertinggi dengan fitur Word2vec terdapat model kelima sebesar 95,38 dengan rata-rata presisi, recall, dan F1-score 95. Sementara itu, akurasi tertinggi untuk LSTM tanpa fitur tambahan adalah pada model pertama 95,22 dengan presisi rata-rata, recall, dan F1-score 95. Dapat disimpulkan bahwa hasil evaluasi kinerja LSTM menggunakan fitur Word2Vec adalah fitur terbaik untuk akurasi dan loss model kurva, sedangkan untuk akurasi dan metrik pengukuran presisi, recall, dan F1-score fitur Word2vec menghasilkan nilai tertinggi.

Daftar Rujukan

[1] Li, L., Xiao, L., Jin, W., Zhu, H., & Yang, G. (2018). Text Classification Based on Word2vec and Convolutional Neural Network. International Conference on Neural Information Processing, 3, 450–460. https://doi.org/10.1007/978-3-030-04221-9.

[2] Lilleberg, J., Zhu, Y., & Zhang, Y. (2015). Support vector machines and Word2vec for text classification with semantic features. Proceedings of 2015 IEEE 14th International Conference on Cognitive Informatics and Cognitive Computing, ICCI*CC 2015, 136–140. https://doi.org/10.1109/ICCI-CC.2015.7259377.

[3] Rossi, R. G., Lopes, A. D. A., & Rezende, S. O. (2016). Optimization and label propagation in bipartite heterogeneous networks to improve transductive classification of texts. Information Processing and Management, 52(2), 217–257. https://doi.org/10.1016/j.ipm.2015.07.004

[4] Heidarysafa, M., Kowsari, K., Brown, D. E., Meimandi, K. J., & Barnes, L. E. (2018). An Improvement of Data Classification Using Random Multimodel Deep Learning (RMDL). International Journal of Machine Learning and Computing, 8(4), 298–310. https://doi.org/10.18178/ijmlc.2018.8.4.703.

[5] Goudjil, M., Koudil, M., Bedda, M., & Ghoggali, N. (2018). A Novel Active Learning Method Using SVM for Text Classification. International Journal of Automation and Computing, 15(3), 290–298. https://doi.org/10.1007/s11633-015-0912-z.

[6] Zhang, X., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in Neural Information Processing Systems, 1–9. Retrieved from http://arxiv.org/abs/1502.01710.

[7] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference, 2, 427–431. https://doi.org/10.18653/v1/e17-2068.

[8] Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., Carin, L. (2018). Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), 1, 440–450. https://doi.org/10.18653/v1/p18-1041.

[9] Kowsari, K., Brown, D. E., Heidarysafa, M., Jafari Meimandi, K., Gerber, M. S., & Barnes, L. E. (2017). HDLTex: Hierarchical Deep Learning for Text Classification. Proceedings - 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017, 2017-Decem, 364–371. https://doi.org/10.1109/ICMLA.2017.0-134.

[10] Yan, Y., Wang, Y., Gao, W. C., Zhang, B. W., Yang, C., & Yin, X. C. (2018). LSTM^2: Multi-Label Ranking for Document Classification. Neural Processing Letters, 47(1), 117–138. https://doi.org/10.1007/s11063-017-9636-0.

[11] Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2016). Very Deep Convolutional Neural Networks for Text Classification. ArXiv Preprint ArXiv:1606.01781 (2016)., 11727 LNCS, 193–207. https://doi.org/10.1007/978-3-030-30487-4_16.

[12] Wang, G., Li, C., Wang, W., Zhang, Y., Shen, D., Zhang, X., Carin, L. (2018). Joint embedding of words and labels for text classification. ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), 1, 2321–2331. https://doi.org/10.18653/v1/p18-1216.

[13] Choi, K., Fazekas, G., & Sandler, M. K. C. (2017). Convolutional recurrent neural networks for music classification. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). https://doi.org/10.1109/ICASSP.2017.7952585.

[14] Zen, H., & Sak, H. (2015). Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2015-August, 4470–4474. https://doi.org/10.1109/ICASSP.2015.7178816.

[15] Chiu, C., Sainath, T. N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., … Bacchiani, M. (2018). State-Of-The-Art Speech Recognition With Sequence-To-Sequence Models. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4774–4778. https://doi.org/10.1109/ICASSP.2018.8462105.

[16] Kumar, A., & Rastogi, R. (2019). Attentional Recurrent Neural Networks for Sentence Classification. In Innovations in Infrastructure, 549-559. Springer, Singapore. https://doi.org/10.1007/978-981-13-1966-2_49.