# Perbandingan Metode *Word Embedding* untuk Klasifikasi *Name Entity Recognition* (NER) pada Teks Berita Bahasa Indonesia

Narandha Arya Ranggianto, Swardiantara Silalahi

## 1. Latar Belakang

*Named Entity Recognition* (NER) merupakan bagian dari *Natural Language Processing* (NLP) yang digunakan untuk mengenali sebuah entitas pada kata seperti nama orang, lokasi, organisasi, dsb. NER melakukan ekstraksi informasi untuk menyelesaikan tugas-tugas pada domain NLP seperti *question answering*, *information retrieval*, *topic modelling*, dsb. Proses identifikasi untuk mengenali entitas secara manual akan mengalami keterbatasan ketika menghadapi data yang banyak dengan kata atau bahasa yang beragam. Oleh karena itu, tugas tersebut dapat dilakukan secara otomatis dengan menggunakan pendekatan *deep learning*. Proses identifikasi NER merupakan sebuah kasus model pelabelan secara sekuensial sehingga bisa dilakukan dengan teknik klasifikasi. Klasifikasi sederhana yang biasa digunakan yaitu RNN, LSTM, GRU, dan CNN [1].

Proses identifikasi NER dilakukan dengan mengubah kata menjadi sebuah vektor agar dapat diproses oleh mesin. Salah satu teknik representasi yang digunakan adalah *word embedding. Word embedding* mampu mengurangi beban terhadap data dengan sumber daya yang rendah. Hal itu dikarenakan sifatnya yang mampu membangun fungsi semantik dari setiap kata yang nantinya akan membantu proses identifikasi melalui kedekatan nilai vektor dari setiap kata. Beberapa metode *word embedding* yang biasa digunakan yaitu Word2Vec, GloVe, dan fastText [1]. Dari metode tersebut, kami melakukan perbandingan antara metode *word embedding* untuk digunakan pada model *deep learning*.

## 2. Tujuan

Beberapa tujuan yang ingin penulis capai dari uji coba ini adalah sebagai berikut:

1. Mendapatkan metode *word embedding* untuk menghasilkan performa terbaik pada klasifikasi NER Berita Bahasa Indonesia
2. Mengaplikasikan metode klasifikasi deep learning sederhana yaitu RNN, LSTM, GRU, dan CNN untuk klasifikasi NER Berita Bahasa Indonesia

## 3. Dataset

Dataset yang digunakan yaitu dataset publik yang dikeluarkan pada tahun 2016 oleh Syaifudin dan Nurwidyantoro, yang dianotasi ulang oleh Khairunnisa, Imankulova dan Komachi dengan jumlah data teks sebanyak 48699 [2]. Dataset ini merupakan dataset berita bahasa Indonesia yang telah dianotasi setiap kata untuk pengenalan entitasnya (NER). Jumlah anotasi pada dataset tersebut yaitu 7 yang terdiri dari 'B-LOC', 'B-ORG', 'B-PER', 'I-LOC', 'I-ORG', 'I-PER', dan 'O'. Anotasi 'LOC' sebagai lokasi, 'ORG' sebagai organisasi, 'PER' sebagai nama orang, 'O' sebagai *other*. Dataset dibagi menjadi data *training* 30248 dan data *testing* 18451. Secara detail statistika data ditunjukkan pada Tabel 1.

Tabel 1. Statistika Data

| **Pembagian Data** | **Kalimat** | ***Tokens*** |
|---|---|---|
| Train | 1464 | 30248 |
| Test | 509 | 18451 |
| Total | 6554 | 48699 |

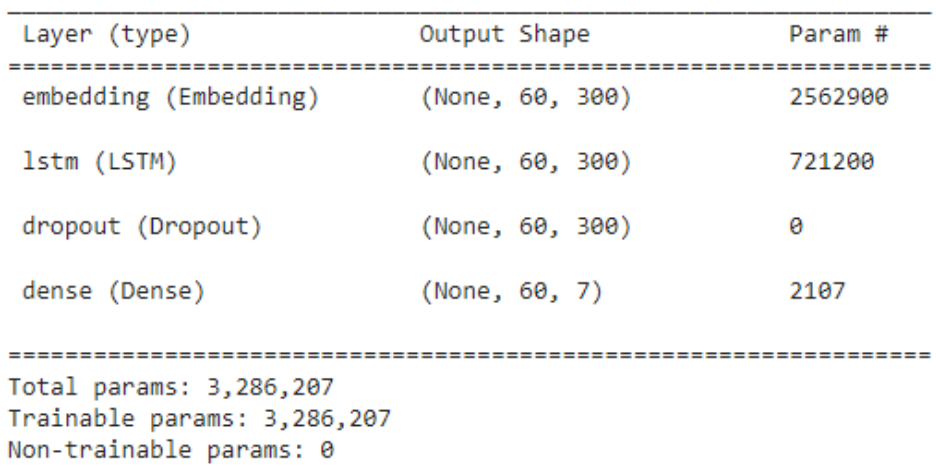
## 4. Metode yang Digunakan

Proses mengenali dan memprediksi entitas dan POS (Part of Speech) pada kalimat merupakan permasalahan klasifikasi pada data sekuensial. Beberapa metode *deep learning* yang dapat digunakan adalah RNN, LSTM, dan GRU. Selain itu, CNN juga digunakan pada uji coba ini. Data sekuensial yang digunakan berupa teks seperti pada Gambar 1, sehingga perlu dikonversi menjadi vektor numerik sebagai representasi setiap kata dengan menggunakan metode *word embedding* seperti GloVe, word2vec, dan fastText. Sebelum mengubah bentuk kata menjadi vektor numerik, kata diubah menjadi *index* kata pada *vocabulary*, kemudian menyeragamkan panjang kata dengan melakukan *padding*. Ukuran *padding* yang dipilih adalah 60 setelah melihat distribusi panjang kalimat pada dataset. Untuk mengimplementasikan metode-metode yang disebutkan di atas, digunakan *library* Keras dari Tensorflow [3].

| word | tag |
|---|---|
| [Berikut, adalah, tujuh, kota, di, Indonesia, ... | [O, O, O, O, O, B-LOC, O, O, O, O, O, O, O] |
| [Soal, calon, presiden, itu, urusan, nanti, „... | [O, O, O, O, O, O, O, O, O, O, O, O, O, B-P... |
| [Kalau, itu, tanya, PPP, saja, „ ujar, Nasir,... | [O, O, O, B-ORG, O, O, O, B-PER, O, O, O, O, O... |
| [Jokowi, bisa, mengatakan, mencabut, pengaduan... | [B-PER, O, O, O, O, O, O, O, O, O, O] |
| [Akbar, menyebutkan, pemerintahan, Jokowi-JK, ... | [B-PER, O, O, O, O, O, O, O, O, O, O, O, O,... |

Gambar 1. Sekuens kata per kalimat

Pemilihan parameter untuk setiap metode menggunakan beberapa referensi yang diambil dari berbagai sumber. Arsitektur pada RNN, LSTM dan GRU terinspirasi dari struktur yang dibuat oleh Jaswani[1] pada platform Kaggle. Struktur model dapat dilihat pada Gambar 2. Layer ke-2 akan menyesuaikan model yang digunakan. Pada gambar terlihat LSTM, maka jika pada saat proses train model CNN, layer tersebut diganti menjadi layer konvolusi. Pengaturan parameter pada setiap model ditampilkan pada Tabel 2.



| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding (Embedding) | (None, 60, 300) | 2562900 |
| lstm (LSTM) | (None, 60, 300) | 721200 |
| dropout (Dropout) | (None, 60, 300) | 0 |
| dense (Dense) | (None, 60, 7) | 2107 |

Total params: 3,286,207
Trainable params: 3,286,207
Non-trainable params: 0

Gambar 2. Arsitektur Model Klasifikasi

Parameter pada model *word embedding* yang digunakan merujuk pada pengaturan parameter pengembang awal[2][3]. Ukuran vektor yang dipilih adalah 300, setelah melakukan percobaan menggunakan ukuran 100, 200 dan 300 pada saat proses training model LSTM. Namun beberapa parameter disesuaikan dengan kondisi dataset yang digunakan. Dikarenakan dataset tidak terlalu besar, maka saat melakukan training model *word embedding*, diperlukan iterasi yang besar (100). Ukuran *skip-gram* yang digunakan adalah 5 [4], dengan minimum count untuk kata yang akan dimasukkan ke *vocabulary* adalah 1. Khusus untuk *word embedding* Keras, iterasi yang digunakan mengikuti *epoch* pada model klasifikasi, yaitu 25.

Tabel 2. *Hyperparameter* Model Klasifikasi

| Parameter | Nilai |
|---|---|
| Epoch | 25 |
| Activation | softmax |
| Units | 300 |

[1] https://www.kaggle.com/namanj27/ner-dataset/code
[2] https://github.com/stanfordnlp/GloVe
[3] https://github.com/RaRe-Technologies/gensim/tree/develop/gensim/models

| Parameter | Nilai |
|---|---|
| Dropout | 0.5 |
| Optimizer | Adam (lr = 0.01) |
| Loss | Sparse categorical crossentropy |
| Evaluation metrics | Accuracy, Precision, Recall, F1 |

Nilai *precision*, *recall* dan *F1-score* dihitung menggunakan pendekatan *Macro Average*, yaitu dengan menghitung rata-rata dari *precision*, *recall* dan *F1-score* setiap kelas. Pendekatan ini adalah salah satu cara untuk menghitung nilai *precision*, *recall* dan *F1-score* pada kasus *multiclass classification*. Perhitungan dilakukan dengan bantuan *library* Scikit-learn [5].

## 5. Hasil dan Analisis Ujicoba

Proses training dilakukan dengan mengkombinasikan setiap metode *word embedding* ke setiap model klasifikasi. Untuk mendapatkan *embedding matrix* dari *embedding layer* Keras, pada saat proses training model LSTM, *output* dari *embedding layer* disimpan sebagai vektor numberik dari kata untuk digunakan pada saat proses training model lainnya. Sebelum memutuskan untuk menggunakan 300 sebagai ukuran dimensi vektor kata, dilakukan percobaan sebanyak 3 kali dengan ukuran 100, 200 dan 300 masing-masing percobaan. Hasil evaluasi dari model klasifikasi yang dibangun dapat dilihat pada Tabel 3 – Tabel 7.

Tabel 3. Hasil Evaluasi Glove dengan LSTM

| Metode | LSTM | | | |
|---|---|---|---|---|
| | Accuracy | Precision | Recall | F1-score |
| **Glove 100** | 75,792% | 32,560% | 74,626% | 40,662% |
| **Glove 200** | 75,684% | 35,377% | **76,620%** | 43,932% |
| **Glove 300** | **78,952%** | **36,573%** | 73,735% | **45,247%** |

Tabel 4. Hasil Evaluasi *Word Embedding* dengan RNN

| Metode | RNN | | | |
|---|---|---|---|---|
| | Accuracy | Precision | Recall | F1-score |
| **Keras Embedding** | **89,247%** | **49,120%** | **73,290%** | **53,950%** |
| **Glove** | 14,316% | 21,386% | 52,692% | 14,494% |
| **Word2Vec** | 44,057% | 17,802% | 29,373% | 14,270% |

| Metode | RNN | | | |
|---|---|---|---|---|
| | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| **Fasttext** | 3,595% | 17,266% | 24,678% | 4,449% |

Tabel 5. Hasil Evaluasi *Word Embedding* dengan LSTM

| Metode | LSTM | | | |
|---|---|---|---|---|
| | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| **Keras Embedding** | **95,711%** | **72,985%** | 78,204% | **73,802%** |
| **Glove** | 78,952% | 36,573% | 73,735% | 45,247% |
| **Word2Vec** | 86,408% | 43,331% | **85,861%** | 54,767% |
| **Fasttext** | 86,863% | 43,493% | 85,510% | 54,830% |

Tabel 6. Hasil Evaluasi *Word Embedding* dengan GRU

| Metode | GRU | | | |
|---|---|---|---|---|
| | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| **Keras Embedding** | **94,270%** | **65,238%** | **71,921%** | **65,587%** |
| **Glove** | 69,368% | 31,122% | 59,464% | 32,304% |
| **Word2vec** | 57,800% | 23,706% | 62,802% | 25,905% |
| **Fasttext** | 62,515% | 29,500% | 60,281% | 30,486% |

Tabel 7. Hasil Evaluasi *Word Embedding* dengan CNN

| Metode | CNN | | | |
|---|---|---|---|---|
| | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| **Keras Embedding** | **95,246%** | **67,477%** | 75,931% | **69,071%** |
| **Glove** | 75,927% | 30,417% | 71,465% | 37,769% |
| **Word2Vec** | 88,340% | 44,250% | 81,504% | 54,990% |
| **Fasttext** | 86,916% | 42,596% | **81,704%** | 53,525% |

Seperti yang terlihat pada Tabel 3 – Tabel 7, model *word embedding* Keras memiliki performa yang mengungguli tiga model *word embedding* lainnya jika merujuk pada akurasi, presisi dan F1-score model. Menurut analisis penulis, hal ini disebabkan oleh algoritma yang digunakan pada proses training *embedding layer* Keras adalah algoritma Backpropagation, dengan menggunakan informasi kelas (dalam hal ini tag entitas) sebagai acuan dalam penghitungan bobot matrix setiap kata

(*Supervised Learning*), sehingga bobot matrix yang dihasilkan membuat proses pembelajaran model klasifikasi lebih cepat mendapatkan bobot terbaik untuk setiap kata. Dengan demikian, bobot matrix yang dimiliki oleh kata dengan tag yang sama, akan memiliki kedekatan antar satu kata dengan kata lainnya. Pada Gambar 3 dapat dilihat *confusion matrix* dari model LSTM dengan menggunakan *word embedding* Keras. Jika dilihat total data pada gambar, terlihat tidak sinkron dengan jumlah kata pada data test. Hal ini dikarenakan *confusion matrix* dibangun dari hasil prediksi terhadap data *test* yang sudah melalui proses *padding*. Sehingga jumlah *token* menjadi sebesar $ukuran\ padding$ x $jumlah\ kalimat$.

| actual_class<br>predicted_class | B-LOC | B-ORG | B-PER | I-LOC | I-ORG | I-PER | O |
|---|---|---|---|---|---|---|---|
| B-LOC | 229 | 9 | 5 | 12 | 3 | 1 | 152 |
| B-ORG | 3 | 484 | 17 | 1 | 17 | 14 | 203 |
| B-PER | 1 | 2 | 416 | 2 | 1 | 29 | 52 |
| I-LOC | 5 | 2 | 5 | 122 | 9 | 6 | 46 |
| I-ORG | 1 | 6 | 25 | 4 | 341 | 12 | 170 |
| I-PER | 1 | 0 | 8 | 3 | 0 | 188 | 21 |
| O | 13 | 77 | 244 | 15 | 37 | 76 | 27450 |

Gambar 3. *Confusion Matrix* LSTM + *Embedding* Keras

Berbeda dengan tiga metode *word embedding* lainnya yang menggunakan pendekatan *Self-supervised learning*, dimana informasi kelas tidak digunakan dalam menghitung bobot matrix setiap kata. Analisis penulis terhadap kurang baiknya performa model GloVe, word2vec dan fastText dikarenakan ketiga model ini menggunakan statistik dari kata dan kelompok kata sebagai metode untuk men-*generate* label yang digunakan saat proses *training*, membuat ketiga model ini bergantung pada ukuran *corpus*. Sementara, ukuran *corpus* yang digunakan pada uji coba ini relatif kecil. Terutama bagi GloVe, yang tidak hanya menggunakan *local statistic*, tapi juga *global statistic*, sangat bergantung pada ukuran *vocabulary* dan *corpus* yang digunakan saat training [6]. Meskipun sama-sama menggunakan Backpropagation, ketiga model ini tidak melibatkan informasi kelas sebagai acuan saat

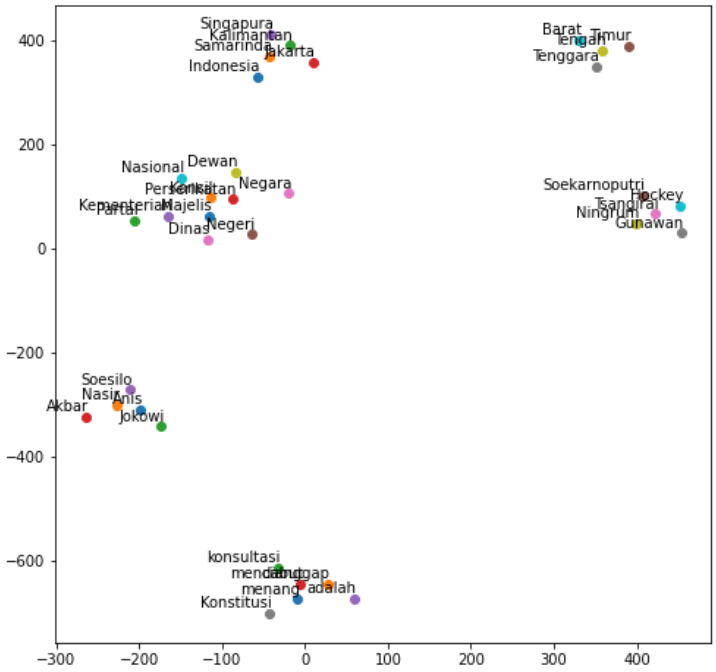
menghitung *loss* dan meng-*update* bobot. Acuan yang digunakan saat proses training adalah label yang dibuat sendiri, yaitu dari konteks kata yang terdapat pada sebelum dan seduah kata, yang jumlahnya bergantung pada ukuran *window* yang dipilih[4]. Sementara, *task* NER adalah tentang memprediksi tag dari satu kata. Sehingga metode Keras *embedding* menghasilkan *embedding matrix* yang sangat menguntungkan model klasifikasi.

Berikut adalah visualisasi dari sejumlah kata yang diambil secara acak dari *data train* sebanyak lima kata dari masing-masing kelas untuk melihat kedekatan representasi numerik dari satu kata dengan kata lainnya. Visualisasi berikut merupakan hasil perhitungan dari model *embedding* Keras, untuk melihat mengapa model *embedding* Keras lebih unggul dibandingkan tiga metode *word embedding* lainnya.

Tabel 8. Sampel Kata untuk Visualisasi Representasi

| Kata | Tag |
|---|---|
| Indonesia, Samarinda, Kalimantan, Jakarta, Singapura | B-LOC |
| Timur, Negara, Tenggara, Tengah, Barat | I-LOC |
| Majelis, Konsil, Partai, Perserikatan, Kementerian | B-ORG |
| Negeri, Dinas, Konstitusi, Dewan, Nasional | I-ORG |
| Anis, Nasir, Jokowi, Akbar, Soesilo | B-PER |
| Soekarnoputri, Tsangirai, Gunawan, Ningrum, Hockey | I-PER |
| Menang, dianggap, konsultasi, mencabut, adalah | O |

[4] https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314




Gambar 4. Visualisasi Representasi Kata dari *Embedding* Keras

Dari gambar di atas, terlihat bahwa Keras embedding dapat menghasilkan vektor numberik yang berdekatan untuk kata yang memiliki tag yang sama. Meskipun demikian, terlihat beberapa kata lebih dekat ke kelompok kata dengan tag lain. Hal ini merupakan kesalahan dari data tersebut, yaitu kesalahan pelabelan. Seperti yang dapat dilihat pada kata "Negara" dengan tag "I-LOC", namun Keras embedding memberikan vektor numerik yang lebih dekat ke kelas "B-ORG" ataupun "I-ORG". Sehingga, visualisasi hanya menunjukkan 6 kelompok data saja, walaupun kata-kata tersebut berasal dari 7 kelas yang berbeda.

## 6. Kesimpulan dan Saran

Studi perbandingan penggunaan metode *word embedding* pada performa model *deep learning* untuk task NER telah dilakukan. Digunakan empat metode *word embedding* dan empat model klasifikasi. Proses training pada model *word embedding* dan model klasifikasi dilakukan dengan menggunakan parameter yang sama. Hasil dari uji coba menunjukkan bahwa Keras *embedding* mampu unggul dibandingkan dengan tiga metode *word embedding* lainnya dilihat dari tiga metrik evaluasi, yaitu *accuracy*, *precision* dan *F1-score*. Hal ini dikarenakan Keras *embedding* menggunakan teknik *supervised* untuk menghitung *embedding matrix*, sehingga menguntungkan model klasifikasi dalam melakukan deteksi entitas. Pada uji coba ini, tidak dilakukan *search* dalam menentukan parameter pada model klasifikasi, sehingga

perbandingan performa model kemungkinan bukan membandingkan performa optimal masing-masing model. Pemilihan model word embedding dan model klasifikasi didasarkan hanya pada tingkat popularitas dan relevansi arsitektur dengan permasalahan (data sekuensial). Ukuran corpus pada uji coba ini relatif kecil, sehingga model *embedding* yang bergantung pada ukuran corpus menghasilkan matriks yang kurang optimal.

Sebagai saran dari penelitian ini, sebelum menentukan parameter model klasifikasi, sebaiknya dilakukan *hyperparameter search* dan *tuning* setelah melakukan *training*. Jika ukuran corpus dinilai kurang cukup, sebaiknya menggunakan model *word embedding* yang sudah dilatih menggunakan *corpus* yang cukup besar, sehingga *embedding matrix* yang dihasilkan lebih baik. Pada saat evaluasi, mencari cara lain yang lebih representatif terhadap kondisi permasalahan, seperti menggunakan *cost-sensitive evaluation*. Sehingga nilai-nilai metrik evaluasi lebih kontekstual dan merepresentasikan permasalahan. Pada kasus ini, komposisi kelas sangat tidak seimbang, sehingga nilai *precision*, *recall* dan *F1* secara umum masih berada di bawah 80%. Kode sumber dari uji coba ini dapat ditemukan di github[5].

## Referensi


[1] J. Li, A. Sun, J. Han, and C. Li, "A Survey on Deep Learning for Named Entity Recognition," *CoRR*, vol. abs/1812.0, 2018, [Online]. Available: http://arxiv.org/abs/1812.09449.

[2] S. O. Khairunnisa, A. Imankulova, and M. Komachi, "Towards a Standardized Dataset on Indonesian Named Entity Recognition," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop*, 2020, pp. 64–71.

[3] Martin Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." 2015, [Online]. Available: https://www.tensorflow.org/.

[4] C. N. Dos Santos and B. Zadrozny, "Learning Character-Level Representations for Part-of-Speech Tagging," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, 2014, pp. II–1818–II–1826.

[5] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[6] S. M. Rezaeinia, A. Ghodsi, and R. Rahmani, "Improving the accuracy of pre-trained word embeddings for sentiment analysis," *arXiv Prepr. arXiv1711.08609*, 2017.


[5] https://github.com/swardiantara/fp-dm-2021