Identifikasi Tweet yang Mengandung Sarkasme Dalam Studi Kasus Pemilihan Presiden 2019 Menggunakan Metode Long Short-term Memory

Nadine Azhalia Purbani[1], Anisa Herdiani[2], Ade Romadhony[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]nadineazhalia@students.telkomuniversity.ac.id, [2]anisaherdiani@telkomuniversity.ac.id,
[3]aderomadhony@telkomuniversity.ac.id

Abstrak

Twitter saat ini menjadi salah satu sarana media sosial yang digunakan untuk melakukan kampanye publik, termasuk kampanye calon presiden pada pemilu tahun ini. *Tweet* kampanye sering kali mengandung sarkasme, yaitu penggunaan kata-kata pedas yang bertujuan untuk menyakiti hati orang lain. Adanya sarkasme yang terkandung di dalam suatu *tweet* membuat kebanyakan orang gagal memahami makna yang disampaikan. Sarkasme juga sering digunakan oleh seseorang untuk menyampaikan cemoohan atau ejekan kasar, sehingga dapat memicu pertikaian. Oleh karena itu, deteksi sarkasme perlu dilakukan agar dapat mengurangi potensi pertikaian. Deteksi sarkasme merupakan sebuah tugas yang menantang, karena sarkasme sendiri cukup sulit untuk ditentukan. Pada Tugas Akhir ini dilakukan deteksi sarkasme pada *tweet* dengan topik pemilihan presiden Indonesia 2019. Metode yang digunakan untuk klasifikasi adalah Long Short-Term Memory (LSTM), dan fitur kata direpresentasikan dalam bentuk vektor *word embedding*. Pada Tugas Akhir ini dilakukan pembangunan dataset dengan mengambil *tweet* pada rentang waktu 10 hari yang terkait dengan topik pilpres 2019. Berdasarkan hasil pengujian, nilai terbaik yang didapat dari *accuracy*, *precision*, dan *recall* secara berurut adalah sebesar 93%, 92%, dan 95%.

Kata kunci : identifikasi, klasifikasi teks, twitter, sarkasme, LSTM.

Abstract

Twitter is currently one of the means of social media used for public campaigns, including the campaign for presidential candidates in this year's election. Campaign tweets often contain sarcasm, which is the use of scathing words to hurt others. The existence of sarcasm contained in a tweet makes most people fail to understand the meaning conveyed. Sarcasm is also often used by someone to convey harsh ridicule or ridicule, so that it can trigger conflict. Therefore, detection of sarcasm needs to be done in order to reduce the potential for conflict. Detection of sarcasm is a challenging task, because sarcasm itself is quite difficult to determine. In this Final Project, sarcasm detection was carried out on tweets with the topic of the 2019 Indonesian presidential election. The method used for classifying text was Long Short-Term Memory (LSTM), and the word feature was represented in the form of word embedding vectors. In this Final Project, a dataset was developed by taking tweets in the span of 10 days related to the topic of the 2019 presidential election. Based on the test results, the best values obtained from accuracy, precision, and recall are sequentially is 93 %, 92 %, and 95 %.

Keywords: identification, text classification, twitter, sarcasm, LSTM.

1. Pendahuluan

Saat ini twitter menjadi salah satu media sosial yang paling dipilih untuk melakukan interaksi antar individu secara *online*. Twitter merupakan situs *microblogging* yang memungkinkan penggunanya untuk membaca atau menulis opini yang biasa disebut sebagai *tweet*, dengan panjang maksimal sebanyak 280 karakter. Hingga saat ini terdapat sebanyak 285 juta pengguna aktif twitter di seluruh dunia yang telah menulis hingga jutaan *tweet* perharinya [5].

Pada masa pemilihan presiden 2019, twitter dipenuhi dengan *tweet* bertema politik, termasuk juga yang mengandung sarkasme. Adanya *tweet* yang mengandung sarkasme membuat iklim pilpres menjadi tidak sehat, karena banyak terlontar kalimat-kalimat yang memiliki makna yang tidak baik, sehingga dapat memicu pertikaian. Keberadaan pendeteksi sarkasme dapat membantu mencegah potensi pertikaian.

Menurut Kamus Besar Bahasa Indonesia (KBBI) pengertian sarkasme adalah penggunaan kata-kata pedas untuk menyakiti hati orang lain seperti cemoohan atau ejekan kasar [1]. Sarkasme merupakan suatu majas yang erat kaitannya dengan ironi [14], terdapat banyak definisi antara ironi dan sarkasme. Secara tradisional, perbedaan antara keduanya terletak pada penyampaiannya, ironi bersifat tidak langsung sedangkan sarkasme bersifat langsung [15]. Penyampaian sarkasme tanpa potensi pertikaian dapat terjadi dengan syarat adanya pengetahuan bersama antara pembicara dan pembaca[2], di mana hal ini sulit tercapai pada media sosial yang dapat diakses publik dengan pengetahuan yang bervariasi. Kondisi ideal adanya pengetahuan bersama antara pembicara dan pembaca membuat maksud dari sindiran yang diberikan dapat dipahami dengan baik oleh kedua belah pihak.

Deteksi sarkasme merupakan salah satu *task* dalam bidang pemrosesan bahasa alami[19]. Kalimat yang mengandung sarkasme cukup sulit dideteksi jika dilakukan secara manual dan akan menyesatkan pembaca jika tidak disadari[4].

Fokus penelitian ini adalah deteksi sarkasme yang terkandung dalam sebuah *tweet* bertema politik. Klasifikasi dilakukan dengan menggunakan metode Long Short-Term Memory (LSTM), karena secara umum pendekatan *deep learning* memiliki akurasi yang lebih baik dibandingkan dengan pendekatan pembelajaran mesin konvensional seperti *naive bayes*, *support vector machine*, dll[21]. Pengklasifikasi memproses *tweet* yang direpresentasikan dengan vektor oleh *word embedding* (*word2vec*), karena keterkaitan semantik antar kata dapat ditangkap dengan lebih baik dibandingkan dengan metode berbasis *vector space model*[21]. LSTM merupakan salah satu bagian dari *Recurrent Neural Network* (RNN), yang diciptakan untuk mengatasi masalah *vanishing gradient* yang terjadi pada RNN[9]. Kelebihan lain dari LSTM adalah memiliki performansi yang baik jika digunakan untuk data yang bersifat *sequence* dan dapat mengingat informasi yang didapat pada teks sebelumnya[9].

2. Studi Terkait

2.1 Klasifikasi Teks

Telah banyak penelitian yang membahas tentang klasifikasi teks termasuk pada teks bahasa Indonesia, khususnya untuk kasus identifikasi. Pada penelitian sebelumnya telah dilakukan klasifikasi teks berbahasa Indonesia untuk menentukan analisis sentimen pada *tweet* berbahasa Indonesia menggunakan pendekatan *deep learning* yang menghasilkan nilai akurasi yang lebih baik dibandingkan menggunakan metode *machine learning* konvensional yaitu *naive bayes*[1].

2.2 Deteksi Sarkasme

Penelitian serupa tentang deteksi sarkasme pada *tweet* dilakukan oleh Tomas Ptacek, Ivan Habernal, dan Jun Hong[20]. Bahasa yang digunakan pada penelitian tersebut adalah Bahasa Ceko dan Bahasa Inggris. Penelitian tersebut menggunakan 140,000 *tweet* berbahasa Ceko yang dilakukan untuk identifikasi sarkasme, dengan 7.000 data yang diberi label secara manual. Model klasifikasi dibangun dengan metode *Maximum Entropy* dan *Support Vector Machine*. Penelitian tersebut menghasilkan nilai *F-Measure* sebesar 58% untuk bahasa ceko dan 92% untuk bahasa inggris dengan data yang tidak seimbang[20].

Sementara penelitian serupa pada teks bahasa Indonesia dilakukan oleh Edwin Lunando, dkk[13]. Deteksi sarkasme pada penelitian tersebut dilakukan untuk melakukan analisis sentimen terhadap media sosial dengan topik politik, makanan, film, dan publik figur. Sedangkan pada penelitian ini topik dari dataset hanyak mencakup topik politik. Metode klasifikasi yang digunakan pada [13] adalah *Machine Learning* yaitu *naive bayes*, *maximum entropy*, dan *support vector machine* dengan nilai akurasi yang dihasilkan secara berurut adalah 45,7%, 47,1%, dan 48,5%.

2.3 Long Short-Term Memory (LSTM)

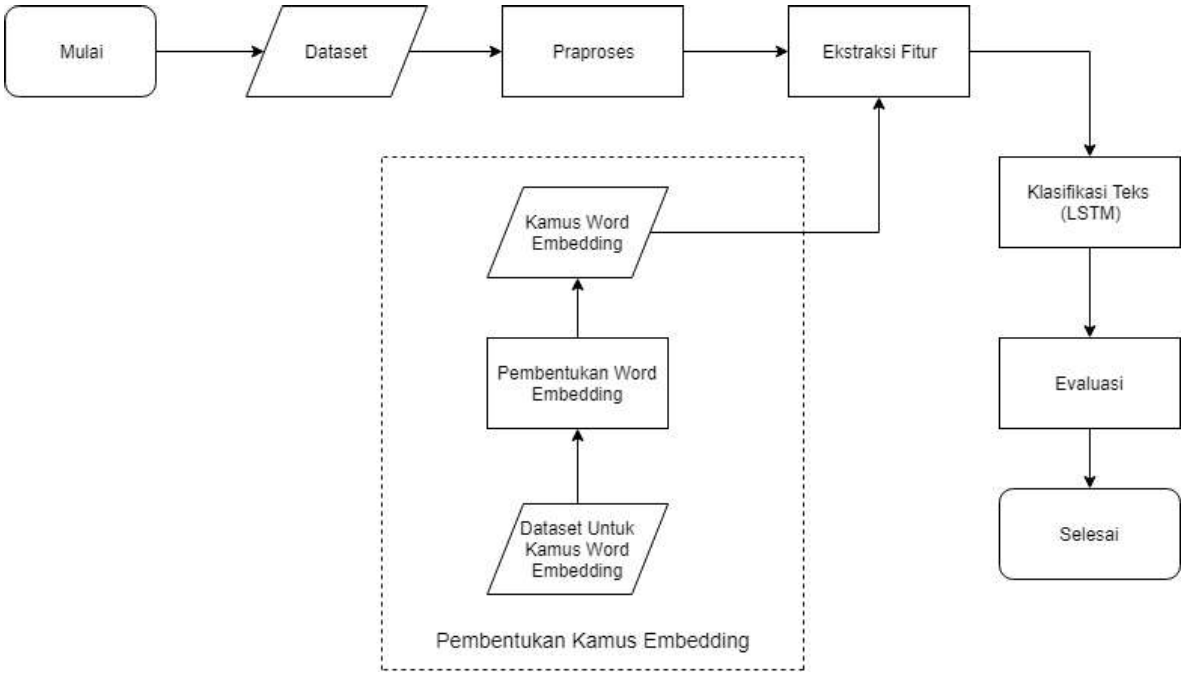
LSTM telah banyak dilakukan pada beberapa penelitian dan menghasilkan model dengan performansi yang baik. Metode LSTM bisa digunakan untuk melakukan klasifikasi teks[21], pengenalan suara[7], mesin penerjemah[8], hingga pemberian keterangan pada gambar atau *image captioning* [17]. Metode LSTM memiliki performansi yang baik jika data yang digunakan bersifat *sequence* seperti teks.

Pada penelitian yang dilakukan oleh Peng Zhou, dkk[23], melakukan klasifikasi teks dengan 6 data berbeda menggunakan metode *bidirectional* LSTM. Membangun kamus embedding menggunakan *Gloves* dengan data yang diambil dari wikipedia 2014 dan gigaword 5. Menghasilkan hasil yang baik pada 4 dari 6 data yang ada.

[1] https://kbbi.web.id/sarkasme

3. Sistem yang Dibangun

Terdapat tiga tahap utama untuk membangun pendeteksi sarkasme pada penelitian ini, yaitu tahap persiapan data, tahap pembangunan model, dan tahap pengujian model. Gambaran umum sistem deteksi sarkasme menggunakan LSTM dapat dilihat pada gambar 1. Pada tahap persiapan data dilakukan praproses agar data memiliki bentuk yang sama dan dilakukan pemisahan data menjadi data latih dan data uji. Setelah melalui tahap persiapan data, selanjutnya adalah tahap pembangunan model yaitu klasifikasi teks dengan LSTM. Terdapat juga tahap pembentukan vektor kata yang digunakan sebagai representasi teks masukan dengan menggunakan *word embedding*. Tahap terakhir adalah tahap pengujian model yang dilakukan dengan menghitung nilai *accuracy*, *precision*, *recall*, dan *loss*.



Gambar 1. **Gambaran Umum Sistem**

3.1 Dataset

Data yang digunakan diambil dari twitter menggunakan *API Streaming Twitter*. *Tweet* yang diambil berkaitan dengan acara politik, yaitu pemilihan presiden Indonesia 2019. Pada masa pilpres merupakan waktu yang cocok untuk mengambil *tweet* yang mengandung sarkasme, karena terdapat banyak tweet yang bersifat mencela.

Data diambil selama masa kampanye presiden mulai tanggal 22 Februari 2019 sampai 5 Maret 2019, terkumpul sebanyak 9.374 data yang diambil menggunakan keyword "#pilpres2019", "#debatcapres2019", "jokowi", "prabowo", dll. Setelah menghilangkan data yang duplikat, data yang terkumpul sebanyak 6.633 yang belum memiliki label. Pelabelan data dilakukan dengan cara meminta pengguna internet untuk memberi label pada *tweet* yang diberikan melalui *website crowdsourcing* yang dapat diakses pada URL: https://sarcasm-dataset.herokuapp.com/. Anotator diberi panduan untuk melabeli data berdasarkan pengertian sarkasme yang diacu dari kamus besar bahasa Indonesia. Tidak diberikan daftar kata yang mempengaruhi kalimat yang memiliki makna sarkasme. Anotator mendefinisikan sendiri *tweet* tersebut mengandung sarkasme atau tidak berdasarkan acuan dari KBBI.

Setelah melakukan pelabelan data secara *crowdsourcing*, didapat sebanyak 2.183 *tweet* yang telah diberi label. Terdiri dari 890 *tweet* berlabel sarkasme dan 1.292 tweet berlabel tidak sarkasme. Contoh dataset dapat dilihat pada tabel 1. Data dikatakan mengandung sarkasme apabila terdapat kata kasar, bermakna kasar, atau mengejek. Sebaliknya, data dikatakan tidak mengandung sarkasme apabila tidak terdapat kata tersebut.

Dataset dibagi menjadi tiga bagian yaitu data latih, data validasi, dan data uji. Pembagian data dilakukan dengan cara *train-validation-test split* yang terdiri dari 1.847 data latih, 98 data validasi, dan 103 data uji.

Tabel 1. **Dataset**

Tweet	Label
Partainya bangsat Capresnya bangsat Pengurusnya bangsat Otaknya bangsat Mulutnya bangsat	Yes
Jokowi Suguhkan Menu Khas Aceh saat Terima Para Ulama Serambi Mekkah Surya Paloh	No

3.2 Praproses

Praproses dilakukan untuk mempersiapkan data agar memiliki bentuk yang standar, untuk kasus ini bentuk yang standar berupa perubahan kata menjadi bentuk numerik yang direpresentasikan dalam vektor kata. Pada umumnya, praproses dilakukan dengan cara mengeliminasi data yang tidak sesuai atau mengubah data menjadi bentuk yang lebih mudah diproses oleh proses lain. Praproses sangat penting dalam melakukan deteksi teks, terutama untuk media sosial yang sebagian besar berisi kata-kata atau kalimat yang tidak formal dan tidak terstruktur serta memiliki *noise* yang besar[16]. Pada sistem ini praproses yang digunakan mencakup:

1. *Data Cleaning* merupakan proses pembersihan data yang mencakup: penghapusan tautan, username, RT, dan tagar yang terdapat pada dataset. Contoh dapat dilihat pada tabel 2.

Tabel 2. **Proses Data Cleaning**

proses	input	output
penghapusan tautan	Dalam catatannya, upaya pemerintah untuk menggunakan instrumen hukum dengan cara ini telah menjadi jauh lebih terbuka dan sistematis di bawah Jokowi https://t.co/wFcnsZrmWC	Dalam catatannya, upaya pemerintah untuk menggunakan instrumen hukum dengan cara ini telah menjadi jauh lebih terbuka dan sistematis di bawah Jokowi
penghapusan username	RT @ryzenrx: #01BandarSabu menolak lupa	RT #01BandarSabu menolak lupa
penghapusan RT	RT #01BandarSabu menolak lupa	#01BandarSabu menolak lupa
penghapusan tagar	#01BandarSabu menolak lupa	01 Bandar Sabu menolak lupa

2. *Case Folding* merupakan proses pengkonversian teks dari huruf kapital menjadi huruf kecil. Contoh dapat dilihat pada tabel 3.

Tabel 3. **Proses Case Folding**

input	Presiden Jokowi Perintahkan Perbaiki Jalan Rusak di Sumsel Usai Viral Jadi Lokasi Foto Anak Muda .
output	presiden jokowi perintahkan perbaiki jalan rusak di sumsel usai viral jadi lokasi foto anak muda .

3. *Punctuation Removal* merupakan proses penghilangan tanda baca yang terdapat dalam teks. Contoh dapat dilihat pada tabel 4.

Tabel 4. **Proses Punctuation Removal**

input	presiden jokowi perintahkan perbaiki jalan rusak di sumsel usai viral jadi lokasi foto anak muda .
output	presiden jokowi perintahkan perbaiki jalan rusak di sumsel usai viral jadi lokasi foto anak muda

4. *Tokenizing* merupakan proses memecah suatu kalimat menjadi potongan kata. Panjang token kata yang digunakan dalam kasus ini adalah 46 kata, nilai ini didapat dari rata-rata banyaknya kata yang ada pada dataset. Ilustrasi dari proses ini dapat dilihat pada tabel 5.

Tabel 5. **Ilustrasi Proses Tokenizing**

input	presiden jokowi perintahkan perbaiki jalan rusak di sumsel usai viral jadi lokasi foto anak muda
output	[presiden, jokowi, perintahkan, perbaiki, jalan, rusak, di, sumsel, usai, viral, jadi, lokasi, foto, anak, muda]

5. *pad sequence* merupakan proses penambahan angka nol di awal vektor kata yang jumlah katanya kurang dari 46 kata dan penghilangan kata yang terletak di akhir kalimat jika vektor kata melebihi 46 kata. Hal ini dilakukan agar *classifier* dapat menentukan jumlah modul sesuai dengan panjang kata yang ditentukan. Ilustrasi dari proses ini dapat dilihat pada tabel 6.

Tabel 6. **Ilustrasi Proses Pad Sequence**

input	[89, 5, 76, 32, 7, 15, 8, 31, 88, 30, 43, 9, 52, 11, 83]
output	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 89, 5, 76, 32, 7, 15, 8, 31, 88, 30, 43, 9, 52, 11, 83]

Hasil yang dikeluarkan pada tahap ini adalah representasi kata yang diubah dalam bentuk vektor.

3.3 Model

Tahap pembangunan model merupakan salah satu tahap yang dijalankan untuk membangun sistem pendeteksi sarkasme. Pada tahap ini dilakukan klasifikasi dengan LSTM menggunakan fitur yang sudah diekstraksi menggunakan *word embedding*.

3.3.1 *Word Embedding*

Ekstraksi fitur dilakukan dengan cara embedding. Pada dasarnya proses embedding merupakan proses inisialisasi vektor kata, tujuannya adalah untuk menentukan kemiripan antar kata yang direpresentasikan dalam bentuk vektor. *Word embedding* memiliki komputasi yang efisien karena memiliki jumlah dimensi yang tetap[21]. Pada penelitian ini dilakukan dua skenario pembangunan kamus embedding, yaitu dengan cara *pre-trained* dan tanpa melakukan *pre-trained*. Untuk membangun kamus embedding dengan cara *pre-trained*, kamus embedding dilatih menggunakan data yang diambil dari twitter dengan topik pilpres agar memiliki struktur yang serupa dengan dataset. Untuk membangun kamus embedding digunakan *library* gensim [2]. Sedangkan dengan cara tanpa melakukan *pre-trained*, kamus embedding dibangun menggunakan data yang digunakan pada proses klasifikasi. Pada penelitian ini parameter yang digunakan untuk membangun kamus embedding adalah:

- *size*: *size* yang digunakan sebesar 300 untuk menentukan dimensi dari *word embedding*. Nilai untuk *size* pada *word embedding* selalu berkisar antara 100 hingga 300, dan menurut penelitian [6] nilai *size* yang optimal untuk digunakan adalah sebesar 300.
- *window*: window yang digunakan sebesar 3 untuk menentukan tetangga antar kata. Karena satu masukan kata berjumlah sedikit, maka kata yang memiliki makna terkait terletak 3 kata dari kata yang sedang diproses.
- *min count*: Kata yang muncul sebanyak kurang dari 2 kali tidak akan dimasukkan ke dalam kamus embedding, karena kata yang muncul kurang dari 2 kali tidak mempengaruhi pembangunan kamus *word embedding*.

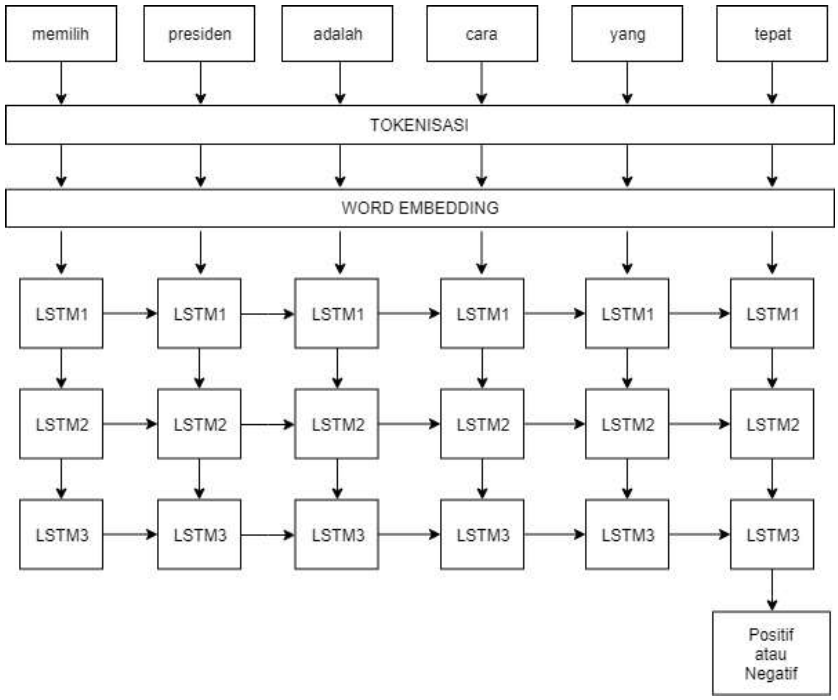
Hasil dari proses tersebut menghasilkan kamus embedding berbentuk kumpulan angka yang direpresentasikan dalam bentuk vektor. Nilai dari vektor tersebut didapatkan dari proses latih saat pembangunan kamus embedding.

[2]https://www.pydoc.io/pypi/gensim-3.2.0/autoapi/models/word2vec/index.html

3.3.2 Long Short Term-Memory (LSTM)

Pada tahap klasifikasi digunakan metode LSTM untuk membangun model, terdapat masukan berupa vektor kata yang sebelumnya sudah melalui praproses dan embedding. Mode LSTM yang digunakan adalah *many-to-one*, mode ini digunakan karena masukan yang diterima berupa sekuens dan keluaran yang dihasilkan berupa kategori dari data masukan[10]. Digunakan tiga *layer* LSTM untuk membangun model, model LSTM diilustrasikan dalam gambar 2. Untuk mengimplementasikan metode LSTM digunakan *library* keras [3] pada python. Pada penelitian ini parameter yang digunakan untuk LSTM adalah *units*, *dropout*, dan *recurrent dropout* nilai yang digunakan pada parameter tersebut dipilih dengan percobaan empiris yang telah dilakukan. Sedangkan parameter yang diatur saat sebelum melakukan training mencakup:

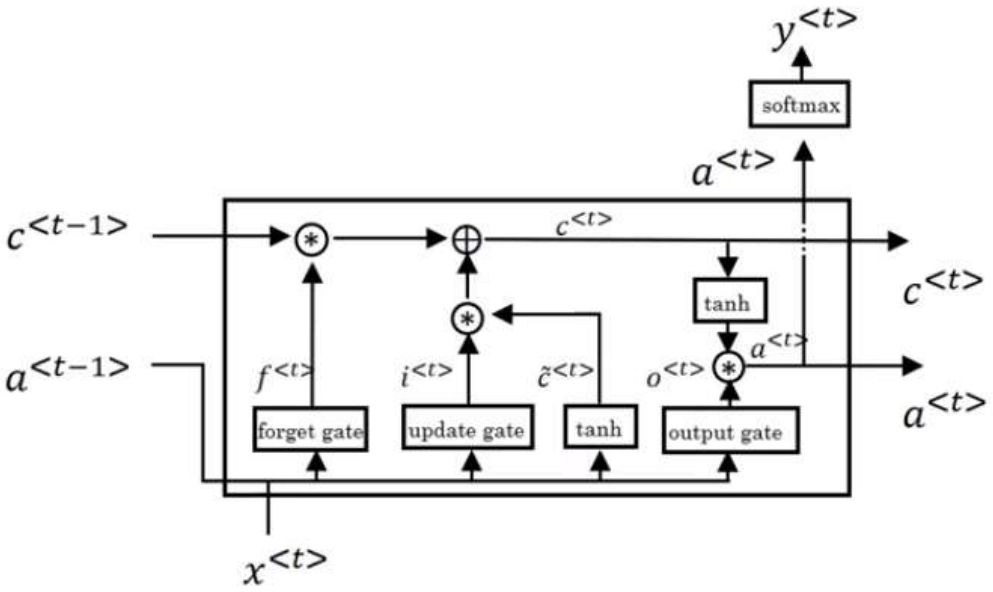
- **loss**: fungsi *loss* yang digunakan adalah *binary crossentropy*, fungsi ini dipilih karena kasus pada penelitian ini adalah *binary classification* dan fungsi aktivasi yang digunakan adalah *sigmoid*[21].
- **optimizer**: *optimizer* yang digunakan adalah rmsprop, digunakan untuk pengbaharuan *weight*. Rmsprop baik digunakan untuk metode LSTM.
- **validation split**: data validasi diambil sebanyak 5% dari data latih. Pada metode *deep learning*, data validasi dan uji disarankan tidak terlalu banyak.
- **epoch**: *epoch* yang dilakukan adalah sebanyak 10 kali, karena menghasilkan nilai *loss* yang optimal dan perubahan *loss* yang terjadi tidak memiliki selisih yang besar.
- **batch size**: untuk menjalankan satu *epoch* ukuran data dibagi menjadi 32[3], hal ini dilakukan untuk menghindari *local minimum* dan proses komputasi lebih efisien.



Gambar 2. **Layer LSTM**

LSTM memiliki memiliki tiga *gates* yang dapat mengontrol kebutuhan LSTM dan memperbarui nilai dari informasi teks yang ada sebelumnya. *Gates* yang dimiliki meliputi *update gate*, *forget gate*, dan *output gate*. Terdapat fungsi *sigmoid* pada setiap *gate*. Ketiga *gates* tersebut terletak dalam satu modul LSTM. Selain *gates* terdapat juga *memory cell*, tugas dari *memory cell* adalah untuk membaca, menyimpan, dan memperbarui informasi jarak jauh yang telah didapat sebelumnya[12]. *Memory cell* diciptakan untuk menangani masalah *vanishing gradient* yang terjadi pada RNN[9]. Alur informasi yang dilakukan dalam satu unit LSTM dapat dilihat pada gambar 3.

[3]https://keras.io/layers/recurrent/lstm



Gambar 3. **Modul LSTM**

Forget gate berfungsi untuk menentukan informasi mana yang harus disimpan dan yang harus dilupakan dari modul sebelumnya. *Forget gate* diambil dari [11] dihitung menggunakan persamaan 1.

$$f^{<t>} = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \tag{1}$$

$f^{<t>}$ merupakan representasi dari hasil *forget gate*. *Gate* ini akan membaca nilai $a^{<t-1>}$ merujuk pada *output* yang didapat dari modul sebelumnya dan $x^{<t>}$ yang merujuk pada *input* kata berbentuk vektor[18]. W_f merupakan bobot pada modul saat ini. Fungsi *sigmoid* yang digunakan pada persamaan 2 digunakan untuk mendapat nilai *forget gate*.

Update gate berfungsi untuk memperbaharui informasi dari unit sebelumnya. Informasi yang tidak diperlukan akan dihilangkan dan yang diperlukan akan dilanjutkan. *Update gate* diambil dari[11] dihitung mengunakan persamaan 2.

$$u^{<t>} = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{2}$$

$u^{<t>}$ merupakan hasil dari *update gate*. *Gate* ini akan membaca nilai $a^{<t-1>}$ merujuk pada *output* yang didapat dari modul sebelumnya dan $x^{<t>}$ yang merujuk pada *input* kata berbentuk vektor[18]. W_u merupakan bobot pada modul saat ini. Fungsi *sigmoid* yang digunakan pada persamaan 2 digunakan untuk mendapat nilai *update gate*.

Selanjutnya terdapat fungsi aktivasi tanh 3 untuk mendapatkan nilai $c^{<t>}$ yang dihitung menggunakan persamaan 3, dimana rumus dari tanh dapat dilihat pada persamaan 8.

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \tag{3}$$

$c^{<t>}$ merupakan hasil dari fungsi aktivasi tanh yang digunakan untuk memperbaharui nilai *memory cell* yaitu $c^{<t>}$.

Output gate berfungsi untuk memutuskan keluaran yang dihasilkan oleh satu modul LSTM. *output gate* diambil dari[11] dihitung mengunakan persamaan 4.

$$o^{<t>} = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{4}$$

$o^{<t>}$ merupakan hasil dari *output gate*. *Gate* ini membaca nilai $a^{<t-1>}$ merujuk pada *output* yang didapat dari modul sebelumnya dan $x^{<t>}$ yang merujuk pada *input* kata berbentuk vektor[18]. W_o merupakan bobot pada modul saat ini. Fungsi *sigmoid* yang digunakan pada persamaan4 digunakan digunakan untuk mendapat nilai *output gate*.

Selanjutnya terdapat fungsi aktivasi tanh untuk mendapatkan nilai $a^{<t>}$ menggunakan nilai dari *memory cell* yang paling baru[18], dimana rumus dari tanh dapat dilihat pada persamaan 8.

$$a^{<t>} = o^{<t>} * tanhc^{<t>} \tag{5}$$

$a^{<t>}$ merupakan hasil dari fungsi aktivasi tanh 5 yang digunakan sebagai hasil *output* dari satu modul LSTM.

Memory cell merupakan ide utama dari LSTM, dengan adanya *memory cell* maka dapat mengatasi masalah *vanishing gradient* dengan cara mengingat informasi kata yang letaknya jauh atau terletak di awal kalimat.

$$c^{<t>} = i^{<t>} * \tilde{c}^{<t>} + f^{<t>} * c^{<t-1>} \tag{6}$$

Untuk menentukan nilai dari *memory cell* digunakan persamaan 6, Nilai yang dibutuhkan untuk menentukan *memory cell* sebelumnya telah didapatkan pada *input gate*, *forget gate*, dan *output gate*.

Setelah vektor kata melalui semua modul yang ada pada setiap *layer* LSTM, selanjutnya sistem akan mengeluarkan nilai yang menetukan kelas pada setiap data.

3.4 Fungsi Aktivasi

Fungsi aktivasi adalah suatu proses matematika yang digunakan untuk mentransformasikan nilai *output* pada suatu neuron yang diteruskan ke *layer* selanjutnya. Terdapat banyak fungsi aktivasi yang dapat digunakan, dalam kasus ini digunakan fungsi aktivasi sigmoid, tanh dan relu.

Fungsi sigmoid digunakan pada setiap *gate* yang terdapat di modul LSTM dan pada *output layer*. Fungsi ini cocok digunakan pada kasus *binary classification*. Fungsi sigmoid menghasilkan nilai antara 0 hingga 1 yang diformulasikan dengan persamaan 7.

$$sig(x) = \frac{1}{1+e^{-x}} \tag{7}$$

Fungsi tanh digunakan untuk mengkalkulasikan kandidat vektor pada *memory cell* dan menghasilkan nilai sebenarnya dari satu modul. Fungsi tanh diformulasikan dengan persamaan 8.

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{8}$$

Fungsi relu adalah salah satu fungsi aktivasi yang paling umum digunakan pada jaringan saraf tiruan. Relu digunakan karena dapat membuat proses pelatihan menjadi lebih cepat. Fungsi relu adalah fungsi *linier* yang menampilkan *input* secara langsung jika yang dihasilkan bernilai positif, dan akan menghasilkan 0 jika yang dihasilkan bernilai negatif. Fungsi relu diformulasikan dengan persamaan 9.

$$R(x) = max(0,x) \tag{9}$$

3.5 Fungsi Loss

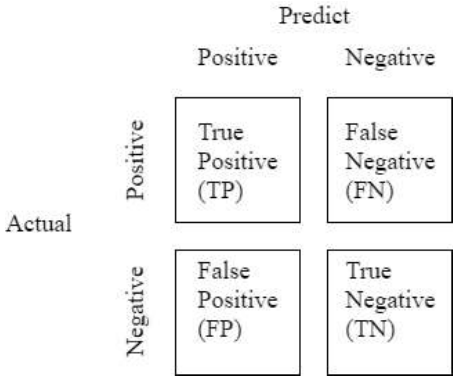
Fungsi loss adalah suatu metode untuk mengevaluasi model dalam memprediksi dataset. Jika prediksi yang dihasilkan tidak sesuai maka nilai loss yang dihasilkan tinggi. Sebaliknya, jika prediksi yang dihasilkan sesuai maka nilai loss yang dihasilkan rendah. Nilai loss juga digunakan untuk menentukan pembaharuan bobot pada saat proses *backpropagation*. Dalam kasus ini fungsi loss yang digunakan adalah *binary crossentropy*. Formula dari *Binary Crossentropy* terdapat pada persamaan 10.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log(p(y_i)) + (1-y_i) * \log(1-p(y_i)) \tag{10}$$

3.6 Metrics

Metrics merupakan suatu tabel yang digunakan untuk menentukan kualitas dari suatu model menggunakan nilai *accuracy*, *precision*, dan *recall*. Pada penelitian ini digunakan semua nilai untuk menentukan kualitas yang dihasilkan oleh sistem pendeteksi sarkasme yang dibangun menggunakan metode LSTM. Gambaran nilai *metrics* dapat dilihat pada gambar 4. Nilai *accuracy* merupakan nilai kedekatan antara hasil klasifikasi prediksi dan aktual, untuk menghitung nilai *accuracy* digunakan formula 11. Sedangkan nilai *precision* merupakan tingkat ketepatan

informasi yang dihasilkan oleh sistem, untuk menghitung nilai *precision* digunakan formula 12. Dan nilai *recall* merupakan tingkat keberhasilan sistem untuk menentukan kembali informasi, untuk menghitung nilai *recall* digunakan formula 13.



Gambar 4. **Confusion Metrics**

$$accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{11}$$

$$precision = \frac{TP}{TP+FP} \tag{12}$$

$$recall = \frac{TP}{TP+FN} \tag{13}$$

3.7 Pelabelan Otomatis

Pelabelan otomatis yang dilakukan pada penelitian ini digunakan untuk memperbaiki kualitas dari dataset yang sudah dilabeli secara *crowdsourcing*. Pelabelan otomatis digunakan menggunakan metode *naive bayes* seperti pada [22]. Dipilih 100 data terbaik dari dataset yang dijadikan sebagai data latih, dan 1.948 data sisanya digunakan sebagai data uji. Data yang digunakan sebagai data latih dipilih karena memiliki kriteria data yang baik. Kriteria yang baik artinya data yang memiliki kesesuaian antara data dengan label.

4. Evaluasi

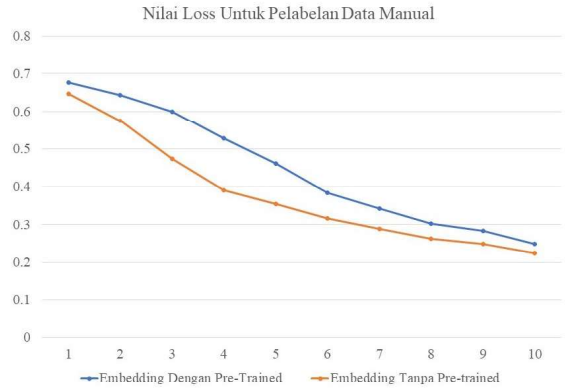
4.1 Hasil Pengujian

Pada bagian ini ditunjukkan hasil dari pengujian yang dilakukan. Pengujian deteksi sarkasme dilakukan dengan cara menghitung nilai *accuracy*, *precision*, dan *recall* yang didapat dari sistem yang telah dibangun. Dilakukan beberapa skenario untuk menentukan nilai akurasi yang didapat dari sistem pendeteksi sarkasme. Skenario pertama yang dijalankan adalah cara untuk mengatasi label dari dataset yang memiliki kualitas yang kurang baik. Dalam skenario kedua, dilihat pengaruh *word embedding* yang diterapkan.

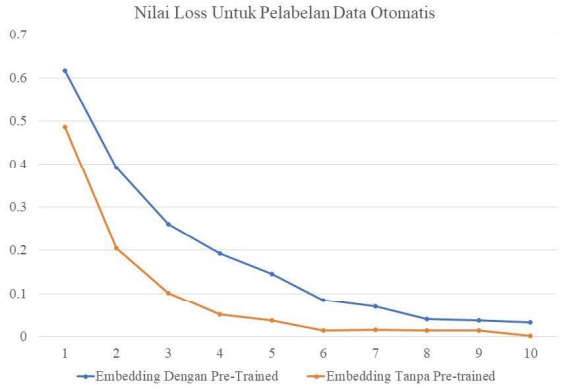
Untuk menjalankan skenario pertama yaitu melakukan pelabelan data secara otomatis dengan menggunakan metode *naive bayes*. Dan untuk menjalankan skenario kedua yaitu dengan cara membangun kamus embedding menggunakan data yang diambil dari twitter dengan topik pilpres 2019.

Tabel 7. **Hasil Metric Evaluasi**

Metric Evaluasi	Pelabelan Manual		Pelabelan Otomatis	
	Embedding Dengan Pre-Trained	Embedding Tanpa Pre-Trained	Embedding Dengan Pre-Trained	Embedding Tanpa Pre-Trained
Accuracy	62%	64%	93%	92%
Precision	44%	50%	92%	91%
Recall	24%	57%	95%	95%



Gambar 5. **Grafik Nilai Loss Untuk Pelabelan Dataset Secara Manual**



Gambar 6. **Grafik Nilai Loss Untuk Pelabelan Dataset Secara Otomatis**

4.2 Analisis Hasil Pengujian

Dari hasil nilai *accuracy*, *precision*, *recall* dan nilai *loss* dapat disimpulkan bahwa skenario dengan menggunakan dataset yang dilabeli secara otomatis dan dengan melakukan *pre-trained embedding* menggunakan data twitter yang berbeda dengan data klasifikasi, menghasilkan performansi yang lebih baik dibandingkan dengan skenario yang lain. Hal terbebut dikarenakan dataset yang dilabeli secara manual memiliki label yang konsisten dan sesuai dengan data, dapat dilihat pada tabel 8 perbandingan pelabelan dataset yang dilakukan secara manual dan otomatis. Sedangkan *pre-trained embedding* dilakukan agar kamus dari embedding memiliki bentuk yang serupa dengan dataset yang dimiliki untuk klasifikasi.

Dari hasil analisis yang didapat dilihat bahwa dataset yang akurat akan memberikan pengaruh yang sangat besar untuk sistem lebih dari pengaruh embedding. Maka dari itu pada skenario dilakukan proses pelabelan secara otomatis karena dataset yang dimiliki mempunyai kualitas yang kurang baik karena anotator yang kurang baik juga, maka hasil akurasi yang didapat kurang baik.

5. Kesimpulan

Untuk menyimpulkan hasil yang didapat dari penelitian yaitu, membuat model deteksi sarkasme untuk kasus pilpres 2019, menggunakan metode LSTM untuk membangun sistem, dan membangun dataset sarkasme dengan cara *crowdsourcing* dan *semi-supervised learning*.

Model dibuat menggunakan *pre-trained embedding* dilatih dengan dataset twitter dan 3 layer LSTM. Model yang dibangun berhasil menghasilkan nilai *accuracy* terbaik sebesar 93%, nilai *precision* sebesar 92%, dan nilai *recall* sebesar 95%. Nilai ini didapat dari skenario pelabelan data secara otomatis dan dengan embedding secara *pre-trained*. Dipilih 100 data dari dataset yang paling sesuai dengan labelnya. Kemudian data tersebut digunak-

Tabel 8. **Perbandingan Dataset Dengan Pelabelan Otomatis dan Manual**

Tweet	Pelabelan Manual	Pelabelan Otomatis
KIS - JKN produk andalan @jokowi utk kepentingan masyarakat luas! jika dlm prakteknya masih ada kekurangan disana sini, ya memang harus diakui itu ada! Lalu bukan berarti program ini tak jalan , apalagi asal2an, tentu harus didorong terus agar rakyat mendapatkan manfaat maksimal!	Ya	Tidak
Jokwoi bangun Tol Langit #2019JokowiKyaiMaruf #JokowiMembangun1	Ya	Tidak
Lawan Tagar #02Nyabu, Netizen Paparkan Bukti Telak #01BandarSabu #2019GantiPresiden #2019PrabowoPresidenRI #17April2019GantiPresiden	Tidak	Ya
Apa-apa salah jokowi.. berikut illustrasinya #fakkampret	Tidak	Ya

an untuk melatih model *naive bayes* dalam melakukan pelabelan otomatis pada data lain yang ada pada dataset. *Pre-trained embedding* dilakukan dengan membangun kamus embedding menggunakan data dari twitter yang berkaitan dengan pilpres 2019, terdapat sebanyak 2994 tweet yang terkumpul untuk membangun kamus embedding.

Daftar Pustaka

[1] A. D. ARUMSARI. *Analisis Sentimen pada Tweet Indonesia Menggunakan Recurrent Convolutional Neural Network*. PhD thesis, Universitas Gadjah Mada, 2017.

[2] D. Bamman and N. A. Smith. Contextualized sarcasm detection on twitter. In *Ninth International AAAI Conference on Web and Social Media*, 2015.

[3] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[4] S. Bharti, B. Vachha, R. Pradhan, K. Babu, and S. Jena. Sarcastic sentiment detection in tweets streamed in real time: a big data approach. *Digital Communications and Networks*, 2(3):108–121, 2016.

[5] M. Bouazizi and T. Ohtsuki. Sarcasm detection in twitter:" all your products are incredibly amazing!!!"-are they really? In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

[6] S. Exchanged. Ratio between embedded vector dimensions and vocabulary size. `https://datascience.stackexchange.com/questions/31109/ratio-between-embedded-vector-dimensions-and-vocabulary-size`, 2018. Online; Accessed 30 Juni 2019.

[7] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[8] E. Greenstein and D. Penner. Japanese-to-english machine translation using recurrent neural networks. 2015.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] A. Karpathy. The unreasonable effectiveness of recurrent neural networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`, 2015. Online; Accessed 4 November 2018.

[11] A. Karpathy. Long short term memory (lstm) - recurrent neural networks. `https://www.coursera.org/lecture/nlp-sequence-models/long-short-term-memory-lstm-KXoay`, 2018. Online; Accessed 18 October 2018.

[12] K. Kawakami. *Supervised sequence labelling with recurrent neural networks*. PhD thesis, Ph. D. thesis, Technical University of Munich, 2008.

[13] E. Lunando and A. Purwarianti. Indonesian social media sentiment analysis with sarcasm detection. In *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 195–198. IEEE, 2013.

[14] H. Lyu et al. *Sarcasm detection on Twitter*. PhD thesis, 2016.

[15] D. Maynard and M. A. Greenwood. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *LREC 2014 Proceedings*. ELRA, 2014.

[16] S. Mujilahwati. Pre-processing text mining pada data twitter. *Semin. Nas. Teknol. Inf. dan Komun*, 2016(Sentika):2089–9815, 2016.

[17] A. A. Nugraha, A. Arifianto, and S. Suyanto. Generating image description on indonesian language using convolutional neural network and gated recurrent unit. In *2019 7th International Conference on Information and Communication Technology (ICoICT 2019)*, 2019.

[18] C. Olah. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. Online; Accessed 5 November 2018.

[19] N. Parde and R. Nielsen. Detecting sarcasm is extremely easy;-. In *Proceedings of the Workshop on Computational Semantics beyond Events and Roles*, pages 21–26, 2018.

[20] T. Ptáček, I. Habernal, and J. Hong. Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 213–223, 2014.

[21] J.-H. Wang, T.-W. Liu, X. Luo, and L. Wang. An lstm approach to short text sentiment classification with word embeddings. In *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing (ROCLING 2018)*, pages 214–223, 2018.

[22] A. F. Wicaksono, C. Vania, B. Distiawan, and M. Adriani. Automatically building a corpus for sentiment analysis on indonesian tweets. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing*, pages 185–194, 2014.

[23] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*, 2016.

Lampiran