Indonesian Journal of Computer Science
ISSN 2302-4364 (*print*) dan 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam, No. 1, Padang, Indonesia, Telp. (0751) 7056199, 7058325
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Opinion Mining menggunakan Algoritma Deep Learning untuk Menganalisis Penggunaan Aplikasi Jamsostek Mobile

Zahra Azhari[1], Lusiana Efrizoni[2], Wirta Agustin[3], Rini Yanti[4]
zahraazhrii23@gmail.com[1*], lusiana@stmik-amik-riau.ac.id[2], wirtaagustin10@gmail.com[3], riniyanti@stmik-amik-riau.ac.id[4]
[1,2,3,4] STMIK AMIK Riau Pekanbaru

Informasi Artikel

Diterima : 10 Apr 2023
Direview : 19 Apr 2023
Disetujui : 27 Apr 2023

Kata Kunci

BPJS Ketenagakerjaan, *Deep Learning*, Jamsostek *Mobile*, *Long Short Term Memory*, *Opinion Mining*

Abstrak

BPJS Ketenagakerjaan berperan dalam menjaga kesejahteraan para pekerja dan buruh melalui program-program pendidikan dan pelatihan yang diberikan, pelayanan menjadi prioritas terhadap pelanggan untuk memberikan kenyamanan. Melalui aplikasi Jamsostek *Mobile* yang terdapat di *Google Playstore* akan diambil komentar-komentar untuk mendapatkan respon pelanggan terhadap aplikasi Jamsostek *Mobile* untuk dilakukan *opinion mining*. Komentar yang diambil dari *google playstore* menggunakan bantuan *googleplayscraper*, sebanyak 3000 komentar berhasil diambil yang kemudian akan dilakukan tahap pembersihan data, pelabelan, pembobotan kata menggunakan *word2vec* 300 dimensi dan dilanjutkan menggunakan algoritma *Long Short Term Memory*. Hasil *opinion mining* menunjukkan dominasi sentimen negatif sebesar 80.58% dan 19.42% positif dengan tingkat akurasi terbaik yang dihasilkan oleh algoritma LSTM sebesar 87.36%. Hasil penelitian ini akan memberikan wawasan yang berguna bagi pengembang aplikasi untuk meningkatkan kualitas pelayanan dan pengalaman pengguna.

Keywords

Employment BPJS, *Deep Learning*, Jamsostek *Mobile*, *Long Short Term Memory*, *Opinion Mining*

Abstrak

Employment BPJS plays a role in maintaining the welfare of workers and laborers through the education and training programs provided, service is a priority for customers to provide comfort. Through the Jamsostek Mobile application available on the Google Playstore, comments will be taken to get customer responses to the JMO application for opinion mining. Comments were taken from Google Playstore using the help of GooglePlayScraper, as many as 3000 comments were successfully retrieved which would then be carried out in the data cleaning, labeling, word weighting stages using 300 dimensional word2vec and continued using the Long Short Term Memory algorithm. Opinion mining results show the dominance of negative sentiment of 80.58% and positive 19.42% with the best accuracy rate produced by the LSTM algorithm of 87.36%. The results of this study will provide useful insights for application developers to improve service quality and user experience.

A. Pendahuluan

BPJS Ketenagakerjaan merupakan program publik yang memberikan perlindungan bagi tenaga kerja untuk mengatasi risiko sosial ekonomi tertentu dan penyelenggaraannya menggunakan mekanisme asuransi sosial [1] PT Jamsostek (Persero) memberikan perlindungan 4 (empat) program, yang mencakup Program Jaminan Kecelakaan Kerja (JKK), Jaminan Kematian (JKM), Jaminan Hari Tua (JHT) dan Jaminan Pensiun (JP) bagi seluruh tenaga kerja dan keluarganya [2]. BPJS Ketenagakerjaan memiliki segmentasi masyarakat yang merupakan tenaga kerja di Indonesia sehingga BPJS Ketenagakerjaan harus mampu memberikan pelayanan yang terbaik kepada pesertanya. Pelayanan yang diberikan kepada peserta berkaitan erat dengan *customer service*. *Customer Service* di bawah bidang Pelayanan memiliki tugas yang penting dalam memberikan pelayanan secara langsung kepada peserta [3]. Untuk mendapatkan pelayanan yang baik kepada pelanggan kita harus mengetahui respon dari masyarakat terhadap Aplikasi Jamsostek mobile dengan melakukan *opinion mining*.

Opinion mining merupakan cabang penelitian dari *text* mining [4] digunakan untuk mengatasi masalah dalam mengelompokan opini atau *review* sebagai opini positif, negatif, atau netral secara otomatis [5]. *Opinion mining* dapat dimanfaatkan untuk mengetahui respon pengguna media sosial terhadap suatu isu, sehingga apabila diperlukan pengambilan kebijakan bagi banyak pihak, hasil dari *opinion mining* dapat dijadikan bahan pertimbangan [6]. Respon masyarakat terhadap aplikasi JMO diambil dari komentar pada aplikasi JMO yang kemudian diolah menggunakan algoritma *Deep Learning*.

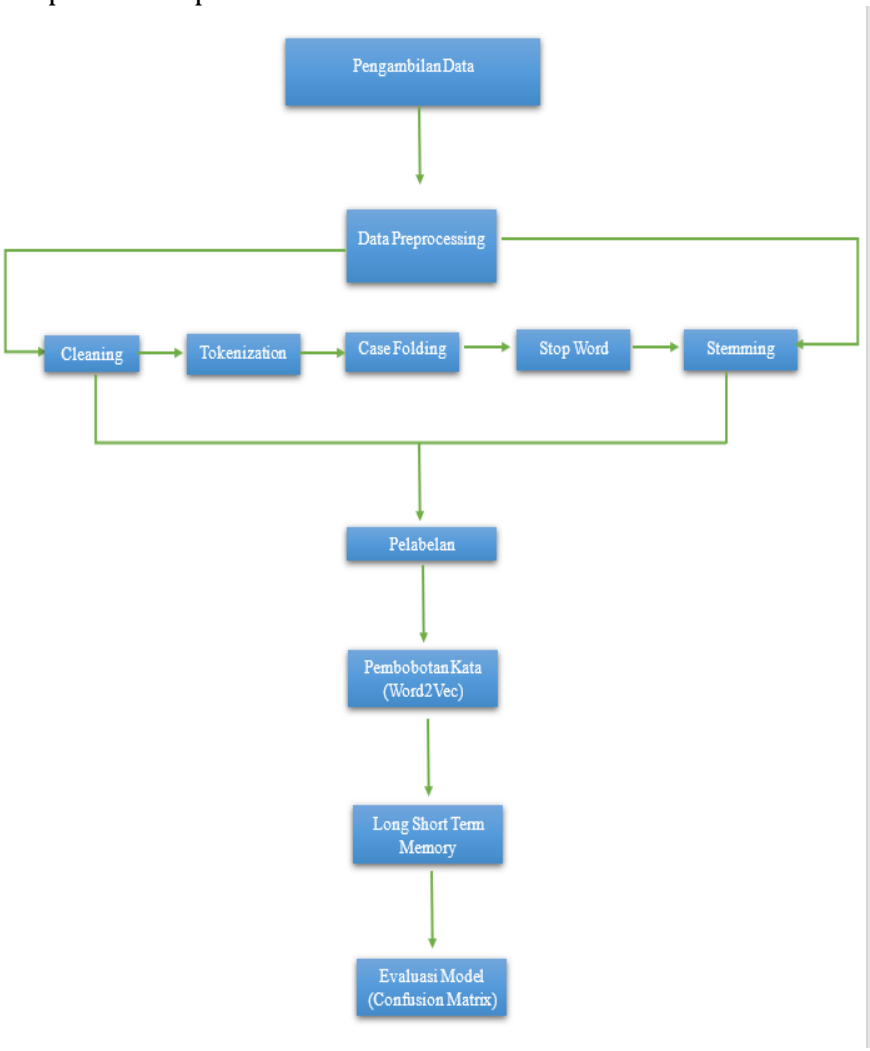
Deep Learning adalah merupakan cabang ilmu *machine learning* berbasis Jaringan Saraf Tiruan (JST) atau bisa dikatakan sebagai perkembangan dari JST [7]. Algoritma jaringan saraf *Deep Learning* tidak memerlukan informasi apapun terhadap data yang akan dipelajari. Algoritma dapat melakukan *tuning* (penyetelan) secara mandiri dan melakukan pemilihan model yang paling optimal [8], salah satu algoritma *deep learning* yang dapat digunakan untuk melakukan klasifikasi teks yaitu *Long Short Term Memory*.

LSTM merupakan pengembangan dari algoritma *Recurrent Neural Network* (RNN) dimana mengatasi masalah utama RNN yaitu tidak bisa mengolah informasi *sequensial* dalam jangka panjang [9]. LSTM mempunyai kelebihan dapat menyimpan informasi jangka panjang [10] Terdapat tiga jenis unit gerbang berbeda yang digunakan dalam LSTM, yaitu *input gate, forget gate, dan output gate* [11].

Penelitian sebelumnya pernah dilakukan oleh [12] menggunakan algoritma *machine learning* yaitu *naïve bayes* untuk analisis sentimen pada BPJS ketenagakerjaan, penelitian ini mendapatkan nilai akurasi sebesar 90%. Penelitian berikutnya pernah dilakukan oleh [13] menggunakan machine learning yaitu support vector machine mendapatkan nilai akurasi sebesar 82.92%. Berdasarkan penelitian sebelumnya, analisis sentimen sudah pernah dilakukan menggunakan model machine learning. Penemuan terbaru dari penelitian ini yaitu data diambil langsung berdasarkan komentar pengguna pada aplikasi Jamsostek Mobile dan menggunakan model deep learning.

B. Metode Penelitian

Metode penelitian analisis sentimen pada aplikasi BPJS Ketenagakerjaan yaitu Jamsostek *Mobile* menggunakan model *deep learning* dengan algoritma *Long Short Term Memory* dapat dilihat pada Gambar 1



Gambar 1. Alur Penelitian

:

Pengambilan Data

Pengambilan data dilakukan menggunakan *script Python*, data yang diambil berasal dari komentar aplikasi JMO yang terdapat pada *google playstore*. Data diambil pada periode 21 Desember 2018 hingga 1 Februari 2023 dengan jumlah data yang terkumpul sebanyak 3000 komentar.

Data *Preprocessing*

Data *preprocessing* bertujuan untuk mengolah data sebelum dilakukan analisis dan pemodelan agar data tersebut sesuai dengan kebutuhan dan memiliki kualitas yang baik sehingga mempermudah dan mempercepat proses analisis dan pemodelan. Ada 5 tahap yang dilakukan pada *prerpocessing* yaitu

1. *Data Cleaning* adalah proses pembersihan dan pembersihan data sebelum digunakan untuk analisis atau pemodelan.

2. *Tokenization* proses pemecahan suatu teks atau dokumen menjadi potongan-potongan kecil yang lebih terstruktur, biasanya disebut "token".
3. *Case Folding* proses pemformatan teks dalam suatu dokumen yang mengubah huruf besar menjadi huruf kecil atau sebaliknya.
4. *Stop Word* adalah kata-kata yang sering muncul biasanya tidak memiliki informasi yang penting dalam konteks pencarian dan analisis teks.
5. *Stemming* adalah proses pengurangan kata menjadi kata dasar atau akar kata. Ini dilakukan untuk membuat kata-kata yang memiliki akar yang sama dapat diperlakukan sebagai satu kata saja

Pelabelan

Pelabelan teks adalah proses menandai atau memberikan label pada bagian-bagian tertentu dalam teks untuk mengindikasikan jenis informasi yang terkandung di dalamnya, dalam kasus ini teks akan diberikan label positif dan negatif. Pelabelan teks sangat berguna bagi pemrosesan data teks karena mempermudah dalam mengidentifikasi bagian-bagian penting dalam teks dan mempermudah dalam analisis data.

Pembobotan Kata

Pembobotan kata adalah proses memberikan nilai atau bobot tertentu pada setiap kata dalam suatu dokumen atau teks. Tujuan pembobotan ini adalah untuk menentukan penting atau signifikansi dari setiap kata dalam teks tersebut, yang akan membantu dalam proses pencarian informasi, pengelompokan dokumen, dan analisis teks, pada penelitian ini akan menggunakan *word2vec* untuk pembobotan kata

Long Short Term Memory

Long Short-Term Memory (LSTM) adalah jenis algoritma pembelajaran mesin yang digunakan untuk memproses data seri dan memprediksi hasil berdasarkan data sejarah. LSTM memiliki kemampuan untuk mengingat informasi jangka panjang dan mengabaikan informasi yang tidak relevan, sehingga dapat membuat prediksi yang lebih akurat. LSTM digunakan dalam berbagai aplikasi, seperti pengenalan suara, pemrosesan bahasa alami, dan pemrosesan data teks. Rumus algoritma LSTM [14] adalah sebagai berikut

Forget Gate

$$ft = \sigma(WfxXt + Wfhht - 1 + WfcCt - 1 + bf) \quad (1)$$

Input Gate

$$it = \sigma(WixXt + Wih\,ht - 1 + WicCt - 1 + bi) * tanh(WcxXt + Wchht - 1 + bi) \quad (2)$$

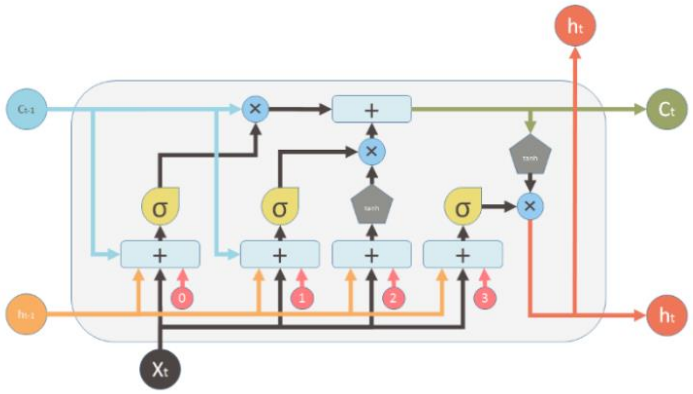
$$Ct = ft * Ct - 1 + it \quad (3)$$

Output Gate

$$Ot = \sigma(WoxXt + Wohht - 1 + WocCt - 1 + bo) \quad (4)$$

$$ht = Ot * tanh(Ct) \quad (5)$$

ft adalah *Forget Gate*, it adalah *Input Gate*, Ct adalah *Update Cell State*, Ot adalah *Output Gate*, ht adalah Nilai *Output*, Xt adalah Nilai *Input*, $ht-1$ adalah Nilai *Output Cell* sebelumnya, $Ct-1$ adalah *Cell State* sebelumnya, b adalah Bias, W adalah *Weight*, σ adalah *Sigmoid*. Arsitektur LSTM [15] dapat dilihat pada Gambar 2 dapat dilihat pada Gambar 2



Gambar 2. Arsitektur Long Short Term Memory

LSTM memiliki unit-unit cel yang membuat informasi baru dan mempertahankan informasi yang sudah ada. Cel ini memiliki tiga kontrol lup: *input*, *forget*, dan *output*. Kontrol *input* memperbarui informasi yang disimpan dalam cel, kontrol *forget* memungkinkan cel untuk memutuskan informasi yang tidak perlu untuk diteruskan, dan kontrol *output* memperbarui informasi yang disimpan dan diteruskan ke cel berikutnya, mempertahankan informasi yang relevan dan membuang informasi yang tidak perlu, LSTM dapat mengatasi masalah *vanishing gradient* yang sering terjadi pada RNN. Masalah ini terjadi ketika informasi yang diteruskan dari waktu ke waktu menjadi sangat kecil dan tidak berguna.

Evaluasi Model (*Confusion Matrix*)

Confusion matrix adalah sebuah tabel yang digunakan untuk mengevaluasi kinerja suatu model pembelajaran mesin. Ini menunjukkan bagaimana model memprediksi setiap label dan membandingkan hasil prediksi dengan label sebenarnya. Tabel 1 menyajikan *confusion matrix*

Tabel 1. Confusion Matrix

		Predicted	
		Positif	**Negatif**
Positif		TP	FN
Negatif		FP	TN

True Positif (TP): Jumlah data yang diklasifikasikan sebagai positif oleh model dan juga benar sebagai positif.
False Positif (FP): Jumlah data yang diklasifikasikan sebagai positif oleh model namun salah sebagai positif.
True Negatif (TN): Jumlah data yang diklasifikasikan sebagai negatif oleh model dan juga benar sebagai negatif.

False Negatif (FN): Jumlah data yang diklasifikasikan sebagai negatif oleh model namun salah sebagai negatif.

Dalam *confusion matrix*, dapat ditemukan beberapa metrik yang digunakan untuk mengevaluasi kinerja model seperti akurasi, presisi, recall, dan nilai F1-*score*. Rumus untuk masing-masing metrik tersebut adalah sebagai berikut

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (6)$$

Akurasi adalah suatu metrik yang digunakan untuk mengukur tingkat keakuratan dalam memprediksi suatu model.

$$Recall = \frac{TP}{TP+FN} \quad (7)$$

Recall adalah ukuran kemampuan suatu model dalam mengidentifikasi seluruh item benar sebagai positif.

$$Precision = \frac{TP}{TP+FP} \quad (8)$$

Presisi (*precision*) adalah mengukur seberapa sering model memprediksi benar bahwa instance memiliki label positif.

$$F1 - Score = 2\,x\frac{Recall\ Precision}{Recall\ Precision} \quad (9)$$

F1 *score* menggabungkan precision dan recall menjadi satu nilai untuk memperoleh gambaran umum tentang performa model.

C. Hasil dan Pembahasan

Pengambilan Data

Pengambilan data dilakukan menggunakan bantuan *google play scraper*, data yang diambil berjumlah 3000 komentar dengan 3 variabel yaitu *username*, *rating* dan *content*. Gambar 3 menampilkan cara pengambilan data

```
In [43]: from google_play_scraper import Sort, reviews

result, continuation_token = reviews(
    'com.bpjstku',
    lang='id',
    country='id',
    sort=Sort.MOST_RELEVANT,
    count=3000,
    filter_score_with=None
)
```

Gambar 3. Pengambilan Data

Berdasarkan Gambar 3 pengambilan data dilakukan menggunakan bantuan *google play scraper*, data berasal dari komentar aplikasi Jamsostek *Mobile*, hanya komentar berbahasa indonesia dan serta komentar terbaru dengan jumlah 3000 yang akan diambil. Hasil data yang diambil disajikan dalam Tabel 2

Tabel 2. Hasil Pengambilan Data

Username	Rating	Content
Kelxxx	5	Alhamdulillah aplikasinya bagus, ga ada masalah, pencarian klaim Jamsostek juga tidak lama, hari itu juga, penggunaan aplikasinya juga mudah.
Abedxxx	1	Mau klaim JHT, aplikasinya malah bug.. nggak berguna
Rifkxxx	4	Harus bersabar Sangat susah diakses awalnya. Namun begitu sudah bisa masuk cepet prosesnya.. terimakasih JMO
Ismxxx	3	terkadang susah masuk aplikasi,diakalin sampai mau stresss hahaha
Khaxx	2	Mintak update tapi pas di buka lari ke playstore gak ada update nya.lamaÂ² jmo jadi burik

Data *Preprocessing*

Data *preprocessing* dilakukan untuk pembersihan, transformasi, dan seleksi data sebelum dilakukan analisis, untuk memastikan hasil yang akurat dan relevan. Tabel 3 menyajikan hasil dari tahap *prepocessing*.

Tabel 3. Tahap Data *Preprocessing*

Tahap	Sebelum	Sesudah
Data Cleaning	Suka banget, dulu sempat buat antrian untuk klaim..tapi tak sempat sempat, dengan adanya aplikasi ini Alhamdulillah bisa klaim.. mantap	Suka banget dulu sempat buat antrian untuk klaim tapi tak sempat sempat dengan adanya aplikasi ini Alhamdulillah bisa klaim mantap
Case Folding	Suka banget, dulu sempat buat antrian untuk klaim..tapi tak sempat sempat, dengan adanya aplikasi ini Alhamdulillah bisa klaim.. mantap	suka banget dulu sempat buat antrian untuk klaim tapi tak sempat sempat dengan adanya aplikasi ini alhamdulillah bisa klaim mantap
Tokenization	Suka banget, dulu sempat buat antrian untuk klaim..tapi tak sempat sempat, dengan adanya aplikasi ini Alhamdulillah bisa klaim.. mantap	['suka', 'banget', 'dulu', 'sempat', 'buat', 'antrian', 'untuk', 'klaim', 'tapi', 'tak', 'sempat', 'sempat', 'dengan', 'adanya', 'aplikasi', 'ini', 'alhamdulillah', 'bisa', 'klaim', 'mantap']

Stopword	['suka', 'banget', 'dulu', 'sempat', 'buat', 'antrian', 'untuk', 'klaim', 'tapi', 'tak', 'sempat', 'sempat', 'dengan', 'adanya', 'aplikasi', 'ini', 'alhamdulillah', 'bisa', 'klaim', 'mantap']	['suka', 'banget', 'antrian', 'klaim', 'aplikasi', 'alhamdulillah', 'klaim', 'mantap']
Stemming	['suka', 'banget', 'antrian', 'klaim', 'aplikasi', 'alhamdulillah', 'klaim', 'mantap']	suka banget antri klaim aplikasi alhamdulillah klaim mantap

Pelabelan

Pelabelan dilakukan dengan menggunakan rumus *if* pada *python* dengan ketentuan

If *score* ≥ 4 maka label positif
If *score* = 3 maka label netral
If *score* ≤ 2 maka label negatif
Proses pelabelan dapat dilihat pada Gambar 4

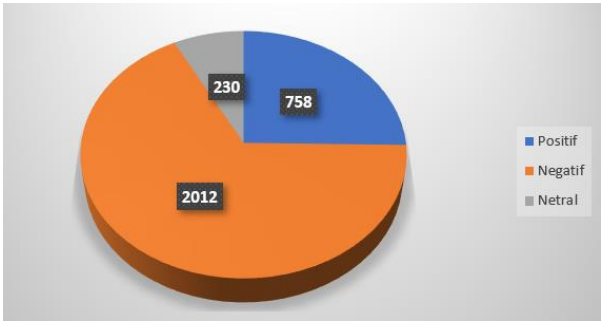
```
In [18]: conditions = [
             label.score >= 4,
             label.score == 3,
             label.score <= 2,
         ]

         values = ['Positive', 'Neutral', 'Negative']

         label['score'] = np.select(conditions, values)
```

Gambar 4. Pelabelan Teks

Setelah dilakukan pelabelan, maka akan dihitung jumlah data dari masing-masing label, perhitungan label tersebut disajikan pada Gambar 5

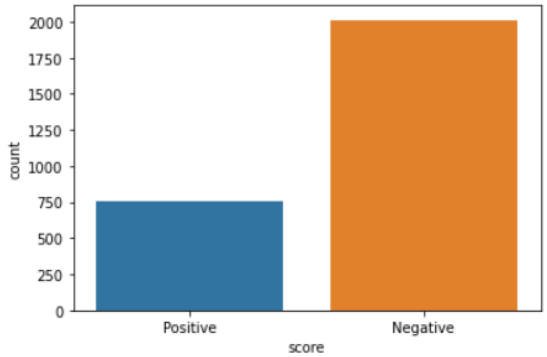


Gambar 5. Jumlah Data Pelabelan

Berdasarkan Gambar 5 hasil pelabelan menunjukkan jumlah komentar negatif 2012, netral 230 dan positif 758. Label Netral akan dibuang karena tidak dijadikan sebagai acuan pada penelitian ini, sehingga hanya positif dan negatif yang digunakan, hasil penghapusan label netral disajikan pada Gambar 6

```
Negative    2012
Positive     758
Name: score, dtype: int64

<AxesSubplot:xlabel='score', ylabel='count'>
```



Gambar 6. Label Positif dan Negatif

Berdasarkan Gambar 6 setelah dilakukan penghapusan data pada label netral sehingga data yang digunakan menjadi 2770.

Pembobotan Kata (*Word2Vec*)

Pembobotan kata bertujuan untuk memberikan angka pada setiap *text*, kata-kata yang berhubungan dengan 'aplikasi' dapat dilihat pada Tabel 4

Tabel 4. Similar Kata

Similar Kata dengan Kata aplikasi	Nilai
tutup	0.9945491552352905
magang	0.9934059381484985
pikir	0.9928140640258789
benerin	0.9926470518112183
erorr	0.9925312995910645
kesini	0.9924899339675903
gin	0.9922713041305542
ajj	0.9920620918273926
querynya	0.992050290107727

Berdasarkan Tabel 4 dapat dilihat bahwa kata aplikasi banyak berkaitan dengan kata yang bersifat negatif seperti tutup, benerin, eror yang menunjukan bahwa pengguna tidak puas terhadap aplikasi Jamsostek *Mobile*. Selain itu kata layanan yang berhubungan dengan aplikasi dan jamsostek dapat dilihat pada Gambar 7

```
pre_w2v_model.wv.similarity('aplikasi', 'jmo')

0.88862735
```

Gambar 7. Similar Kata aplikasi dan jmo

Berdasarkan Gambar 7 similaritas antara kata aplikasi dan jmo mendapatkan nilai sebesar 0.88 atau 88%.

Long Short Term Memory

LSTM adalah jenis algoritma Jaringan Saraf Tiruan (*Artificial Neural Network*) yang dikembangkan khusus untuk mempelajari dan memproses data dengan urutan atau sekuensial, seperti data dalam bentuk teks, suara, atau *video*. Penerapan algoritma LSTM pada penelitian ini dapat dilihat pada Gambar 8.

```
In [31]: ##### Algoritma LSTM
def lstm_nn(embeddings, max_sequence_length, num_words, embedding_dim, labels_index):

    embedding_layer = Embedding(num_words,
                                embedding_dim,
                                weights=[embeddings],
                                input_length=max_sequence_length,
                                trainable=False)

    sequence_input = Input(shape=(max_sequence_length,), dtype='int32')
    embedded_sequences = embedding_layer(sequence_input)

    lstm = LSTM(256)(embedded_sequences)

    x = Dense(100, activation='relu')(lstm)
    x = Dropout(0.2)(x)
    preds = Dense(labels_index, activation='sigmoid')(x)

    model = Model(sequence_input, preds)
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['acc'])
    model.summary()
    return model
```

Gambar 8. Algoritma LSTM

Berdasarkan Gambar 8 algoritma LSTM menggunakan 100 *filter* dengan *activation* Relu, menggunakan *dropout* 0.2 yang berfungsi untuk mencegah terjadinya *overfitting*, menggunakan *activation sigmoid* pada *layer dense* yang berarti klasifikasi teks hanya memiliki dua label. Hasil pengujian pengujian dengan algoritma LSTM menggunakan 5 kali *splitting data* dengan *epoch* 10 disajikan pada Tabel 4

Tabel 4. Pengujian Algoritma LSTM

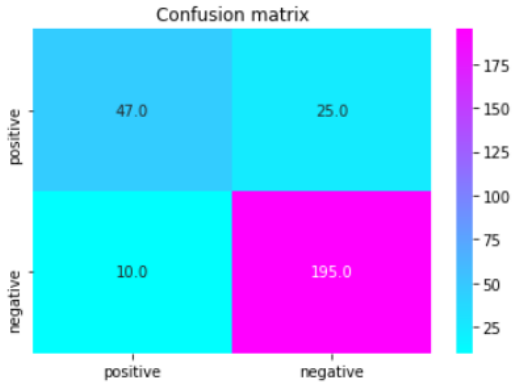
Splitting Data	Data Pelatihan	Data Uji	Epoch	Akurasi
50% training 50% testing	1385	1385	10	85,27%
60% training 40% testing	1662	1108	10	86,10%
70% training 30% testing	1939	831	10	85,08%
80% training 20% testing	2216	554	10	86,64%
90% training 10% testing	2493	277	10	87,36%

Berdasarkan Tabel 4 dilihat bahwa pada 90% data *training* dan 10% data *testing* menunjukkan tingkat akurasi terbaik. Pengujian dengan 10 *epoch* disajikan pada Gambar 9

```
Epoch 1/10
10/10 [==============================] - 17s 1s/step - loss: 0.5828 - acc: 0.7200 - val_loss: 0.5547 - val_acc: 0.8014
Epoch 2/10
10/10 [==============================] - 9s 886ms/step - loss: 0.5232 - acc: 0.7645 - val_loss: 0.4769 - val_acc: 0.8339
Epoch 3/10
10/10 [==============================] - 7s 703ms/step - loss: 0.4861 - acc: 0.7998 - val_loss: 0.4562 - val_acc: 0.8267
Epoch 4/10
10/10 [==============================] - 7s 714ms/step - loss: 0.4620 - acc: 0.8159 - val_loss: 0.4102 - val_acc: 0.8448
Epoch 5/10
10/10 [==============================] - 7s 708ms/step - loss: 0.4818 - acc: 0.7998 - val_loss: 0.4848 - val_acc: 0.7870
Epoch 6/10
10/10 [==============================] - 7s 688ms/step - loss: 0.5122 - acc: 0.7706 - val_loss: 0.4417 - val_acc: 0.8267
Epoch 7/10
10/10 [==============================] - 7s 683ms/step - loss: 0.4647 - acc: 0.8067 - val_loss: 0.4016 - val_acc: 0.8773
Epoch 8/10
10/10 [==============================] - 7s 680ms/step - loss: 0.4328 - acc: 0.8279 - val_loss: 0.4137 - val_acc: 0.8809
Epoch 9/10
10/10 [==============================] - 9s 937ms/step - loss: 0.4107 - acc: 0.8416 - val_loss: 0.3426 - val_acc: 0.8845
Epoch 10/10
10/10 [==============================] - 9s 915ms/step - loss: 0.3931 - acc: 0.8524 - val_loss: 0.3329 - val_acc: 0.8736
```

Gambar 9. Pengujian 10 Epoch

Setelah menguji dengan masing-masing *splitting* data maka didapatkan nilai akurasi terbaik pada *splitting* data 90% *training* 10% *testing*, hasil tersebut disajikan dalam *confusion matrix* yang dapat dilihat pada Gambar 10



Gambar 10. Hasil *Confusion Matrix*

Pada confusion matrix terdapat nilai akurasi, presisi, *recall* dan f1-*score*. Pehitungan dari masing-masing terminologi tersebut adalah sebagai berikut

1. Akurasi

$$akurasi = \frac{47 + 195}{47 + 195 + 10 + 25} = 0.8736$$

2. Presisi

$$positif = \frac{47}{47 + 10} = 0.82$$

$$negatif = \frac{195}{195 + 25} = 0.89$$

$$rata - rata\ presisi = \frac{0.82 + 0.89}{2} = 0.86$$

3. *Recall*

$$positif = \frac{47}{47 + 25} = 0.65$$

$$negatif = \frac{195}{195 + 10} = 0.95$$

$$rata - rata\ presisi = \frac{0.65\ +\ 0.95}{2} = 0.80$$

4. F1-*Score*

$$f1 - score = 2 * \frac{0.86 * 0.80}{0.86\ +\ 0.80} = 0.82$$

Berdasarkan perhitungan tersebut dapat disimpulkan nilai akurasi sebesar 87.36%, presisi 86%, recall 80% dan f1-*score* 82%. Hasil perhitungan tersebut dapat dibuktikan pada laporan klasifikasi yang disajikan pada Gambar 11

```
print(classification_report(y_test.argmax(axis=1), Y_pred.argmax(axis=1)))
```

	precision	recall	f1-score	support
0	0.82	0.65	0.73	72
1	0.89	0.95	0.92	205
accuracy			0.87	277
macro avg	0.86	0.80	0.82	277
weighted avg	0.87	0.87	0.87	277

Gambar 11. Laporan Klasifikasi

D. Simpulan

Hasil penelitian ini didapatkan bahwa model *deep learning* dapat digunakan untuk klasifikasi teks, dengan menggunakan algoritma LSTM mendapatkan nilai tertinggi pada *splitting* data 90% *training* 10% *testing* yaitu akurasi sebesar 87.36%, presisi 86%, recall 80% dan f1-*score* 82%. Berdasarkan penelitian sebelumnya hasil ini menunjukan bahwa metode *deep learning* memiliki kemampuan skalabilitas yang lebih baik dari machine learning, sehingga dapat bekerja pada jumlah data yang sangat besar dan memprosesnya dengan cepat. Hasil klasifikasi teks menunjukkan komentar negatif sebesar 80.58% dan positif sebesar 19.42% dari hasil tersebut pengguna aplikasi Jamsostek *Mobile* tidak puas dengan aplikasi, hasil ini diharapkan bisa menjadi evaluasi untuk pengembangan aplikasi Jamsostek *Mobile* agar lebih baik. Peneliti memberikan saran untuk peneliti selanjutnya dapat mengkombinasikan algoritma pada metode *deep learning* sehingga dapat memberikan hasil yang lebih baik dari penelitian ini.

E. Ucapan Terima Kasih

Ucapan terima kasih kepada STMIK AMIK Riau dan pihak BPJS Ketenagakerjaan yang sudah membantu dalam penelitian ini

F. Referensi

[1] P. T. Rimba and M. Mahkota, "2) 1,2," vol. 2, no. 11, pp. 3719–3724, 2022.

[2] H. Sutrisno, "Pengaruh bpjs ketenagakerjaan dalam meningkatkan kesejahteraan tenaga kerja," vol. 4, no. April, 2020.

[3] "PROGRAM STUDI KOMUNIKASI," 2022.

[4] T. Naraloka, "Pemanfaatan teknologi untuk menganalisa sentimen masyarakatdalam membantu peningkatan ekonomi kreatif di era new normal," vol. 1, no. 1, pp. 121–137, 2021.

[5] O. D. Instagram, "OPINION MINING POPULARITAS TEMPAT WISATA MENGGUNAKAN ALGORITMA SUPPORT VECTOR MACHINE (SVM) DAN PARTICLE SWARM," vol. 02, pp. 295–301, 2021.

[6] A. R. Maulana and N. Rochmawati, "Opinion Mining Terhadap Pemberitaan Corona di Instagram menggunakan Convolutional Neural Network," vol. 02, pp. 53–59, 2020.

[7] H. A. Pratiwi, M. Cahyanti, and M. Lamsani, "IMPLEMENTASI DEEP LEARNING FLOWER SCANNER MENGGUNAKAN," vol. 25, no. 1, pp. 124–130, 2021, doi: 10.46984/sebatik.v25i1.1297.

[8] H. Purnomo, H. Suyono, and R. N. Hasanah, "PERAMALAN BEBAN JANGKA PENDEK SISTEM KELISTRIKAN KOTA BATU MENGGUNAKAN DEEP LEARNING LONG SHORT-TERM MEMORY," no. 3, pp. 97–102, 2021.

[9] S. Zahara, Sugianto, and M. Bahril Ilmiddafiq, "Prediksi Indeks Harga Konsumen Menggunakan Metode Long Short Term Memory (LSTM) Berbasis Cloud Computing," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 3, no. 3, pp. 357–363, 2019, doi: 10.29207/resti.v3i3.1086.

[10] S. Zahara and Sugianto, "Peramalan Data Indeks Harga Konsumen Berbasis Time Series Multivariate Menggunakan Deep Learning," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 5, no. 1, pp. 24–30, 2021, doi: 10.29207/resti.v5i1.2562.

[11] L. Zaman, S. Sumpeno, and M. Hariadi, "Analisis Kinerja LSTM dan GRU sebagai Model Generatif untuk Tari Remo," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 8, no. 2, p. 142, 2019, doi: 10.22146/jnteti.v8i2.503.

[12] F. A. Lestari, L. Efrizoni, E. Ali, and R. Rahmiati, "Sistem Klasifikasi Pengaduan Masyarakat Pada BPJS Ketenagakerjaan Menggunakan Algoritma Naïve Bayes Berbasis Mobile," *Build. Informatics, Technol. Sci.*, vol. 4, no. 1, pp. 217–226, 2022, doi: 10.47065/bits.v4i1.1685.

[13] A. Al Kaafi, H. Rachmi, T. Informasi, F. Teknik, U. Bina, and S. Informatika, "Komparasi Algoritma Klasifikasi pada Analisis Sentimen Klaim Manfaat Jaminan Hari Tua Comparison of Classification Algorithms toward Sentiment Analysis on Claim of Old Age Guaranteed," vol. 12, pp. 79–88, 2023.

[14] T. Online, B. A. Aprian, Y. Azhar, and R. Setya, "Jurnal Politeknik Caltex Riau Prediksi Pendapatan Kargo Menggunakan Arsitektur Long Short Term Memory," vol. 6, no. 2, pp. 148–157, 2020.

[15] M. Rizki, S. Basuki, and Y. Azhar, "Implementasi Deep Learning Menggunakan Arsitektur Long Short Term Memory Untuk Prediksi Curah Hujan Kota Malang," vol. 2, no. 3, pp. 331–338, 2020.