Stance Classification Post Kesehatan di Media Sosial Dengan FastText *Embedding* dan *Deep Learning*

Ernest Lim, *Teknologi Informasi ISTTS*, Esther Irawati Setiawan, *Teknologi Informasi ISTTS*, dan Joan Santoso, *Teknologi Informasi ISTTS*

***Abstrak*— Misinformasi merupakan fenomena yang semakin sering terjadi di media sosial, tidak terkecuali Facebook, salah satu media sosial terbesar di Indonesia. Beberapa penelitian telah dilakukan mengenai teknik identifikasi dan klasifikasi *stance* di media sosial Indonesia. Akan tetapi, penggunaan Word2Vec sebagai *word embedding* dalam penelitian tersebut memiliki keterbatasan pada pengenalan kata baru. Hal ini menjadi dasar penggunaan fastText *embedding* dalam penelitian ini. Dengan menggunakan pendekatan *deep learning*, penelitian berfokus pada performa model dalam *klasifikasi stance* suatu judul *post* kesehatan di Facebook terhadap judul *post* lainnya. *Stance* berupa *for* (setuju), *observing* (netral), dan *against* (berlawanan). Dataset terdiri dari 3500 judul *post* yang terdiri dari 500 kalimat klaim dengan enam kalimat *stance* terhadap setiap klaim. Model dengan fastText pada penelitian ini mampu menghasilkan *F1 macro score* sebesar 64%.**

Keywords*—Bahasa Indonesia, *Deep Learning*, fastText, Media Sosial, *Stance Classification

I. PENDAHULUAN

Penyebaran informasi lewat media sosial memiliki sifat yang berbeda dari penyebaran di dunia nyata. Aspek komunikasi media sosial sangat dipengaruhi oleh fenomena *echo chamber* [1], [2]. Bentuk umum dari fenomena ini adalah kepercayaan *user* terhadap suatu informasi bila informasi tersebut sejalan dengan keyakinan politik *user* [3]. Dengan demikian, misinformasi ataupun *hoax* menyebar dengan cepat di sosial media, seperti Facebook [4]. Selain itu, aspek *bot* juga turut membantu penyebaran *hoax* secara signifikan [5]. Hal ini berdampak pada berbagai masalah penting seperti pemilu [6], penanganan konflik dan krisis [7], dan penanganan wabah [8].

Pada kasus Zika, rumor dibagikan tiga kali lebih banyak dibanding berita yang terverifikasi, meskipun sudah terdapat sosialisasi di media sosial. Kelemahan strategi komunikasi ini disebabkan oleh situasi yang selalu berkembang secara cepat sehingga banyak informasi yang beredar dan mengurangi visibilitas informasi kesehatan yang benar [9].

Oleh sebab itu, penting untuk dilakukan identifikasi dan klasifikasi *hoax*, terutama di bidang kesehatan. Pendekatan dalam klasifikasi *hoax* pada umumnya hanya klasifikasi konten. Akan tetapi, media sosial memberikan peluang untuk meneliti *stance classification* dengan banyaknya data yang tersedia, seperti profil *user* hingga interaksi pada *post*.

Stance classification adalah teknik untuk mengetahui secara otomatis pandangan penulis teks terhadap suatu *statement*, apakah mendukung, menentang, atau netral [10]. Proses *stance classification* secara umum dapat dibagi menjadi dua tahap, yaitu: pembentukan vektor kata dan klasifikasi *stance*. Dalam pembentukan vektor kata, banyak penelitian *stance classification* yang menggunakan Word2Vec untuk *embedding*. Popularitas Word2Vec ini dikarenakan kemampuannya dalam mengetahui similaritas makna antar kata. Informasi similaritas ini didapatkan dengan memperhatikan kesamaan kata-kata di sekitar kata target [11].

Pada tahap klasifikasi, *Convolutional Neural Network* (CNN) termasuk salah satu *classifier* yang populer untuk *stance classification*. CNN pertama kali digunakan oleh Collobert untuk mengambil fitur penting secara otomatis [12]. Informasi hirarki fitur ini didapatkan melalui operasi *max-over-time pooling* yang dikembangkan lebih lanjut oleh Kim untuk klasifikasi kalimat [13]. Namun, banyak juga metode klasifikasi *stance* yang menggunakan LSTM. Tren ini disebabkan oleh kemampuan LSTM dalam menyimpan informasi sekuens kata dalam kalimat [14].

Dalam konteks bahasa Indonesia, penelitian *stance classification* pada umumnya menggunakan Word2Vec, seperti pada penelitian [15] dan *classifier* LSTM [16]. Hal ini dapat menyebabkan hasil klasifikasi yang kurang akurat karena faktor kesalahan pengetikan. Oleh karena itu, peneliti mengusulkan penggunaan fastText saat pembentukan vektor kata dan kalimat dan menggunakan *deep learning*, terutama Bi-LSTM, untuk menghitung probabilitas stance.

Usulan tersebut mengacu pada penelitian [16] yang mampu menghasilkan model yang cukup akurat dengan menggunakan Word2Vec dan *classifier* LSTM. Pemilihan fastText sendiri berdasarkan kepada kelebihan fastText saat membaca kata. Dengan *n-gram character*, sebuah kata akan dibaca per karakter sesuai nilai n-gram tersebut [17]. Dengan demikian, permasalahan *Out-of-Vocabulary words* (OOV) dapat teratasi. Bi-LSTM sendiri dipilih karena terbukti lebih akurat dan tidak serentan LSTM untuk *overfitting* [18]. Peneliti juga memilih CNN karena proses pembelajaran yang membutuhkan relatif sedikit *epoch* dan

Ernest Lim, Teknologi Informasi, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Jawa Timur, Indonesia (e-mail: ernest1@mhs.stts.edu)
Esther Irawati Setiawan, Teknologi Informasi, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Jawa Timur, Indonesia (e-mail: esther@stts.edu)
Joan Santoso, Teknologi Informasi, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Jawa Timur, Indonesia (e-mail: joan@stts.edu)

mampu menghasilkan akurasi yang tinggi [19] dan LSTM karena selain menjadi dasar perbandingan dengan penelitian Setiawan [16], LSTM terbukti tepat untuk permasalahan teks yang berhubungan dengan sekuens [20].

Tujuan dari penelitian ini adalah menemukan pendekatan dan parameter *stance classification* yang efektif untuk membantu klarifikasi *hoax* di media sosial. Secara khusus, peneliti mencari kombinasi metode dan pengaturan parameter dalam penggunaan fastText dan *deep learning* untuk klasifikasi *stance* yang paling akurat.

II. Penelitian Terkait

Penelitian mengenai *stance*, seperti terlihat pada Tabel I, banyak yang menggunakan *word embedding* Word2Vec dan *classifier* CNN dan LSTM.

Tabel I
Penelitian Stance Detection dan Classification

Topik	Riset	Pendekatan	Hasil
Stance Classification	Chen, 2016	Pemodelan CNN + fitur akun media sosial (jumlah *like*, *comment*)	Model memberikan hasil akurasi tertinggi dibanding tanpa informasi tambahan dari medsos
Stance Classification	Kochkina , 2017	Pemodelan kalimat rata-rata W2V + jumlah kata negatif + *classifier* LSTM	Pendekatan memenangkan SemEval 2017 Task 8 Sub-task A
Stance Classification	Lozano, 2017	*POS Tagging* W2V + fitur akun Twitter + *handwritten rules* + *classifier* CNN	Peringkat 4 dalam SemEval 2017 RumourEval 8A, akurasi 74,9%
Stance Detection	Shalini, 2019	GloVe / W2V + *classifier* BoT / CNN / Bi-LSTM	Bi-LSTM + BoT emb memberi performa tertinggi
Stance Classification	Jannati, 2019	Pemodelan kalimat rata-rata W2V + SVM	*F1 macro* 63,54%
Stance Classification	Setiawan, 2020	Pemodelan kalimat rata-rata W2V / LSTM W2V + *classifier* LSTM	Pendekatan *full* LSTM memberi *F1 macro* tertinggi

Pada penelitian Chen [21], *tweet* yang dikumpulkan berhasil diklasifikasi dengan akurasi tertinggi dengan menambahkan data media sosial pada input dengan *classifier* CNN. Sementara itu, penelitian Kochkina [22] melakukan *stance classification* pada *tweet* dengan menghitung jumlah kata negatif pada *tweet* dan pemodelan kalimat dengan pendekatan rata-rata. *Word embedding* yang digunakan adalah model Word2Vec *pre-trained* dengan *Google News dataset*. *Classifier* yang digunakan adalah *branch-LSTM*. Pendekatan ini memenangkan kompetisi RumourEval dengan akurasi 78,4%.

Pendekatan yang mirip untuk mengklasifikasi *stance* di media sosial juga dilakukan oleh Lozano [19]. Namun, berbeda dengan Kochkina, Lozano menggunakan *classifier* CNN. Pendekatan ini memperoleh peringkat keempat dengan akurasi 74,9%. Lozano berargumen CNN memberikan hasil yang akurat pada jumlah data yang kecil tapi juga berisiko *overfitting* secara cepat.

Sementara itu, Shalini [23] mendapat performa model tertinggi dengan Bi-LSTM dan Bag-of-Tricks *embedding* dengan *F1 score* sebesar 79,84%. Model digunakan untuk mendeteksi *stance* suatu *tweet* apakah mendukung atau menolak suatu topik.

Dalam konteks bahasa Indonesia, penelitian yang paling dekat yang juga melakukan *stance classification* antara dua kalimat judul adalah penelitian Setiawan [16]. Penelitian tersebut menunjukkan *stance classification* dapat dilakukan dengan pemodelan kalimat dengan LSTM dan diproses ke *classifier* LSTM dengan *F1 macro* 71%. Dalam penelitian tersebut dilakukan pula uji coba pemodelan kalimat dengan pendekatan rata-rata dan *classifier* LSTM yang menghasilkan *F1 macro* sebesar 60,66%.

Penulis memosisikan penelitian ini sebagai pengembangan penelitian *stance classification* berbahasa Indonesia, terutama dari penelitian [16]. Pengembangan ini dilakukan dengan mengubah word embedding menjadi fastText dan menggabungkan model dan fitur-fitur dari penelitian Setiawan [16] dengan pendekatan kalimat pada penelitian Kochkina [22] yang mampu menghasilkan akurasi lebih tinggi. Mengingat penelitian ini meneliti *post* media sosial, maka diambil juga aspek pengambilan data *post* seperti jumlah *like* dan *comment* seperti pada penelitian [21] dan [19].

A. *Stance Classification*

Penelitian *stance* pada awalnya banyak dilakukan pada bidang-bidang dilluar topik penyebaran *hoax*, seperti debat kongres [24] dan forum debat online [25]. Namun, pada tahun 2017, diperkenalkan kompetisi *Fake News Challenge* untuk mengidentifikasi *hoax*. Fokus dari kompetisi tersebut adalah mencari *stance* isi berita dengan judul dari artikel berita tersebut. Pada *Fake News Challenge*, pendekatan yang memiliki akurasi tertinggi adalah pendekatan *multi-layer perceptrons* [26] dan CNN dengan *decision tree*. Di Indonesia, salah satu penelitian terbaru mengenai *stance classification* adalah analisis pendapat terhadap berita kesehatan [16]. Dalam paper tersebut, LSTM digunakan untuk mencari vektor kalimat dan sebagai *classifier stance*.

Penelitian ini dirancang sebagai pengembangan dari penelitian analisis pendapat berita kesehatan dalam konteks media sosial. Penelitian ini juga mengeksplorasi model pembentukan vektor kalimat pendekatan sederhana rata-rata dengan *classifier* berbasis *deep learning*. Metode *deep learning* yang dipilih adalah CNN, LSTM, dan Bi-LSTM.

B. *NLP dengan CNN*

Penggunaan *Convolutional Neural Network* (CNN) untuk penelitian NLP dilakukan pertama kali di tahun 2011 [12]. Konsep CNN dengan lapis *max-pooling* dikembangkan dengan membuat dua kelompok vektor kata dimana setiap kelompok dianggap sebagai *channel*. Dengan *backpropagation* hanya pada satu *channel*, model dapat melatih satu kelompok vektor sesuai penugasan yang ada. Hal ini juga membuat kelompok vektor yang lain statis [13]. Aplikasi CNN untuk *stance classification* sendiri dapat

digabung dengan *automatic rule mining, hand written rules*, dan *voting* [19]. Untuk konteks media sosial, penggabungan tersebut adalah model CNN dengan informasi *user* dan *post* [21].

C. NLP dengan LSTM

LSTM sendiri merupakan pengembangan dari RNN sederhana. Bedanya, LSTM memiliki *forget gate* yang Dalam penugasan *stance*, LSTM dapat mengatasi permasalahan kalimat panjang yang dihadapi oleh CNN. Pada penelitian Hanselowski [27], model baru dengan penambahan *stacked LSTM* mampu menghasilkan peningkatan akurasi klasifikasi *Fake News Challenge*.

D. NLP dengan Bi-LSTM

Bi-LSTM telah diaplikasikan dalam *stance detection* seperti pada SemEval 2016 Task 6 dalam konteks Twitter [28]. Pada penelitian Habernalt [29], *bidirectional LSTM* terbukti lebih akurat dalam membandingkan makna kalimat satu dengan yang lain terhadap sebuah topik. Mrowca [30] menggunakan Bi-LSTM untuk mengambil informasi fitur kalimat judul berita dan teks dari isi berita tersebut dalam *Fake News Challenge*. Klasifikasi sendiri dilakukan dengan menggabungkan *output* dari Bi-LSTM dan vektor fitur.

E. Word dan Sentence Embedding

Word embedding berbasis *neural network* dimulai pada tahun 2003 dengan *feedforward neural network model* pada penelitian Bengio [31]. Untuk mengurangi kompleksitas komputasional, Mikolov memperkenalkan Word2Vec yang juga mampu mengekstrak similaritas antar vektor kata dengan *cosine similarity* [11].

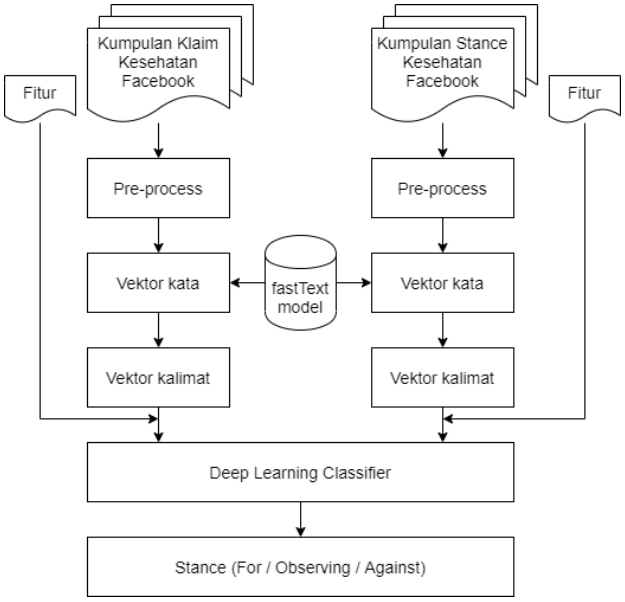
FastText [17] adalah hasil pengembangan dari Word2Vec. Berbeda dari Word2Vec, fastText tidak memakai hanya satu kata secara utuh untuk diproses, tapi fastText menggunakan n-gram. Contoh aplikasi n-gram pada kata “program” dengan trigram (n=3) berupa “pro”, “rog”, “ogr”, “gra”, “ram”. Kelebihan fastText adalah waktu proses yang relatif cepat. FastText terbukti efektif dalam klasifikasi teks seperti *sentiment analysis* [32]. Kombinasi CNN dan fastText juga terbukti lebih akurat dalam klasifikasi sentimen dibandingkan menggunakan Word2Vec [33].

Pada bidang *sentence embedding*, Kiros memperkenalkan *Skip-thought vector*, dimana sebuah kalimat akan ditebak berdasarkan kalimat disekitarnya, seperti konsep *Skip-gram* [34]. Permasalahan durasi pelatihan yang lama dijawab dengan *Quick-thought vectors* [35]. Penurunan waktu pelatihan ini dikarenakan penentuan kalimat berikutnya yang menggunakan *classifier*, bukan *encoder-decoder* RNN seperti *Skip-thought*.

Namun, pembentukan vektor kalimat dengan metode sederhana seperti rata-rata dari *word embedding* juga dapat menghasilkan performa yang bagus [36]. Dalam penelitian Wieting [37], model *paragram-phrase* yang dibentuk dengan rata-rata dari vektor kata mampu mengalahkan model LSTM dalam tugas pengenalan similaritas teks.

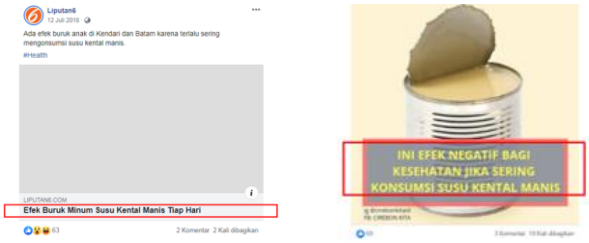
III. METODOLOGI PENELITIAN

Proses penelitian ini dimulai dari pengumpulan data, pembentukan vektor kata dan kalimat, pengambilan fitur kalimat dan *post*, dan *deep learning classifier* untuk klasifikasi *stance*. Gambar 1 menjabarkan alur sistem penelitian ini.



Gambar 1. Alur Sistem Penelitian

Penelitian ini sendiri akan dibatasi dengan hanya mengambil data dari *post* Facebook bertopik kesehatan. Peneliti hanya akan mengambil judul artikel yang *dishare*, judul pada gambar di *post*, atau teks judul pada *post* seperti pada Gambar 2. Panjang dari teks tersebut adalah satu kalimat atau lebih. Klasifikasi akan dilakukan hanya dengan CNN, LSTM, dan Bi-LSTM. Kelas klasifikasi sendiri terdiri dari tiga *stance*, yaitu: *for*, *observing* dan *against*.



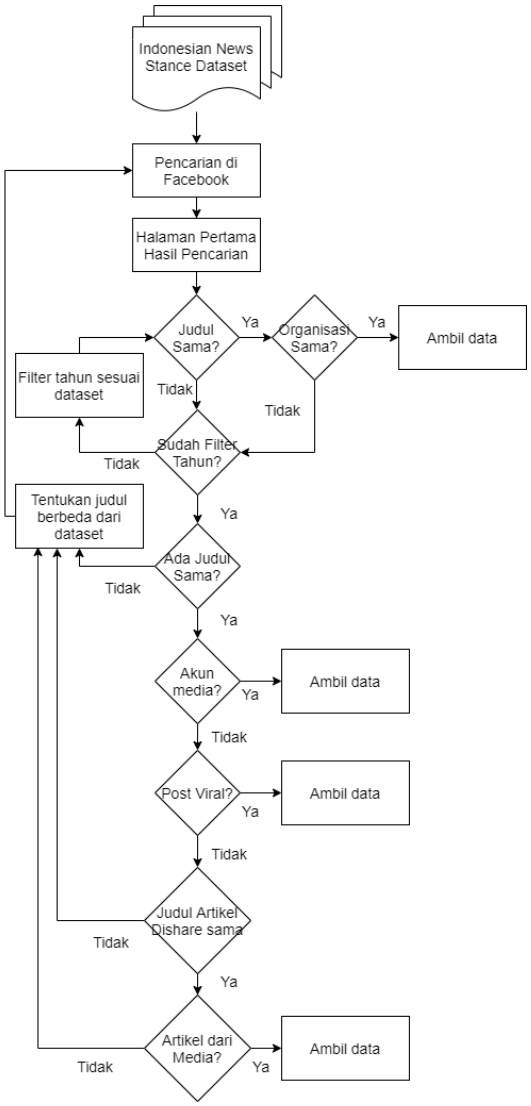
Gambar 2. Contoh Pengambilan Teks dan *Post* (Kotak Merah)

A. Dataset

Kalimat klaim maupun *stance* didasarkan pada dataset *Indonesian News Stance Dataset* [16]. Dalam penelitian ini, peneliti mencari *post* klaim dan *post* reaksi di Facebook berdasar kalimat klaim dan kalimat-kalimat reaksinya. Saat tidak ditemukan judul *post* Facebook yang berisi teks-teks tersebut, data yang diambil berupa judul-judul *post* Facebook dengan topik lain yang masih termasuk dalam bidang kesehatan. Hal ini mengakibatkan perubahan 350 kalimat klaim (70%) dari total 500 kalimat klaim di Indonesian News Dataset. Peneliti juga menambahkan informasi jumlah *like* dan *comment* pada setiap *post*.

Pengambilan data dilakukan secara manual di Facebook. Data yang dicari adalah judul artikel/judul gambar/judul *post* Facebook bersifat klaim, tanggal post klaim, URL *post* klaim, jumlah *like* pada *post*, dan jumlah *comment*. Data

yang dicari pada *post stance* sama dengan *post* klaim dengan penambahan data *stance*. Penentuan klaim dan *stance* dilakukan berdasar opini pengambil data. Data disimpan dalam bentuk file .CSV dan dipisah menjadi kolom-kolom tersendiri. Proses pengambilan data ini dapat diilustrasikan sebagai *flowchart* dalam gambar 3 berikut ini.



Gambar 3. Alur Sistem Pemilihan Post untuk Dataset

Proses pengambilan data *post* klaim dan *post* terkait beserta *stance* diawali dengan pencarian di kotak pencarian Facebook. *Query* pencarian adalah kalimat judul pada *Indonesian News Stance Dataset* dengan nama organisasi pembuat artikel tersebut. Contoh, kalimat pencarian "liputan6 Efek Buruk Minum Susu Kental Manis Tiap Hari" berasal dari artikel liputan6 yang berjudul "Efek Buruk Minum Susu Kental Manis Tiap Hari". Bila ditemukan *post* berjudul artikel tersebut dari akun organisasi yang sama, maka akan diketik data yang dibutuhkan ke dalam file .CSV.

Bila tidak ditemukan *post* dengan kalimat tersebut pada halaman hasil pertama, pengambil data akan melakukan ulang pencarian dengan pemilihan *filter* tahun *post* sesuai dengan data tahun *post* pada *Indonesian News Stance Dataset*. Bila masih tidak ditemukan *post* dengan judul dan organisasi yang sama, maka dapat dilakukan satu dari dua solusi berikut: dilakukan pengambilan *post* dengan judul yang sama tapi dengan organisasi atau *user* yang berbeda, atau pengambil data mencari topik kesehatan baru di kotak pencarian di Facebook.

Pengambil data memilih *post* hasil pencarian bertopik baru dengan prioritas utama *post* milik akun organisasi media, mulai dari media berskala nasional, hingga daerah. Hal ini dilakukan untuk memperbesar kemungkinan mendapat *post* dengan reaksi besar. Prioritas kedua adalah *post* perorangan yang *viral*, terlihat dari jumlah *like* yang bisa mencapai satu juta lebih. Prioritas ketiga adalah *post* berjudul sama dengan kalimat topik pencarian milik akun perorangan. Berikut ini, pada Tabel II, adalah salah satu contoh pasangan judul klaim dan judul berkaitan yang didapat.

TABEL II
CONTOH KLAIM DAN STANCE

Judul Klaim	**Judul Reaksi**	**Stance**
Dua Studi Ini Buktikan Manfaat Rokok Elektrik untuk Berhenti Merokok	Manfaat Vaping untuk Kesehatan yang Perlu Anda Ketahui	for
	Riset tiga tahun dari Italia pastikan rokok elektrik tidak bahaya	for
	Vape Alias Rokok Elektrik, Apakah Bahaya Bagi Tubuh?	observing
	Kelebihan dan Bahaya Rokok Elektrik untuk Kamu yang Masih Mahasiswa!	observing
	5 Bahaya yang Mengintai di Balik Nikmatnya Vape	against
	Bahaya dan Efek Samping Vape yang Negatif Mengintai Para Penggunanya	against

Secara total, peneliti mendapat 3.500 judul *post* Facebook bertopik kesehatan yang berbahasa Indonesia. Dari data tersebut, 500 kalimat adalah kalimat klaim. Setiap kalimat klaim memiliki beberapa kalimat reaksi, yaitu: dua kalimat *stance for* (setuju dengan kalimat klaim), dua kalimat *stance observing* (netral terhadap kalimat klaim), dan dua kalimat *against* (bertentangan dengan kalimat klaim) yang juga diambil dan diberi label secara manual. Distribusi data dapat dilihat pada Tabel III.

TABEL III
DISTRIBUSI DATA DALAM DATASET PENELITIAN

Jenis Data	**Jumlah**
Klaim	500
For	1000
Observing	1000
Against	1000
Total	3500

B. Pembentukan Vektor Kata dan Kalimat

Embedding kata dan kalimat dilakukan untuk memetakan nilai vektor berdasarkan *embedding* fastText. Sebelum dipetakan, setiap kalimat akan melalui *pre-processing* terlebih dahulu.

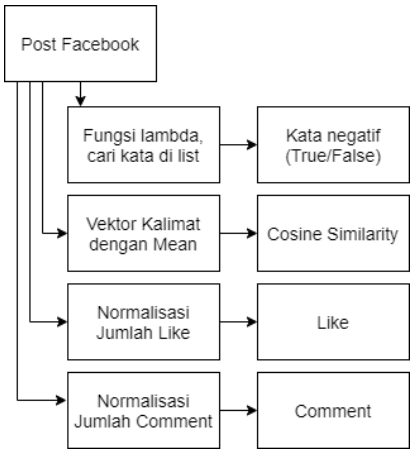
Pre-processing dimulai dengan *case folding*, tokenisasi dengan *word_tokenize* dari nltk, dan *filtering* untuk membuang kata-kata tidak penting seperti "yang", "untuk", "pada", "ke" dan lain-lain berdasar *stopwords* dari Tala [38]. Karakter spesial juga akan dihapus kecuali tanda tanya, tanda hubung, dan 0-9.

Penelitian ini menggunakan *pre-trained* fastText dengan dimensi 300. Dalam setiap kalimat, *token* kata akan diubah

menjadi vektor berdimensi 300 berdasar vektor *pre-trained* fastText cc.id.300.bin.

Semua kalimat akan melalui proses *padding* dengan vektor 0 sehingga semua kalimat akan memiliki besar matriks yang sama dengan kalimat dengan jumlah kata terbanyak. Setelah itu, vektor kalimat klaim dibentuk dengan menghitung rata-rata dari setiap vektor kata dalam kalimat.

C. Pengambilan Fitur Kalimat dan Post



Gambar 4. Pembentukan Fitur

Pembentukan fitur secara garis besar dapat dilihat pada gambar 4. Pengambilan fitur kata negatif dan *cosine similarity* dilakukan pada setiap kalimat klaim dan *stance*. Pada aspek fitur kata negatif, peneliti menambah data Boolean mengenai ada atau tidaknya kata-kata negatif dalam kalimat klaim dan *stance*. Kata-kata negatif tersebut adalah 'tidak', 'bukan', 'jangan', 'tak', 'belum', 'mitos', 'hoaks', dan 'hoax'. Penentuan kata-kata negatif berdasar pada penelitian sebelumnya [16], dimana setiap vektor kalimat akan diperpanjang dengan menambahkan nilai Boolean tadi.

Cosine similarity digunakan untuk mengetahui kemiripan vektor dengan mengukur sudut kosinus dari kedua vektor tersebut. *Cosine similarity* akan menghasilkan nilai 0 hingga 1, dimana hasil yang semakin dekat dengan 1 menunjukkan dua vektor yang memiliki banyak kemiripan. Terlihat pada rumus (1), *cosine similarity* memiliki dua komponen yang dalam penelitian ini adalah kalimat klaim dan kalimat *stance*.

$$cosine\ similarity = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

Pada aspek fitur *post*, peneliti mengambil informasi jumlah *like* dan jumlah komentar setiap *post*. Setelah dinormalisasi, jumlah *like* dan jumlah komentar akan ditambahkan ke vektor kalimat.

D. Fitur Kalimat dan Fitur Media Sosial

Dalam penelitian *stance classification*, Chen [21] memanfaatkan berbagai *metadata* untuk membantu mengetahui *stance* dari *tweet*. Informasi *user* adalah data yang paling penting, diikuti oleh informasi topik dan komentar. Pada penelitian Lozano [19], *metadata* yang diambil adalah terbagi menjadi empat kriteria, yaitu: 1) jumlah tanda baca dan kata negasi pada *tweet*, 2) jumlah status, *follower*, dan *tweet* favorit *user*, 3) status verifikasi *user*, 4) apakah *tweet* termasuk *tweet reply* dan seberapa panjang thread *tweet* tersebut. Selain itu, Lozano juga menerapkan beberapa aturan dalam proses pelabelan *stance* pada *tweet*. Aturan pertama adalah pemberian label *deny* bila *tweet* mengandung frasa '*not believable*'. Aturan kedua adalah pemberian label *support* bila terdapat URL dan tidak mengandung *@-mention*.

Pada penelitian ini, dalam konteks Facebook, peneliti mengambil *metadata* ada atau tidaknya tanda tanya, jumlah *like*, dan jumlah *comment*. Peneliti juga menentukan beberapa kata negatif yang diasosiasikan dengan *stance against*.

E. Parameter Penting dalam Classifier

Beberapa penelitian tentang parameter penting pada CNN [39] dan LSTM [36]. Pada CNN, ada beberapa parameter yang harus diperhatikan, yaitu: 1) penggunaan representasi vektor kata, 2) *filter region size* yang perlu disesuaikan terhadap penugasannya, 3) peningkatan nilai *feature maps* akan meningkatkan waktu pelatihan model, 4) *1-max pooling* memberikan performa terbaik, dan 5) regularisasi tidak berdampak signifikan terhadap performa.

Pada LSTM, akurasi LSTM sangat dipengaruhi oleh jumlah *token*, pengaturan dimensi disesuaikan dengan tujuan, dan penambahan *hidden unit* yang terkadang justru memperburuk akurasi.

F. Deep Learning Classifier

- Model Classifier CNN

Pemrosesan vektor dengan CNN pada algoritme 7.1 dilakukan dengan beberapa tahap. Pertama, vektor kalimat klaim akan diproses dengan filter dari CNN. Ukuran dari filter ini ditentukan pada saat pembuatan lapis convolutional 1D. Dalam kasus teks, nilai kernel menentukan jumlah baris filter, dan nilai *filters* menjadi panjang filter. Penelitian ini menggunakan kernel_size = 1 sehingga filter memproses per 1 kalimat. Setelah menentukan ukuran filter, perhitungan jumlah dari produk per elemen. Dari hasil tersebut, diambil nilai maksimal dengan *1-max pooling*. Program kemudian mengubah *shape* dengan *flatten* sehingga matriks hasil pooling menjadi satu vektor. Vektor tersebut kemudian diproses dengan lapis *dense* untuk mendapat dimensi yang sesuai dengan panjang *one-hot label* dan *softmax* untuk mencari probabilitas *stance* tertinggi pada matriks pasangan vektor tadi.

Parameter *classifier* CNN yang digunakan adalah sebagai berikut: 1) lapis *convolutional 1D* dengan filters=304, kernel_size=1, aktivasi ReLU, 2) lapis *dropout* sebesar 0,5 , 3) lapis *max pooling 1D* dengan *pool size* sebesar satu, 4) lapis *flatten*, 5) lapis *dense* dengan output 50 unit dan aktivasi ReLU, dan 6) lapis *dense* output 3 unit dengan *softmax* untuk mengetahui probabilitas *stance*. Model dibentuk dengan *optimizer* Adam dengan *learning rate* sebesar 0,001.

- Model Classifier LSTM

Pemrosesan vektor dengan LSTM dilakukan dengan menginput vektor gabungan klaim ke dalam lapis LSTM ke

dalam *input gate*. LSTM memiliki sistem *gate* untuk menangani masalah *vanishing gradient*. Sistem proses LSTM terdiri dari lima modul, yaitu: *input gate*, *new memory cell*, *forget gate*, *final memory generation*, dan *output gate*. Dalam *input gate*, program membuat *memory* baru dengan menghitung vektor *input* tadi dengan *activation function* dan melihat *hidden state* dari LSTM sebelumnya, yang saat ini masih tidak ada. Sementara itu, *forget gate* menilai apakah *cell* sebelumnya diperlukan untuk perhitungan cell sekarang dengan melihat vektor *input* dan *hidden state* sebelumnya. Nilai dari kedua proses sebelumnya kemudian dijumlahkan dalam final *memory cell* (Ct). Setelah itu, *output gate* menentukan seberapa besar *memory* Ct yang perlu digunakan dalam langkah waktu saat ini. Nilai *output* dari *cell* ini adalah hasil perkalian dari *output gate* dan Ct yang telah melalui lapis *activation function*. Proses ini diulang Kembali untuk *input* vektor gabungan judul yang berkaitan dengan klaim pada langkah waktu berikutnya dengan *hidden state* dari proses dengan vektor gabungan klaim tadi. *Output* dari *cell* ini yang akan diproses ke lapis *dense* dan lapis *output softmax* untuk mendapat *stance*.

- Model Classifier Bi-LSTM

Bi-LSTM memiliki alur yang mirip dengan LSTM, hanya bedanya terdapat satu rangkaian LSTM tambahan yang menghitung dari input paling terkahir hingga input paling awal (*backward* LSTM). Sehingga terdapat dua rangkaian LSTM, *forward* LSTM dan *backward* LSTM. Algoritme classifier Bi-LSTM dapat dilihat pada algoritme 7.3. Langkah pertama dari algoritme, vektor gabungan klaim akan diinputkan terlebih dahulu ke dalam lapis forward LSTM. Langkah-langkah ini sama dengan langkah 01-10 pada algoritme 7.2. Program juga akan menjalankan *backward* LSTM. Input langkah waktu pertama adalah vektor terakhir, yaitu vektor gabungan judul berkaitan. Vektor ini diproses Dalam *input gate*, program membuat *memory* baru dengan menghitung vektor *input* tadi dengan *activation function* dan melihat *hidden state* dari LSTM sebelumnya, yang saat ini masih tidak ada. Sementara itu, *forget gate* menilai apakah *cell* sebelumnya diperlukan untuk perhitungan cell sekarang dengan melihat vektor *input* dan *hidden state* sebelumnya. Nilai dari kedua proses sebelumnya kemudian dijumlahkan dalam final *memory cell* (Ct). Setelah itu, *output gate* menentukan seberapa besar *memory* Ct yang perlu digunakan dalam langkah waktu saat ini. Nilai *output* dari *cell* ini adalah hasil perkalian dari *output gate* dan Ct yang telah melalui lapis *activation function*. Proses ini diulang untuk *input* vektor gabungan klaim dengan *hidden state* dari proses dengan vektor gabungan judul berkaitan tadi. *Output* dari *cell backward* LSTM ini yang akan diproses ke lapis *dense* dan lapis *output softmax* untuk mendapat *stance*, bersamaan dengan output dari cell *forward* LSTM.

Pada *classifier* LSTM dan Bi-LSTM, pengaturan parameter yang digunakan adalah: 1) lapis LSTM/Bi-LSTM dengan dua langkah waktu, *input* dengan dimensi 304 dan *recurrent dropout* sebesar 0.5, dan *output* 200 unit 2) lapis *dense* dengan aktivasi ReLU dan *output* 100 unit, dan 3) lapis *dense output* tiga unit dengan aktivasi *softmax*. Model LSTM dan Bi-LSTM ini juga dibentuk dengan *optimizer* Adam dengan *learning rate default* sebesar 0,001.

G. *Sistem Evaluasi Performa*

Performa model dilihat berdasarkan *F1 score* setiap kelas *stance* dan *F1 macro*. *F1 score* diperoleh dengan menghitung *harmonic mean* dari *precision* dan *recall*, seperti pada rumus 2. Nilai *F1 score* memiliki rentang 0 hingga 1 dimana nilai 1 menunjukkan akurasi maksimal.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall} \quad (2)$$

Precision, dalam konteks klasifikasi, pada rumus 8.2, adalah nilai dari pembagian *true positive* (tp) dengan jumlah dari *true positive* (tp) dan *false positive* (fp). Rumus 3 berikut menunjukkan rumus *precision*.

$$Precision = \frac{tp}{tp + fp} \quad (3)$$

Sementara itu, recall, dalam konteks klasifikasi, adalah nilai dari pembagian *true positive* (tp) dengan jumlah dari *true positive* (tp) dan *false negative* (fn). Rumus 4 berikut menunjukkan rumus *recall*.

$$Recall = \frac{tp}{tp + fn} \quad (3)$$

Istilah *true positive* (tp), *true negative* (tn), *false positive* (fp), dan *false negative* (fn) dapat dijabarkan sebagai berikut.

Istilah *true* dan *false* dalam rumus-rumus tersebut mengacu pada kebenaran sebuah prediksi. Sebagai contoh, *classifier* memprediksi sebuah judul memiliki *stance for* terhadap sebuah klaim. Bila label judul tersebut (data Y) pada *dataset* adalah *for*, maka hasil prediksi bersifat *true*. Bila data Y pada *dataset* adalah *observing* atau *against*, maka hasil prediksi bersifat *false*.

Sementara itu, istilah *positive* dan *negative* mengacu pada hasil prediksi yang diharapkan. Contoh, saat menghitung *precision stance observing*, maka semua prediksi yang menghasilkan *stance observing* masuk dalam kategori *positive* dan sisanya, prediksi dengan hasil *stance for* dan *observing*, sebagai anggota kategori *negative*.

Penelitian ini menggunakan fungsi classification_report dari sklearn.metrics untuk mendapat tabel mengenai performa model. Aspek-aspek yang terdapat dalam tabel tersebut adalah nilai *precision*, nilai *recall*, *F1 score*, dan *macro average F1 score*.

IV. ANALISIS DAN PEMBAHASAN

Untuk mendapatkan arsitektur yang efektif dalam *stance classification*, peneliti menguji coba model dengan beberapa parameter dan metode yang berbeda. Dalam eksperimen, data uji coba adalah 20% dari jumlah data yang ada. Peneliti menggunakan *macro-average F1-score* untuk mengetahui akurasi prediksi semua label dan *F1-score* untuk mengetahui akurasi prediksi pada setiap label.

Dalam setiap uji coba, program akan menghitung *F1 score* dan *F1 macro* dari sepuluh *seed* dari model, sehingga terdapat 10 *classification report* dalam setiap uji coba. Hasil

uji coba adalah rata-rata *F1 score* dan *F1 macro* dari sepuluh *classification report* tersebut.

Aspek-aspek yang diuji cobakan dalam eksperimen ini, yaitu: 1) jenis *word embedding*, 2) pengaruh hasil pelatihan *word embedding*, 3) jenis *classifier*, 4) fitur kalimat dan media sosial, 5) *dropout* dan jenis *activation*, dan 6) *hidden units.*

A. *Jenis Word Embedding*

Pada uji coba pengaruh jenis *word embedding*, peneliti membandingkan performa antara Word2Vec menggunakan vektor *pre-trained* Kyubyong yang berdimensi 300 (https://github.com/Kyubyong/wordvectors) dan fastText cc.id.300.bin yang juga berdimensi 300. Penentuan Word2Vec sebagai perbandingan mengacu pada penelitian sebelumnya yang menggunakan Word2Vec untuk mengklasifikasi pendapat di Twitter dengan metode pemodelan kalimat yang sama dengan penelitian ini [22].

Pada Tabel IV terlihat bahwa ada perbedaan akurasi sebesar 1,1% antara rata-rata *F1 macro* model dengan Word2Vec (52,7%) terhadap model dengan fastText (53,8%). Penggunaan Word2Vec paling akurat dengan LSTM hanya mencapai 55% dan penggunaan fastText paling akurat diperoleh dengan CNN dengan *F1 macro* 55,4%.

TABEL IV
HASIL UJI COBA JENIS WORD EMBEDDING

Pendekatan	*F1 for*	*F1 observing*	*F1 against*	*F1 Macro*
Word2Vec + CNN	0.454	0.654	0.488	0.532
Word2Vec + LSTM	0.503	0.659	0.488	0.550
Word2Vec + Bi-LSTM	0.453	0.624	0.424	0.500
Rata-rata	**0.470**	**0.646**	**0.467**	**0.527**
fastText + CNN	**0.495**	**0.824**	0.345	**0.554**
fastText + LSTM	0.492	0.801	0.294	0.529
fastText + Bi-LSTM	0.441	0.817	**0.391**	0.530
Rata-rata	**0.476**	**0.814**	**0.343**	**0.538**

B. *Pemilihan Hasil Pelatihan Word Embedding*

Terlilhat pada Tabel V bahwa penggunaan vektor *pre-trained* harus disesuaikan dengan banyaknya fitur yang dipakai dalam penelitian. Bila hanya menggunakan vektor kalimat sebagai *input*, maka wiki.id.bin akan lebih akurat (54,7%) dibandingkan dengan cc.id.300.bin (53,8%). Namun, bila ada penambahan vektor fitur, seperti informasi kata negatif, *cosine similarity*, jumlah *like* dan *comment*, cc.id.300.bin akan menghasilkan model yang lebih akurat dengan *F1 macro* sebesar 60% dibandingkan dengan 50% dengan wiki.id.bin.

TABEL V
HASIL UJI COBA WORD EMBEDDING FASTTEXT PRE-TRAINED

Vektor Pre-Trained	*F1 for*	*F1 observing*	*F1 against*	*F1 Macro*
cc.id.300.bin				
Tanpa Fitur	0,476	0,814	0,343	0,538
Dengan Fitur	0,549	0,789	0,461	**0,600**
wiki.id.bin				
Tanpa Fitur	0,484	0,823	0,337	**0,547**
Dengan Fitur	0,536	0,628	0,406	0,500

C. *Jenis Classifier*

Dalam Tabel VI, terlihat hasil dari uji coba bila input hanya menggunakan vektor kalimat tanpa penambahan fitur. Ditemukan bahwa metode *classifier* CNN justru memberi F1 macro tertinggi dengan cc.id.bin (55,4%). Hasil yang hampir sama juga didapat dengan wiki.id.bin.

TABEL VI
HASIL UJI COBA METODE CLASSIFIER CNN, LSTM, DAN BI-LSTM, DENGAN FASTTEXT CC.ID.300.BIN DAN WIKI.ID.BIN

Classifier	*F1 for*	*F1 observing*	*F1 against*	*F1 Macro*
cc.id.300.bin				
CNN	**0.495**	**0.824**	0.345	**0.554**
LSTM	0.492	0.801	0.294	0.529
Bi-LSTM	0.441	0.817	**0.391**	0.530
wiki.id.bin				
CNN	**0.512**	0.807	0.345	**0.553**
LSTM	0.445	0.827	**0.350**	0.540
Bi-LSTM	0.496	**0.834**	0.315	0.547

D. *Fitur Kalimat dan Post*

Pada Tabel VII tampak bahwa adanya informasi keberadaan kata negatif dan *cosine similarity* mengakibatkan adanya perbaikan performa yang cukup besar. Akurasi Bi-LSTM meningkat sebesar 9,4% menjadi 62,4% dari 53% tanpa penambahan vektor fitur. Akurasi masing-masing aspek uji coba dengan CNN juga memberikan hasil yang tertinggi, kecuali pada fitur *kata* negatif saja dan fitur *cosine similarity* saja. Nilai *F1 Macro* tertinggi didapat oleh CNN dengan 62,9%. Model Bi-LSTM juga memperoleh hasil yang cukup serupa dengan 62,4%. Nilai *F1 Macro* dari pemodelan kalimat dengan rata-rata dan *classifier* CNN dan Bi-LSTM juga memberikan akurasi yang lebih tinggi daripada hasil uji coba sebesar 60,66 % pada penelitian sebelumnya [16]. Akan tetapi, penambahan fitur *likes* dan *comment* justru memperburuk akurasi dari setiap *classifier*. CNN mengalami penurunan menjadi 60% dan Bi-LSTM menjadi 61,3%.

TABEL VII
HASIL UJI COBA VEKTOR KALIMAT DAN FITUR KALIMAT DAN POST

Classifier	*F1 for*	*F1 observing*	*F1 against*	*F1 Macro*
Tanpa Vektor Fitur				
CNN	**0.495**	**0.824**	**0.345**	**0.554**
LSTM	0.492	0.801	0.294	0.529
Bi-LSTM	0.441	0.817	0.391	0.530
Dengan Vektor Fitur Kata Negatif				
CNN	**0.579**	0.778	0.443	0.600
LSTM	0.552	0.785	0.468	0.602
Bi-LSTM	0.539	**0.807**	**0.512**	**0.618**
Dengan Vektor Fitur *Cosine Similarity*				
CNN	0.440	0.745	**0.432**	0.539
LSTM	**0.513**	0.774	0.312	0.535
Bi-LSTM	0.463	**0.793**	0.381	**0.547**
Dengan Vektor Fitur Kata Negatif + *Cosine Similarity*				
CNN	**0.588**	0.791	**0.511**	**0.629**
LSTM	0.530	0.802	0.481	0.605
Bi-LSTM	0.543	**0.822**	0.510	0.624
Dengan Vektor Fitur Kata Negatif + *Cosine Similarity* + *Likes* dan *Comment*				
CNN	**0.561**	**0.802**	0.433	0.600
LSTM	0.534	0.781	0.451	0.588
Bi-LSTM	0.553	0.785	**0.500**	**0.613**

E. *Dropout dan Jenis Activation*

Dalam Tabel VIII, terbukti bahwa akurasi tertinggi sebesar 62,9% dicapai dengan nilai *dropout* 0,5 dengan *activation* ReLU. Peningkatan nilai *dropout* tampak menurunkan akurasi dari semua model, terutama pada model *classifier* CNN yang hanya menghasilkan *F1 macro* sebesar 37,6%.

TABEL VIII
HASIL UJI COBA DROPOUT DAN ACTIVATION RELU

Classifier	*F1 for*	*F1 observing*	*F1 against*	*F1 Macro*
Dropout 0.5 + ReLU				
CNN	**0.588**	0.791	**0.511**	**0.629**
LSTM	0.530	0.802	0.481	0.605
Bi-LSTM	0.543	**0.822**	0.510	0.624
Dropout 0.75 + ReLU				
CNN	**0.591**	0.780	0.405	0.592
LSTM	0.538	0.798	0.452	0.597
Bi-LSTM	0.506	**0.808**	**0.511**	**0.609**
Dropout 0.99 + ReLU				
CNN	0.108	0.484	0.172	0.253
LSTM	0.539	**0.792**	**0.384**	0.572
Bi-LSTM	**0.572**	**0.791**	**0.380**	**0.582**

Terbukti pula bahwa *activation function* Leaky ReLU hanya mampu meningkatkan efektifitas sebesar 0,4% pada model LSTM dibandingkan dengan model yang menggunakan ReLU. *F1 Macro* model justru menurun untuk model lain. *F1 Macro* CNN turun 1,4% menjadi 61,5% dan Bi-LSTM turun 2,8% menjadi 59,6%. Leaky ReLU sendiri paling akurat apabila digunakan dengan nilai *dropout* 0,5. Kombinasi ini menghasilkan *F1 macro* tertinggi pada model *classifier* CNN sebesar 61,5% seperti yang tampak pada Tabel IX.

TABEL IX
HASIL UJI COBA DROPOUT DAN ACTIVATION LEAKY RELU

Classifier	*F1 for*	*F1 observing*	*F1 against*	*F1 Macro*
Dropout 0.5 + Leaky ReLU				
CNN	**0.568**	0.780	**0.504**	**0.615**
LSTM	0.533	**0.809**	0.482	0.609
Bi-LSTM	0.519	0.778	0.495	0.596
Dropout 0.75 + Leaky ReLU				
CNN	**0.587**	0.783	0.406	0.593
LSTM	0.534	**0.805**	**0.485**	**0.609**
Bi-LSTM	0.534	0.772	0.480	0.595
Dropout 0.99 + Leaky ReLU				
CNN	0.138	0.565	0.270	0.324
LSTM	**0.555**	**0.795**	0.362	**0.571**
Bi-LSTM	0.543	0.753	**0.395**	0.563

F. *Hidden Units*

Dalam Tabel X, akurasi dapat ditingkatkan dengan penambahan *hidden units*. Pada model *classifier* LSTM, akurasi bertambah 3% dari 60,2% menjadi 63%. Model *classifier* Bi-LSTM pun mengalami peningkatan sebesar 0,6% menjadi 64%.

Dibandingkan dengan penelitian terakhir mengenai *stance classification* dengan *deep learning* [16], penelitian ini memperoleh peningkatan *F1-score* sebesar 3,3% menjadi 64% dari 60,66%. Perbandingan ini hanya berdasarkan hasil uji coba dengan metode pemodelan kalimat rata-rata dan *classifier* LSTM pada penelitian sebelumnya. Klaim dan judul terkait dalam *dataset* penelitian sebelumnya berjumlah 3.954 judul. Sementara itu, penelitian ini memiliki total 3.500 judul dengan jumlah klaim yang sama dengan penelitian sebelumnya yakni 500. Kedua *dataset* sama-sama memiliki tiga kelas *output* dan berdistribusi sekitar 30% per kelas.

TABEL X
HASIL UJI COBA HIDDEN UNITS

Hidden Units	*F1 for*	*F1 observing*	*F1 against*	*F1 Macro*
LSTM				
200	**0.589**	**0.803**	**0.503**	**0.630**
100	0.570	0.794	0.440	0.602
50	0.566	0.792	0.496	0.619
Bi-LSTM				
200	**0.588**	**0.811**	**0.524**	**0.640**
100	0.562	0.811	0.523	0.634
50	0.578	0.809	0.522	0.637

Akan tetapi, *F1-score* model sebesar 64% ini masih dibawah *F1-score* penelitian Kochkina [22] sebesar 78,4%. Penelitian tersebut menggunakan *dataset* 5568 *tweet*, dengan empat kelas *output*. Distribusi *tweet* per kelas S:D:Q:C adalah 1004:415:464:3685.

V. KESIMPULAN

Stance classification berbahasa Indonesia lebih akurat dilakukan dengan fastText dibandingkan dengan Word2Vec. Word *embedding* fastText sebaiknya digunakan dengan vektor *pre-trained* wiki.id.bin untuk *stance classification* tanpa fitur. Akan tetapi, gunakan cc.id.300.bin bila terdapat informasi fitur kalimat dan *post*. Dalam memilih *classifier*, CNN dan Bi-LSTM adalah dua pilihan terbaik. Terutama bila dilakukan pengaturan *hidden units* pada Bi-LSTM. Penambahan fitur sendiri tidak cukup dengan mencatutkan informasi tanda tanya saja. Perlu ditambahkan informasi ada atau tidaknya kata negatif dalam setiap kalimat dan nilai *cosine similarity* kalimat klaim dan reaksi. Akan tetapi, tidak perlu menambahkan informasi jumlah *like* dan *comment* pada *post* yang justru memperburuk akurasi model. Nilai *dropout* yang paling efektif adalah 0,5 dengan *activation function* ReLU. Diperlukan pula pengaturan *hidden units* agar tidak terlalu sedikit sehingga model dapat menjadi lebih akurat. Penelitian ini dapat dikembangkan lagi pada eksplorasi efek fitur kalimat dan *metadata* media sosial lain terhadap performa. Pemahaman makna kalimat juga dapat dikembangkan dengan pendekatan atensi dan pengaruhnya pada akurasi klasifikasi *stance*.

DAFTAR PUSTAKA

[1] D. Goldie, M. Linick, H. Jabbar, and C. Lubienski, "Using Bibliometric and Social Media Analyses to Explore the 'Echo Chamber' Hypothesis," *Educ. Policy*, vol. 28, no. 2, pp. 281–305, 2014, doi: 10.1177/0895904813515330.

[2] S. Jacobson, E. Myung, and S. L. Johnson, "Open media or echo chamber: the use of links in audience discussions on the Facebook Pages of partisan news organizations," *Inf. Commun. Soc.*, vol. 19, no. 7, pp. 875–891, 2016, doi: 10.1080/1369118X.2015.1064461.

[3] J. J. Van Bavel and A. Pereira, "The Partisan Brain: An Identity-Based Model of Political Belief," *Trends Cogn. Sci.*, vol. 22, no. 3,

pp. 213–224, 2018, doi: 10.1016/j.tics.2018.01.004.
[4] F. Zollo and W. Quattrociocchi, “Misinformation Spreading on Facebook,” pp. 177–196, 2018, doi: 10.1007/978-3-319-77332-2_10.
[5] C. Shao, G. L. Ciampaglia, O. Varol, K. C. Yang, A. Flammini, and F. Menczer, “The spread of low-credibility content by social bots,” *Nat. Commun.*, vol. 9, no. 1, 2018, doi: 10.1038/s41467-018-06930-7.
[6] H. Allcott and M. Gentzkow, “Social Media and Fake News in the 2016 Election,” *J. Econ. Perspect.*, vol. 31, no. 2, pp. 211–236, 2017, doi: 10.1257/jep.31.2.211.
[7] C. Reuter, S. Stieglitz, and M. Imran, “Social media in conflicts and crises,” *Behav. Inf. Technol.*, vol. 39, no. 3, pp. 241–251, 2020, doi: 10.1080/0144929X.2019.1629025.
[8] M. Roy, N. Moreau, C. Rousseau, A. Mercier, A. Wilson, and L. Atlani-Duault, “Ebola and Localized Blame on Social Media: Analysis of Twitter and Facebook Conversations During the 2014–2015 Ebola Epidemic,” *Cult. Med. Psychiatry*, vol. 44, no. 1, pp. 56–79, 2020, doi: 10.1007/s11013-019-09635-8.
[9] S. Sommariva, C. Vamos, A. Mantzarlis, L. U. L. Đào, and D. Martinez Tyson, “Spreading the (Fake) News: Exploring Health Messages on Social Media and the Implications for Health Professionals Using a Case Study,” *Am. J. Heal. Educ.*, vol. 49, no. 4, pp. 246–255, 2018, doi: 10.1080/19325037.2018.1473178.
[10] S. M. Mohammad, P. Sobhani, and S. Kiritchenko, “Stance and Sentiment in Tweets,” vol. 0, no. 0, 2016, [Online]. Available: http://arxiv.org/abs/1605.01655.
[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *ICLR Work.*, pp. 1–12, 2013, [Online]. Available: http://arxiv.org/abs/1301.3781.
[12] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural Language Processing (Almost) from Scratch,” *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011, doi: 10.1109/CIC.2017.00050.
[13] Y. Kim, “Convolutional neural networks for sentence classification,” *EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, pp. 1746–1751, 2014.
[14] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A large annotated corpus for learning natural language inference,” *Conf. Proc. - EMNLP 2015 Conf. Empir. Methods Nat. Lang. Process.*, pp. 632–642, 2015, doi: 10.18653/v1/d15-1075.
[15] R. Jannati, R. Mahendra, C. W. Wardhana, and M. Adriani, “Stance Classification Towards Political Figures on Blog Writing,” *Proc. 2018 Int. Conf. Asian Lang. Process. IALP 2018*, pp. 96–101, 2019, doi: 10.1109/IALP.2018.8629144.
[16] E. I. Setiawan, A. Ferdianto, J. Santoso, Y. Kristian, S. Sumpeno, and M. H. Purnomo, “Analisis Pendapat Masyarakat terhadap Berita Kesehatan Indonesia menggunakan Pemodelan Kalimat berbasis LSTM (Indonesian Stance Analysis of Healthcare News using Sentence Embedding,” *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 9, no. 1, pp. 8–17, 2020.
[17] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information,” *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.
[18] A. Graves and J. Schmidhuber, “Framewise Phoneme Classification with Bidirectional LSTM Networks,” *Proceedings. 2005 IEEE Int. Jt. Conf. Neural Networks, 2005.*, vol. 4, pp. 2047–2052, 2005.
[19] M. García Lozano, H. Lilja, E. Tjörnhammar, and M. Karasalo, “Mama Edha at SemEval-2017 Task 8: Stance Classification with CNN and Rules,” *Proc. 11th Int. Work. Semant. Eval.*, pp. 481–485, 2018, doi: 10.18653/v1/s17-2084.
[20] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, pp. 3104–3112, 2014.
[21] W.-F. Chen and L.-W. Ku, “UTCNN: a Deep Learning Model of Stance Classificationon on Social Media Text,” *Proc. COLING 2016, 26th Int. Conf. Comput. Linguist. Tech. Pap.*, pp. 1635–1645, 2016, [Online]. Available: http://arxiv.org/abs/1611.03599.
[22] E. Kochkina, M. Liakata, and I. Augenstein, “Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM,” no. 2016, pp. 475–480, 2018, doi: 10.18653/v1/s17-2083.
[23] K. Shalini, M. Anand Kumar, and K. Soman, *Deep-Learning-Based Stance Detection for Indian Social Media Text*, vol. 545. Springer Singapore, 2019.
[24] M. Thomas, B. Pang, and L. Lee, “Get out the vote: Determining support or opposition from Congressional oor-debate transcripts,” *In Proc. of EMNLP*, no. July, pp. 327–335, 2006.
[25] K. Hasan and V. Ng, “Stance Classification of Ideological Debates : Data , Models , Features , and Constraints,” *Proc. SIGDIAL 2013 Conf.*, no. October, pp. 1348–1356, 2013.
[26] B. Riedel, I. Augenstein, G. P. Spithourakis, and S. Riedel, “A simple but tough-to-beat baseline for the Fake News Challenge stance detection task,” pp. 1–6, 2017, [Online]. Available: http://arxiv.org/abs/1707.03264.
[27] A. Hanselowski *et al.*, “A Retrospective Analysis of the Fake News Challenge Stance Detection Task,” 2018, [Online]. Available: http://arxiv.org/abs/1806.05180.
[28] I. Augenstein, T. Rocktäschel, A. Vlachos, and K. Bontcheva, “Stance Detection with Bidirectional Conditional Encoding,” pp. 876–885, 2016, doi: 10.18653/v1/d16-1084.
[29] I. Habernalt and I. Gurevych, “Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM,” *54th Annu. Meet. Assoc. Comput. Linguist. ACL 2016 - Long Pap.*, vol. 3, pp. 1589–1599, 2016, doi: 10.18653/v1/p16-1150.
[30] D. Mrowca and E. Wang, “Stance detection for fake news identification,” pp. 1–12, 2017.
[31] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A Neural Probabilistic Language Model,” *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003, doi: 10.1080/1536383X.2018.1448388.
[32] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” *15th Conf. Eur. Chapter Assoc. Comput. Linguist. EACL 2017 - Proc. Conf.*, vol. 2, pp. 427–431, 2017, doi: 10.18653/v1/e17-2068.
[33] I. Santos, N. Nedjah, and L. de Macedo Mourelle, “Sentiment analysis using convolutional neural network with fastText embeddings,” pp. 1–5, 2018, doi: 10.1109/la-cci.2017.8285683.
[34] R. Kiros *et al.*, “Skip-thought vectors,” *Adv. Neural Inf. Process. Syst.*, vol. 2015-Janua, no. 786, pp. 3294–3302, 2015.
[35] L. Logeswaran and H. Lee, “An efficient framework for learning sentence representations,” *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, pp. 1–16, 2018.
[36] Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg, “Fine-grained analysis of sentence embeddings using auxiliary prediction tasks,” *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, pp. 1–13, 2019.
[37] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, “Towards universal paraphrastic sentence embeddings,” *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*, pp. 1–19, 2016.
[38] F. Z. Tala, “A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia,” *M.Sc. Thesis, Append. D*, vol. pp, pp. 39–46, 2003.
[39] Y. Zhang and B. Wallace, “A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification,” 2015, [Online]. Available: http://arxiv.org/abs/1510.03820.

Ernest Lim adalah lulusan S1 jurusan Aristektur di Universitas Katolik Parahyangan pada tahun 2016. Ernest kemudian melanjutkan studi S2 Teknologi Informasi di STTS sejak tahun 2018.

Esther Irawati Setiawan menyelesaikan studi S1 di program Teknik Informatika STTS pada tahun 2006. Menyelesaikan studi master Teknologi Informasi STTS pada tahun 2010. Minat penelitian pada bidang *Social Network Analysis* dan *Web Mining*.

Joan Santoso menyelesaikan studi S1 Teknik Informatika di STTS pada tahun 2011. Menyelesaikan studi S2 Teknologi Informasi di STTS pada tahun 2013. Minat penelitian pada bidang *computational linguistics*, *information extraction*, *machine learning*, dan *big data processing*.