PERBANDINGAN KINERJA *WORD EMBEDDING* WORD2VEC, GLOVE, DAN FASTTEXT PADA KLASIFIKASI TEKS

Arliyanti Nurdin[1)], Bernadus Anggo Seno Aji[2)], Anugrayani Bustamin[3)], Zaenal Abidin[4)]

[1), 2)] Teknologi Informasi, Institut Teknologi Telkom Surabaya
[3)] Universitas Hasanuddin
[4)] Universitas Teknokrat Indonesia
Jl. Gayungan PTT no 17-19, Surabaya
Email : arliyanti.n@ittelkom-sby.ac.id [1)], bernadus.seno@ittelkom-sby.ac.id [2)], anugrayani@unhas.ac.id[3)] zabin@teknokrat.ac.id [4)]

Abstrak

Karakteristik teks yang tidak terstruktur menjadi tantangan dalam ekstraksi fitur pada bidang pemrosesan teks. Penelitian ini bertujuan untuk membandingkan kinerja dari word embedding seperti Word2Vec, GloVe dan FastText dan diklasifikasikan dengan algoritma Convolutional Neural Network. Ketiga metode ini dipilih karena dapat menangkap makna semantik, sintatik, dan urutan bahkan konteks di sekitar kata jika dibandingkan dengan feature engineering tradisional seperti Bag of Words. Proses word embedding dari metode tersebut akan dibandingkan kinerjanya pada klasifikasi berita dari dataset 20 newsgroup dan Reuters Newswire. Evaluasi kinerja diukur menggunakan F-measure. Performa terbaik menunjukkan FastText unggul dibanding dua metode word embedding lainnya dengan nilai F-Measure sebesar 0.979 untuk dataset 20 Newsgroup dan 0.715 untuk Reuters. Namun, perbedaan kinerja yang tidak begitu signifikan antar ketiga word embedding tersebut menunjukkan bahwa ketiga word embedding ini memiliki kinerja yang kompetitif. Penggunaannya sangat bergantung pada dataset yang digunakan dan permasalahan yang ingin diselesaikan

***Kata kunci: w**ord embedding, word2vec, glove, fasttext, klasfikasi teks, convolutional neural network, cnn.*

1. Pendahuluan

Pada bidang pemrosesan teks atau *Natural Language Processing* (NLP), teknik representasi kata ke dalam vektor menjadi topik menarik dalam penelitian yang terus dikembangkan. Representasi ini menjadi sangat penting karena akan berdampak signifikan terhadap akurasi atau kinerja dari model *learning* yang dibangun. *Feature Engineering* dalam data tekstual memiliki tantangan tersediri karena karakteristik dari teks yang tidak terstruktur.

Strategi *feature engineering* tradisional untuk data tekstual yang popular digunakan dikenal sebagai model *Bag of Words*, yang terdiri dari *term frequencies* (TF), *term frequency-inverse document frequency* (TF-IDF), hingga *n-grams. Bag of Words* merepresentasikan teks ke dalam sekumpulan kata yang dapat diwakili oleh frekuensi kemunculannya dalam sebuah dokumen. Kelemahan dari *Bag of Words* ini adalah ketidakmampuannya memberikan informasi terkait semantik, struktur, urutan, dan konteks di sekitar kata dalam setiap dokumen. *Count based* model seperti *Bag of Words* melibatkan setiap kata secara individual dan tidak menangkap hubungan semantik antarkata. Ketika kosakata dalam korpus sangat banyak tetapi frekuensi kemunculan kata dalam dokumen sangat kecil atau bahkan nol, maka representasi vektornya akan sebagian besar bernilai nol. *Sparse word vector* yang terbentuk sangat berpeluang manghasilkan model yang buruk atau bahkan mengakibatkan *overfitting*.

Kemudian pada sekitar tahun 2000, mulai dikembangkan teknik *word embedding* (Bengio *et al.*, 2003; Mikolov *et al.*, 2013; Pennington, Socher and Manning, 2014; Bojanowski *et al.*, 2017). *Word embedding* memetakan setiap kata dalam dokumen ke dalam *dense vector*, di mana sebuah vektor merepresentasikan proyeksi kata di dalam ruang vektor. Posisi kata tersebut dipelajari dari teks atau berdasarkan kata-kata di sekitarnya. *Word embedding* ini dapat menangkap makna semantik dan sintaktik kata.

Pada tahun 2013, Mikolov dkk. memperkenalkan word2vec dengan dua metode utamanya yaitu Skip-gram dan Continous Bag of Words (CBOW) (Mikolov *et al.*, 2013). *Word embedding* salah satu aplikasi *unsupervised learning* menggunakan Deep Neural Network. Kemudian pada tahun berikutnya, diperkenalkan *word embedding* baru oleh Pennington dkk. (Pennington, Socher and Manning, 2014) yaitu GloVe yang menggunakan rasio *co-occurrence probability* antarkata. Kemudian pada tahun 2017, Bojaniwski dkk. (Bojanowski *et al.*, 2017) mengembangkan model *word2vec* dan memperkenalkan FastText yang mempelajari informasi *subword* dari kata.

Semakian populernya penggunaan *word embedding* digunakan dalam bidang NLP, menjadi motivasi untuk melakukan perbandingan kinerja dari setiap model *word embedding* yang ada.

WORD2VEC

Word2vec merupakan salah satu algoritma *word embedding* yang memetakan setiap kata dalam teks ke dalam vektor. Algoritma word2vec ini diciptakan oleh Mikolov dkk. pada tahun 2013. Sejak kemunculannya, model *word embedding* ini banyak digunakan dalam penelitian NLP.

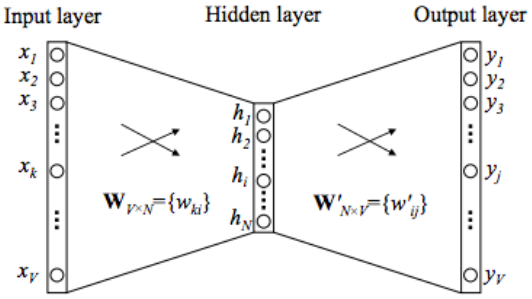
Word2vec merepresentasikan kata ke dalam vektor yang dapat membawa makna semantik dari kata tersebut. Model *word embedding* ini merupakan salah satu aplikasi *unsupervised learning* menggunakan neural network yang terdiri dari sebuah *hidden layer* dan *fully connected layer*. Dimensi dari matriks bobot pada setiap layer adalah jumlah dengan kata dalam korpus dikalikan dengan jumlah *hidden neuron* pada *hidden layer*-nya. Matriks bobot pada *hidden layer* dari model yang telah dilatih digunakan untuk mentransformasikan kata ke dalam vektor. Matriks bobot ini seprti *lookup table*, di mana setiap baris mewakili setiap kata dan kolom mewakili vektor dari kata tersebut.

Wod2vec mengandalkan informasi lokal dari bahasa. Semantik yang dipelajari dari kata tertentu dipengaruhi oleh kata-kata sekitarnya. Model ini mendemonstrasikan kemampuan untuk mempelajari pola linguistik sebagai hubungan linear antarvektor kata.

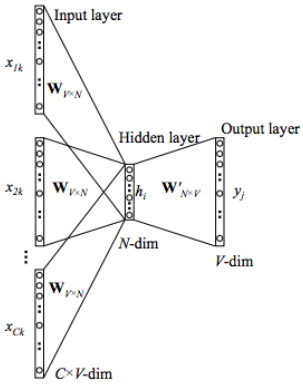
Terdapat dua algoritma word2vec yaitu Continuous Bag-of-Word (CBOW) dan Skip-gram.

a. CBOW

Model ini menggunakan konteks untuk memprediksi target kata. CBOW memiliki waktu training lebih cepat dan memiliki akurasi yang sedikit lebih baik untuk *frequent words*.



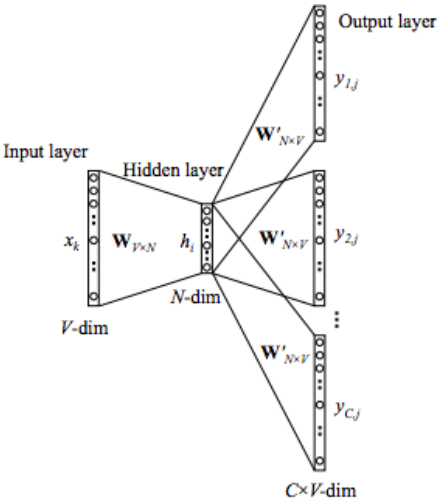
Gambar 1 *One Word Context CBOW(Rong, 2016)*



Gambar 2 *Mutiple Context Words CBOW (Rong, 2016)*

b. Skip-Gram

Model ini menggunakan sebuah kata untuk memprediksi target konteks. Skip-Gram bekerja dengan baik dengan data pelatihan yang jumlahnya sedikit dan dapat merepresentasikan kata-kata yang dianggap langka.



Gambar 3 *Skip-Gram (Rong, 2016)*

GLOVE

Berbeda dengan word2vec yang hanya mengandalkan informasi lokal dari kata dengan *local context window* (CBOW dan Skip-gram), algoritma GloVe juga menggabungkan informasi *co-occurrence* kata atau statistik global untuk memperoleh hubungan semantik antarkata dalam korpus. GloVe menggunakan metode *global matrix factorization*, matriks yang mewakili kemunculan atau ketiadaan kata-kata dalam suatu dokumen (Pennington, Socher and Manning, 2014).

Word2vec adalah model *feedforward neural network* sehingga sering disebut sebagai *neural word embeddings*, sedangkan GloVe adalah model log-bilinear atau secara sederhana dapat disebut sebagai model berbasis hitungan. GloVe mempelajari hubungan kata-kata dengan menghitung seberapa sering kata-kata muncul bersama satu sama lain dalam sebuah korpus yang diberikan. Rasio probabilitas kemunculan kata-kata memiliki potensi untuk mengkodekan beberapa bentuk makna serta membantu meningkatkan kinerja pada permasalahan analogi kata.

Pelatihan model GloVe bertujuan untuk mempelajari vektor kata sedemikian rupa sehingga *dot product* kata-kata tersebut sama dengan logaritma probabilitas kata-kata untuk muncul bersama atau probabilitas *co-occurence* nya.

Algoritma GloVe terdiri dari langkah – langkah berikut (Selivanov, 2020):

1. Mengumpulkan statistik *word co-occurence* dalam bentuk sebuah matriks *word co-occurence X*. Setiap elemen X_{ij} merepresentasikan berapa kali kata *i* muncul dalam konteks kata *j*.

2. Tentukan *soft constraints* untuk setiap pasangan kata :

$$w_i^T w_j + b_i + b_j = \log(X_{ij}) \quad (1)$$

Di mana w_i – vektor kata utama, w_j- vektor kata konteks, b_i,b_j bias skalar untuk kata-kata utama dan kata-kata konteks.

3. Tentukan sebuah *cost function.*

$$J = \sum_{i=1}^{V}\sum_{j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - logX_{ij})^2 \quad (2)$$

Di mana *f* fungsi pembobotan yang membantu kita mencegah belajar hanya dari pasangan kata yang sangat umum. Fungsi tersebut didefinisikan sebagai berikut:

$$f(X_{ij}) = \begin{cases} (\frac{X_{ij}}{x_{max}})^{\alpha} & if\ X_{ij} < XMAX \\ 1 & otherwise \end{cases} \quad (3)$$

FASTTEXT

FastText (Bojanowski *et al.*, 2017) adalah metode *word embedding* yang merupakan pengembangan dari *word2vec*. Metode ini mempelajari representasi kata dengan mempertimbangkan informasi *subword*. Setiap kata direpresentasikan sebagai sekumpulan karakter *n-gram*. Dengan demikian dapat membantu menangkap arti kata-kata yang lebih pendek dan memungkinkan *embedding* untuk memahami sufiks dan prefiks dari kata. Representasi vektor dikaitkan dengan setiap karakter *n-gram*, sedangkan kata-kata direpresentasikan sebagai jumlah dari representasi vektor tersebut. Setelah kata direpresentasikan dengan karakter *n-gram*, model Skip-gram dilatih untuk mempelajari *embedding* vektor dari kata.

Pada umumnya model yang mempelajari representasi kata ke dalam vektor mengabaikan morfologi kata, setiap kata memiliki vektor yang berbeda. Hal ini menjadi keterbatasan untuk merepresentasikan kata dari bahasa dengan kosakata yang besar dan memiliki banyak kata-kata langka.

FastText memiliki kinerja yang baik, dapat melatih model pada dataset yang besar dengan cepat dan dapat memberikan representasi kata yang tidak muncul dalam data latih. Jika kata tidak muncul selama pelatihan model, kata tersebut dapat dipecah menjadi *n-gram* untuk mendapatkan *embedding* vektornya.

2. Pembahasan

Dataset

Dataset yang digunakan pada penelitian ini adalah *20 newsgorup* (The UCI KDD Archive, 1999a) yang terdiri dari sekitar 18.846 artikel, 134.142 kosakata, dengan 20 topik yang dibagi menjadi 11.314 data latih dan 7.532 data uji. Selain itu juga digunakan dataset *Reuters Newswire Topic Classification* (The UCI KDD Archive, 1999b). Dataset ini terdiri dari 11.228 berita yang diperoleh dari kantor berita Reuters, 30.979 kosakata, dan berlabel 46 topik. Data latih dari dataset ini sebanyak 8.982 data dan data uji 2.246 data.

Word Embedding

Pada tahun 2003, Bengio dkk. (Bengio *et al.*, 2003) memperkenalkan istilah *word embedding*. *Word embedding* adalah sebuah fungsi parameter yang memetakan setiap kata ke dalam vektor berdimensi tinggi. Keunggulan *word embedding* tidak membutuhkan anotasi, dapat langsung diturunkan dari korpus tak teranotasi.

Word embedding dapat dibuat langsung dari dataset yang dimiliki atau menggunakan *pre-trained word embedding* yang telah tersedia. *Pre-trained word embedding* ini adalah *word embedding* yang telah dilatih menggunakan dataset yang besar pada domain permasalahan tertentu yang dapat digunakan untuk menyelesaikan permasalahan lain yang serupa. Penggunaan *word embedding* ini harus disesuaikan dengan domain dari kasus yang dimiliki. Misalkan permasalahan pada domain biomedik tidak akan cocok menggunakan *pre-trained word embedding* dari korpus berita atau Wikipedia.

Pada penelitian ini, sesuai dengan domain dataset, *word embedding* yang digunakan adalah *pre-trained word2vec* (*Google Code Archive*, 2013) dari korpus *Google news* dengan 3 juta kosakata, *pre-trained* GloVe (GloVe: Global Vectors for Word Representation, 2014) dari korpus Wikipedia+Gigaword 5 dengan 400.000 kosakata, dan *pre-trained* FastText (English word vectors · fastText, 2020) dari satu juta vektor kata yang dilatih pada korpus berita Wikipedia, UMBC, dan berita *statmt.org*. Ketiga *word embedding* memiliki 300 dimensi vektor kata. *Word embedding* ini kemudian dibandingkan kinerjanya pada klasifikasi berita dari dataset *20 newsgroup* dan *Reuters Newswire*.

Convolutional Neural Network

Dalam bidang *Natural Language Processing*, *Deep Learning* telah terbukti handal dalam sejumlah permasalahan klasifikasi. Salah satunya, *Convolutional Neural Network* (CNN) yang mampu secara efisien menangkap representasi bermakna dari kalimat seperti pada klasifikasi dan pemodelan bahasa (Kalchbrenner, Grefenstette and Blunsom, 2014), analisis sentimen (Kim, 2014) hingga ekstraksi informasi (Chen *et al.*, 2015; Nguyen and Grishman, 2015; Nurdin and Maulidevi, 2018).

Pada penelitian ini, CNN satu dimensi digunakan untuk pemodelan klasifikasi artikel berita ke dalam sejumlah topik pada dataset. CNN satu dimensi ini sangat efektif dalam menurunkan fitur dari segmen dengan panjang yang tetap dari keseluruhan dataset dan bekerja dengan baik untuk permasalahan *Natural Language Processing* (NLP). Tidak ada penambahan *handcrafted features* yaitu fitur yang diperoleh dengan melibatkan

pengetahuan pakar linguistik. Semua fitur dipelajari oleh algoritma langsung dari dataset.

Gambar 4. menunjukkan arsitektur Convolutional Neural Network yang digunakan, terdiri dari layer *input, convolutional, max pooling, dan fully connected*.

1. Input Layer

Teks dari setiap artikel berita terlebih dahulu ditransformasikan ke dalam representasi vektor kata menggunakan *word embedding* kemudian dimasukkan ke input layer. Maksimum panjang sekuens dari input adalah 1000, sehingga input akan berupa matriks berukuran 1000 x 300.

2. Convolutional Layer

Pada layer ini, akan terdapat 128 *filter* dengan *kernel size* 5 akan bergerak vertikal menyusuri keseluruhan matriks input. Setelah dilakukan operasi perkalian dan penjumlahan antara bobot *filter* dan bobot dari matriks input, serta operasi *non-linear* menggunakan fungsi aktivasi ReLU, maka diperoleh *feature map* yang berisi fitur-fitur penting yang berdimensi lebih rendah di *hidden layer* pertama. Kemudian *feature map* dari *hidden layer* pertama akan menjadi input pada *convolutional layer* kedua, dan seterusnya. Pada model CNN yang dibentuk menggunakan 3 *convolutional layer*.

3. Max Pooling Layer

Max Pooling Layer akan mengambil nilai tertinggi dari elemen-elemen yang berada pada lingkup *window* satu dimensi berukuran 5, sehingga diperoleh informasi paling penting dari *feature map* hasil konvolusi.

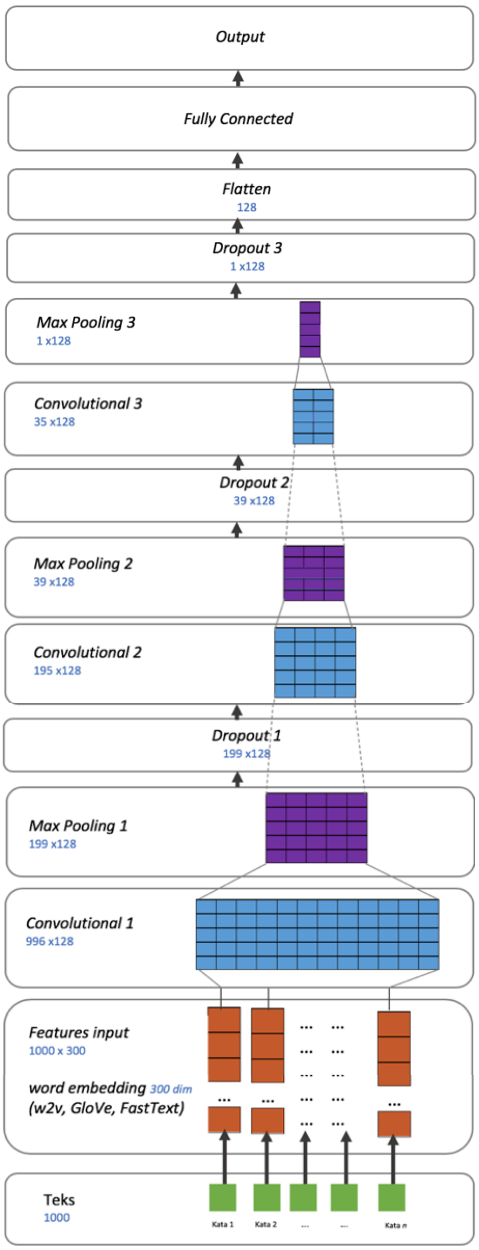
4. Fully Connected Layer

Output dari *hidden layer* sebelumnya, berupa *feature map* yang telah di-*reshape* menjadi sebuah vektor, dihubungkan dengan *output layer* untuk diklasifikasikan. Pada layer ini, digunakan *f*ungsi aktivasi *softmax* dan *loss function categorical_crossentropy*, karena variabel *output multiclass* direpresentasikan ke dalam *one-hot encoding* yang terdiri dari angka 0 dan 1.

CNN dilatih dengan *batch size*=128 dan *epoch*=20. Serta menggunakan fungsi optimasi Adam. Untuk menghindari *overfitting*, pada setiap *hidden layer* diaplikasikan teknik regulasi *dropout* (Srivastava *et al.*, 2014) dengan rate 0.5 yang akan secara acak me-non-aktifkan 50% *neuron* selama fase pelatihan.

Perbandingan Kinerja Word2Vec, GloVe, dan FastText

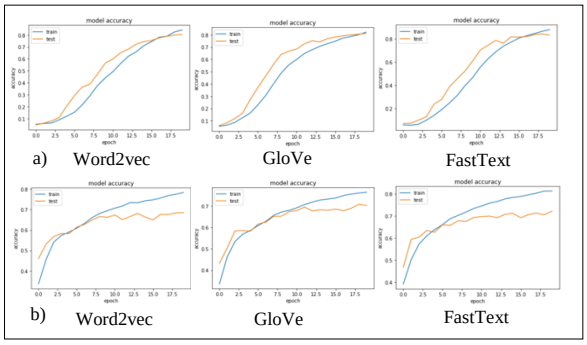
Evaluasi kinerja model CNN yang telah dilatih menggunakan *precision* (P), *recall* (R), dan *F-Measure* (F). Kinerja model CNN dievaluasi dengan menggunakan dua dataset yang berbeda dataset 1 *20 newsgroup* dan dataset 2 *Reuters* masing-masing dengan *word embedding Word2vec*, *GloVe*, dan *FastText*. Tabel 1. dan Gambar 5. menunjukkan hasil eksperimen yang telah dilakukan.



Gambar 4 *Arsitektur CNN untuk Klasifikasi Teks*

Tabel 1. *Kinerja Word Embedding untuk Setiap Dataset*

No	Dataset	Word Embedding	P	R	F
1	20 newsgroups	Word2Vec	0.942	0.933	0.925
		GloVe	0.96	0.958	0.958
		FastText	**0.98**	**0.979**	**0.979**
2	Routers	Word2Vec	0.698	0.704	0.694
		GloVe	0.687	0.711	0.688
		FastText	**0.709**	**0.733**	**0.715**



Gambar 5 *Grafik akurasi training dan testing data pada dataset 20 newsgroup (a) dan Routers (b)*

Berdasarkan hasil evaluasi kinerja ketiga *word embedding*, F-Measure kinerja GloVe pada dataset-1 lebih baik dibandingkan dengan kinerja word2vec, sedangkan pada dataset-2 word2vec lebih baik dari GloVe. Di antara ketiga word embedding ini, kinerja fastText paling unggul untuk kedua dataset tersebut.

Word2vec dan GloVe adalah *pre-trained embedding* yang mendefinisikan sebuah fungsi yang memetakan sebuah kosakata *v* ke dalam vektor *h* (Finding Syntax with Structural Probes · John Hewitt, 2019).

$$f_{vocab} : v \rightarrow h \quad (4)$$

Sedangkan FastText memetakan suku kata dari sebuah kosakata dan urutan karakter ($c_1 \ldots c_n$) ke dalam vektor *h*. Urutan karakter dari bahasa seringkali mengindikasikan komposisi informasi dari makna sebuah kata (Finding Syntax with Structural Probes · John Hewitt, 2019).

$$f_{subword} : (v(c_1, \ldots, c_n)) \rightarrow h \quad (5)$$

FastText memiliki kemampuan memberikan representasi kata yang tidak muncul dalam data latih atau mampu mengatasi permasalahan *out of vocabulary*. Kata yang tidak ditemukan selama proses *training*, kata tersebut dipecah menjadi *n-gram* berupa kumpulan urutan suku kata untuk mendapatkan *embedding* vektornya. Keunggulan ini terbukti dari hasil eksperimen yang menunjukkan kinerja FastText yang lebih baik dari word2vec dan GloVe untuk kedua dataset.

Hasil eksperimen ini sejalan dengan beberapa penelitian menunjukkan *embedding* GloVe bekerja lebih baik pada beberapa dataset, sedangkan penelitian lainnya pada dataset yang berbeda menunjukkan *embedding* word2vec lebih baik. Secara umum kedua *word embedding* ini sangat baik dalam menangkap semantik analogi.

Li H. dkk (Li *et al.*, 2018) melakukan penelitian yang membandingkan tiga *word embedding* (*Word2Vec*, *GloVe* dan *FastText*) pada kasus klasifikasi *crisis tweet*. Word embedding yang digunakan adalah *pre-trained* dari korpus Google News, Wikipedia atau Twitter, dan *word embedding* yang dibangun dari *domain crisis-specific* dari korpus *tweet*. Diperoleh hasil bahwa *crisis-specific embedding* lebih cocok untuk klasifikasi *tweet* terkait krisis yang lebih spesifik, sedangkan *pre-trained embedding* lebih cocok untuk klasifikasi yang lebih general. Dari tiga tipe *word embedding* (word2vec, GloVe and FastText), GloVe memiliki kinerja terbaik untuk ketiga dataset yang digunakan (CrisisLexT6, CrisisLexT26 and 2CTweets).

Kang HJ.,dkk (Kang *et al.*, 2016) menemukan bahwa word2vec memberikan kinerja terbaik pada perbandingan *word embedding* Word2vec CBOW, GloVe, dan Collobert & Weston untuk *English and Cross-Lingual Chinese Word Sense Disambiguation*.

Pada investigasi segmentasi topik pada bahasa Arab dan Inggris yang dilakukan Naili M. dkk (Naili, Chaibi and Ben Ghezala, 2017) diperoleh hasil bahwa word2vec dan GloVe lebih efektif daripada LSA untuk kedua bahasa. Dibandingkan dengan GloVe, word2vec menghasilkan representasi vektor kata terbaik dengan ruang semantik dimensi kecil. Sedangkan kualitas dari segmentasi topik bergantung pada bahasa yang digunakan. Kualitas segmentasi pada bahasa Arab lebih buruk daripada Bahasa Inggris karena kompleksitas bahasanya yang tinggi.

Berdasarkan hasil eksperimen dan beberapa penelitian yang membandingkan kinerja word2vec, GloVe, dan FastText, terbukti ketiga *word embedding* ini memiliki kinerja yang kompetitif. Kinerja *word embedding* bergantung pada dataset yang digunakan dan domain permasalahan yang diselesaikan.

3. Kesimpulan

Ada beberapa metode yang dapat digunakan untuk merepresentasikan teks ke dalam bentuk vektor, salah satunya dengan menggunakan *word embedding*. Word2vec adalah *word embedding* berupa matriks bobot yang diperoleh dari hasil pelatihan *unsupervised learning neural network*. Wod2vec mengandalkan informasi lokal dari bahasa. Semantik yang dipelajari dari kata tertentu dipengaruhi oleh kata-kata sekitarnya. Berbeda dengan word2vec, GloVe menunjukkan bagaimana cara melibatkan informasi statistik global yang terkandung dalam dokumen. Makna semantik dari kata tidak hanya dipengaruhi oleh kata-kata di sekitarnya tetapi juga informasi statistik global dari dokumen. GloVe menggunakan rasio dari co-occurance probability antarkata. Sedangkan FastText dibangun dari model word2vec yang memetakan *subword*/ suku kata.

Berdasarkan hasil eksperimen, kinerja CNN dalam mengklasifikasikan teks menggunakan *word embedding* word2vec, GloVe, dan FastText menggunakan ukuran F-Measure secara berturut-turut untuk dataset *20 newsgroup* adalah 0.925, 0.958, dan 0.979, dan dataset *Reuters News* adalah 0.694, 0.688, dan 0.715. Word2vec dan GloVe tidak mampu merepresentasikan vektor dari kata yang tidak ada dalam korpus (*out of vocabulary*). Berbeda dengan FastText yang dapat diandalkan untuk permasalahan *out of vocabulary* ini. Kinerja terbaik dari eksperimen diperoleh dengan menggunakan *word embedding* FastText. Namun, perbedaan kinerja yang tidak begitu signifikan ini menunjukkan bahwa ketiga *word embedding* ini memiliki kinerja yang kompetitif.

Penggunaannya sangat bergantung pada dataset yang digunakan dan permasalahan yang ingin diselesaikan.

Daftar Pustaka

Bengio, Y. *et al.* (2003) 'A Neural Probabilistic Language Model', *Journal of Machine Learning Research 3*, p. 19.

Bojanowski, P. *et al.* (2017) 'Enriching Word Vectors with Subword Information', *Transactions of the Association for Computational Linguistics*, 5, pp. 135–146. doi: 10.1162/tacl_a_00051.

Chen, Y. *et al.* (2015) 'Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks', in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. *ACL-IJCNLP 2015*, Beijing, China: Association for Computational Linguistics, pp. 167–176. doi: 10.3115/v1/P15-1017.

English word vectors · fastText (2020). Available at: https://fasttext.cc/index.html (Accessed: 21 June 2020).

Finding Syntax with Structural Probes · John Hewitt (no date). Available at: https://nlp.stanford.edu//~johnhew//structural-probe.html?utm_source=quora&utm_medium=referral#the-structural-probe (Accessed: 21 June 2020).

GloVe: Global Vectors for Word Representation (2014). Available at: https://nlp.stanford.edu/projects/glove/ (Accessed: 21 June 2020).

Google Code Archive - Long-term storage for Google Code Project Hosting. (2013). Available at: https://code.google.com/archive/p/word2vec/ (Accessed: 21 June 2020).

Kalchbrenner, N., Grefenstette, E. and Blunsom, P. (2014) 'A Convolutional Neural Network for Modelling Sentences', in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. *ACL 2014*, Baltimore, Maryland: Association for Computational Linguistics, pp. 655–665. doi: 10.3115/v1/P14-1062.

Kang, H. J. *et al.* (2016) 'A Comparison of Word Embeddings for English and Cross-Lingual Chinese Word Sense Disambiguation', in *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 30–39. Available at: https://www.aclweb.org/anthology/W16-4905 (Accessed: 21 June 2020).

Kim, Y. (2014) 'Convolutional Neural Networks for Sentence Classification', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. *EMNLP 2014*, Doha, Qatar: Association for Computational Linguistics, pp. 1746–1751. doi: 10.3115/v1/D14-1181.

Li, H. *et al.* (2018) 'Comparison of Word Embeddings and Sentence Encodings as Generalized Representations for Crisis Tweet Classification Tasks', *New Zealand*, p. 13.

Mikolov, T. *et al.* (2013) 'Efficient Estimation of Word Representations in Vector Space', *arXiv:1301.3781 [cs]*. Available at: http://arxiv.org/abs/1301.3781 (Accessed: 21 June 2020).

Naili, M., Chaibi, A. H. and Ben Ghezala, H. H. (2017) 'Comparative study of word embedding methods in topic segmentation', *Procedia Computer Science*, 112, pp. 340–349. doi: 10.1016/j.procs.2017.08.009.

Nguyen, T. H. and Grishman, R. (2015) 'Event Detection and Domain Adaptation with Convolutional Neural Networks', in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. *ACL-IJCNLP 2015*, Beijing, China: Association for Computational Linguistics, pp. 365–371. doi: 10.3115/v1/P15-2060.

Nurdin, A. and Maulidevi, N. U. (2018) '5W1H Information Extraction with CNN-Bidirectional LSTM', *Journal of Physics: Conference Series*, 978, p. 012078. doi: 10.1088/1742-6596/978/1/012078.

Pennington, J., Socher, R. and Manning, C. (2014) 'Glove: Global Vectors for Word Representation', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, pp. 1532–1543. doi: 10.3115/v1/D14-1162.

Rong, X. (2016) 'word2vec Parameter Learning Explained', *arXiv:1411.2738 [cs]*. Available at: http://arxiv.org/abs/1411.2738 (Accessed: 21 June 2020).

Selivanov, D. (2020) *GloVe Word Embeddings*. Available at: https://cran.r-project.org/web/packages/text2vec/vignettes/glove.html (Accessed: 21 June 2020).

Srivastava, N. *et al.* (2014) 'Dropout: a simple way to prevent neural networks from overfitting', *The Journal of Machine Learning Research*, 15(1), pp. 1929–1958.

The UCI KDD Archive, Information and Computer Science and University of California, Irvine (1999a) *20 Newsgroups*. Available at: https://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html (Accessed: 21 June 2020).

The UCI KDD Archive, Information and Computer Science and University of California, Irvine (1999b) *Reuters-21578 Text Categorization Collection*. Available at: http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html (Accessed: 21 June 2020).