Biomedical Named Entity Recognition Using Word Embedding Model of FastText Technique

Akrm Ben Niran, Sabrina Tiun

Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia
43600 Bangi, Selangor Darul Ehsan, Malaysia

Email: akit.992@gmail.com, sabrinatiun@ukm.edu.my

ABSTRACT

Biomedical Named Entity Recognition (BNER) has been presented as a task to identify all the textual medical entities automatically. In the past, machine learning techniques were trained to detect these entities using traditional features such as dictionaries, gazetteers, word's length and syntactic tag. Recently, most of the literature has turned into using the word embedding technique where every word will be associated with a numeric embedding that distinguishes its contextual meaning. Mainly, the majority of the recent BNER studies have been relying on either Word2Vec or Glove models. Yet, these models suffer from a remarkable limitation which is the out-of-vocabulary. Such problem occurs because these models are mainly working with words as units without considering their character representation. This would make the embedding model to consider only the words that have been encountered in the training and any unseen word would not have an embedding later. This study aims to propose an efficient word embedding using the FastText model for the BNER task. Such a model has the ability to consider the character N-gram of each word in order to produce its embedding. This would contribute toward minimizing the out-of-vocabulary words. In particular, this study utilizes two benchmark biomedical datasets; the first for disease names, and the second for gene expressions. In addition, a Logistic Regression (LR) classifier has been utilized to accommodate the classification after getting the embedding of FastText. Results of applying the proposed embedding showed an outperformance compared to the baseline where an f-measure of 96.61% has been acquired from the first dataset, and an f-measure of 97.46% has been acquired from the second dataset. These results can demonstrate the effectiveness of using FastText as a word embedding technique.

Key words: Biomedical Named Entity Recognition, Word Embedding, Word2Vec, FastText, Logistic Regression

INTRODUCTION

The expansion of medical information has brought many demands and needs in terms of analyzing the textual entities. This can be represented by detecting genes, viruses, drugs, diseases and other medical instances. In this sense, Biomedical Named Entity Recognition (BNER) has been presented as a task to identify all the textual medical entities in an automatic manner (Chen *et al.*, 2019). In the past, machine learning techniques were trained to detect these entities using traditional features such as dictionaries, gazetteers, word's length and syntactic tag (Alshaikhdeeb and Ahmad, 2016). Recently, most of the literature has turned into using the word embedding technique where every word will be associated with a numeric embedding that distinguishes its contextual meaning.

Word embedding technique is a revolutionary method where the text feature space is continuously representing the contexts of words. In this regard, every word is represented in the multiple dimension space where several features can be detected such as its syntactic tag and its contextual meaning (Goldberg and Levy, 2014). Word embedding has been proposed for wide range of textual applications including BNER. Yet, there are still numerous issues that are facing this technique.

State of the art in BNER task studies was based on word embedding in which the biomedical text corpus is being processed using an embedding architecture such as LSTM or CNN to generate the embedding (Li and Jiang, 2017). However, all these studies that have utilized the word embedding were using either word2vec (Zhu *et al.*, 2017) or Glove (Cho and Lee, 2019) paradigms. These paradigms are mainly working with words as units without considering their character representation. This would make the embedding model to consider only the words that have been encountered in the training. Thus, if an unseen word has been encountered, it would not have embedded in the model (Pylieva *et al.*, 2018).

For example, assume the word 'Cancer' is located in a corpus and the embedding model has been trained to generate embedding for such word. After the training and saving the model, if the word 'Cancers' has been encountered, it would have no vector in the model because the model would consider 'Cancer' and 'Cancers' as two different words. This is known as the out-of-vocabulary problem.

The out-of-vocabulary problem is considered to be the main hindering factor toward achieving the high accuracy of biomedical entity detection. Therefore, it is crucial to overcome this problem to improve accuracy. This study aims to propose an efficient word embedding using FastText model for the BNER task. Such model has the ability to consider the character N-gram of each word in order to produce its embedding. This would contribute toward minimizing the out-of-vocabulary words. In particular, this study utilizes two benchmark biomedical datasets; the first for disease names, and the second for gene expressions. In addition, an LR classifier has been utilized to accommodate the classification after getting the embedding of FastText. Results of applying the proposed embedding showed an outperformance compared to the baseline where an F-measure of 96.61% has been acquired from the first dataset, and an F-measure of 97.46% has been acquired from the second dataset. These results demonstrate the effectiveness of using FastText as a word embedding technique for BNER.

RELATED WORK

In the literature, there are plenty of studies that have addressed BNER task. For instance, Bhasuran et al. (2016) presented a BNER method using a CRF classifier. The proposed method utilizes some features such as affixes, dictionary-based and syntactic features. Two benchmark datasets were used including NCBI which contains disease names and BioCreative-II which contains genes. Results of f-measure were 89.12% for NCBI and 76.74% for BioCreative-II.

Gridach (2017) presented a word embedding method for BNER task using a technique known as Long-Short Term Memory (LSTM). This technique was intended to generate word embedding for terms from BioCreative-II dataset. After that, based on the embedding, LSTM will also classify the biomedical terms. Results of f-measure was 89.46%.

In the same regard, Li and Jiang (2017) presented a BNER method based on LSTM for word embedding and biomedical entities detection. Using BioCreative-II dataset, result of f-measure was 89.49%.

Zhu et al. (2017) presented a word embedding technique based on Convolutional Neural Network to detection biomedical entities. Using BioCreative-II dataset and NCBI dataset, the proposed method obtained 87.2% of f-measure for the first dataset, and 87.26% of f-measure for the second dataset.

Cho and Lee (2019) presented a combination method of CRF and LSTM for BNER task. In fact, LSTM has been used to generate word embedding and CRF has been used to detect hidden features of the word such as its mapping from dictionary. Using BioCreative-II and NCBI datasets, the proposed combination method showed an f-measure of 81.44% for the first dataset, and 85.68% for the second dataset.

MATERIALS AND METHODS

The research framework has been designed to articulate the research objectives where the proposed efficient word embedding of FastText is being applied. There are five stages that have to be accomplished in order to fulfill the objectives. The first stage contains the datasets that are intended to be used within the experiments. For this purpose, two different biomedical datasets are being used. The second stage contains the preprocessing tasks that are intended to convert the datasets into an appropriate form for processing. The third stage contains the word embedding where the preprocessed datasets would undergo an analytical task to get embedding for each distinctive word. For this purpose, two word embedding architectures are used including the standard Word2Vec which considered to be the baseline, and the FastText which considered to be the proposed embedding that will be compared against the standard embedding. The fourth stage contains the classification where the biomedical entities (BNEs) are being categorized into its actual class label. For this purpose, a Logistic Regression (LR) classifier is used. Finally, the fifth stage contains the evaluation where the results of classification are being assessed. Figure 1 depicts the aforementioned stages.

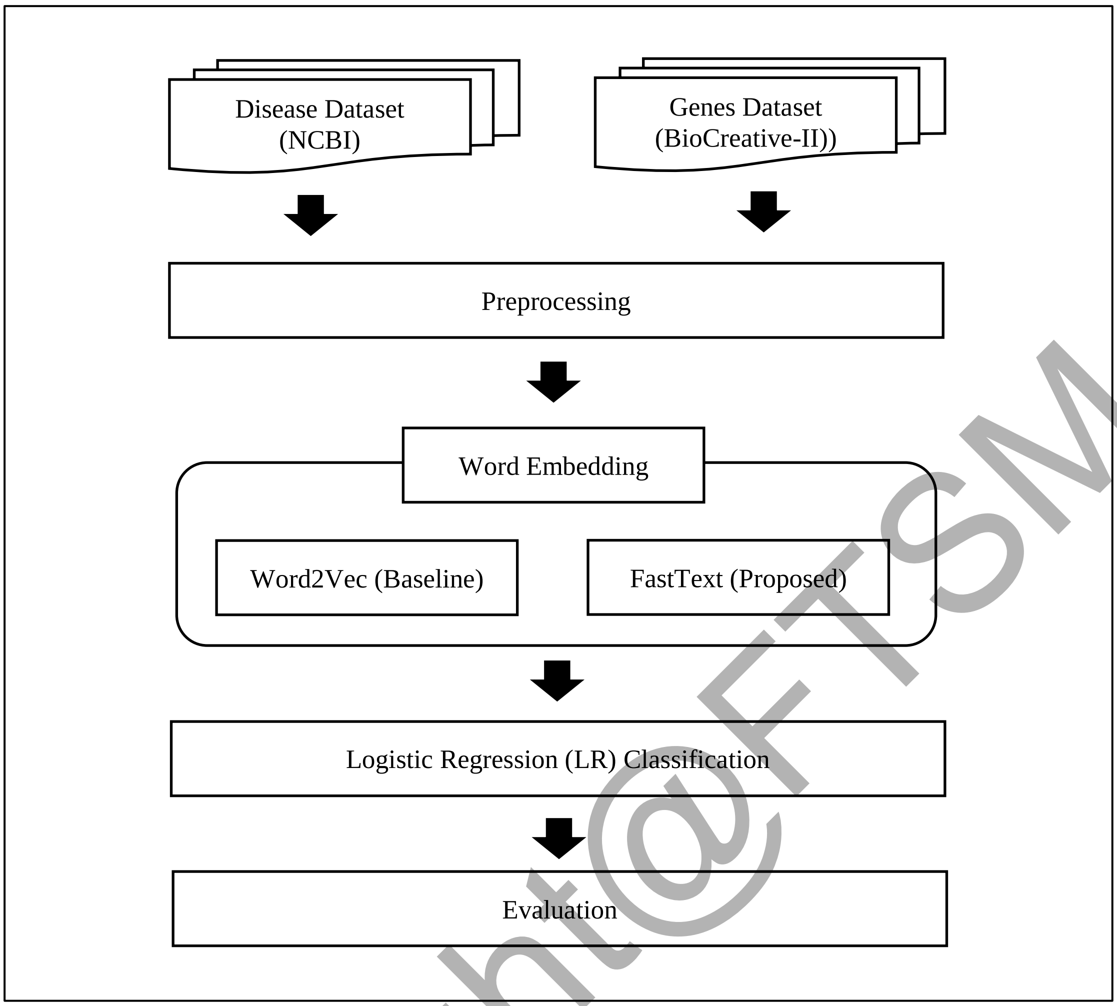


Figure 1. Phases of research framework

Dataset

In this section, the datasets that are intended to be examined within the experiments will be discussed. In this regard, two biomedical datasets have been considered. The reason behind selecting two different biomedical datasets is to provide a demonstrate the ability of word embedding in terms of representing various types of BNEs (e.g. disease names, gene names, etc.). In fact, both datasets will be used to train both Word2Vec and FastText models. Following subsections will discuss each dataset individually.

Table 1 NCBI statistics (Doğan, *et al.*, 2014)

Class	**Explanation**	**Quantity**
UN	Disease name	8475 (6892 diseases, and 790 unique)
O	Non-disease name	120,569

The second dataset has been introduced by Smith et al. (2008). It focuses on genes and gene-related mentions where approximately 20 thousand of genes along with 13 thousand of gene-related are being included. The dataset is accessed at http://biocreative.sourceforge.net/biocreative_2_dataset.html. Table 2 depicts the statistics of such dataset.

Table 2 BioCreative-II statistics (Smith, *et al.*, 2008)

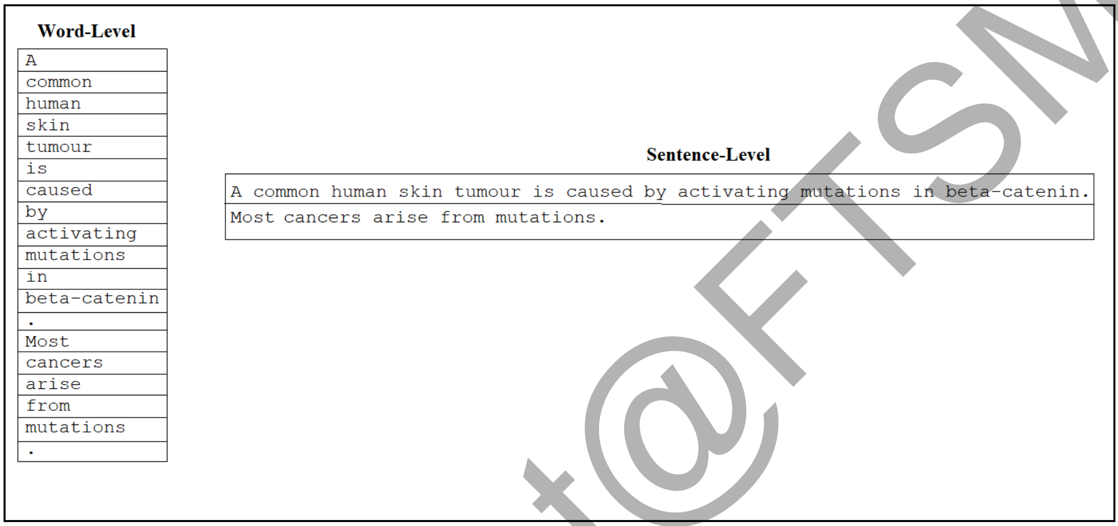
Class	**Explanation**	**Quantity**
UN	Genes	15,700
O	Non-gene	371,000

Preprocessing

In this section, the preprocessing tasks that have been utilized will be discussed. The aim of preprocessing is to convert the data into a suitable format that is able to be processed either via Word2vec or FastTest embedding models. In this regard, there are two different preprocessing tasks that have been applied each of which has been dedicated to one of the two embedding models. Following subsections will describe the two tasks.

Preprocessing for Word2Vec

Since Word2Vec is intended to configure wide range of text contexts, and since the two datasets that have been used in this study are containing word for each row thus, it is important to convert both datasets from word-level into sentence-level. For this purpose, a streaming of word read will be applied until encountering a period '.' which would refer to the end of sentence. Figure 2 shows an example of this task.



Word-Level

```
A
common
human
skin
tumour
is
caused
by
activating
mutations
in
beta-catenin
.
Most
cancers
arise
from
mutations
.
```

Sentence-Level

```
A common human skin tumour is caused by activating mutations in beta-catenin.
Most cancers arise from mutations.
```

Figure 2. Convert words of dataset into sentences

As shown in Figure 2, every word has been joined to the proceeding one until encountering a period of '.' where such period would end the merging of words. Note that, this sentence conversion will occur for generating word embedding through Word2Vec.

Preprocessing for FastText

Unlike the Word2Vec, FastText has the ability to generate word embedding based on the character N-gram rather than the word context. Therefore, there is no need for word-to-sentence conversion. However, to make FastText generate accurate embedding for the words, it is important to concatenate each class label with the word "__label__". This word is added to tell the FastText model that such word is a class label. Table 3 shows an example of performing such task.

Table 3 Example of transforming classes

Word	**Original Class**	**Transformed Class**
A	O	__label__O
common	O	__label__O
human	O	__label__O
skin	I-UN	__label__I-UN
tumour	I-UN	__label__I-UN
is	O	__label__O
caused	O	__label__O
by	O	__label__O
activating	O	__label__O
mutations	O	__label__O
in	O	__label__O
beta-catenin	O	__label__O
.	O	__label__O

Word Embedding Using Word2Vec

Word embedding technique is intended to analyze a large text set in order to give an embedding for each distinctive word (Levy and Goldberg, 2014). The most common architecture of word embedding is the Word2Vec (Goldberg and Levy, 2014). Such architecture is a feed-forward neural network that inputs the one-hot vector of a word and outputs the one-hot vector of the proceeding word. Once the neural network finishes the training and learning to output the target vector, the hidden neurons' values will correspond to the desired embedding. To understand the exact workflow of Word2Vec, let consider a converted sentence resulted from preprocessing in Figure 3.2.

Sentence S = "most cancers arise from mutations"

Prior to process such sentence through the Word2Vec, it is important to transform it into a one-hot encoding. Such encoding aims at generating initial vectors for each word to be processed through the neural network architecture (Seger, 2018). To do so, a table of word-to-word matrix should be initiated. Such table will be populated with ones and zeros where the value one refers to a correspondence among the term and the value zero refers to the non-correspondence. Correspondence refers to the matching of the same word. Note that, insignificant terms such as stop-words (e.g. the words 'most' and 'from' in sentence S) will be discarded. Table 4 depicts the one-hot encoding of S.

Table 4 One-hot encoding for sentence *S*

	cancers	arise	mutations
cancers	1	0	0
arise	0	1	0
mutations	0	0	1

As shown in Table 4, the correspondence between the word 'cancers' from the first column and the word 'cancers' from the first row has been populated with a value of '1'. Otherwise, the remain cells have been populated with a value of '0'.

After generating the one-hot encoding matrix, there are two ways to process the vectors of words. The first way is known as Continuous Bag of Words (CBOW) which aims to input multiple context words and output the target or center word between them. Let consider the vectors of the context words as follow:

Vector of context word 1 '*cancers*'	1	0	0
Vector of context word 2 '*mutations*'	0	0	1

On the other hand, the target word would be listed as follow:

Vector of target word '*arise*'	0	1	0

Note that, the target word is chosen through a loop where every word in a sentence will be selected as target in each iteration

Now, both context words will be processed as input and the target word as output through the CBOW architecture as shown in Figure 3.

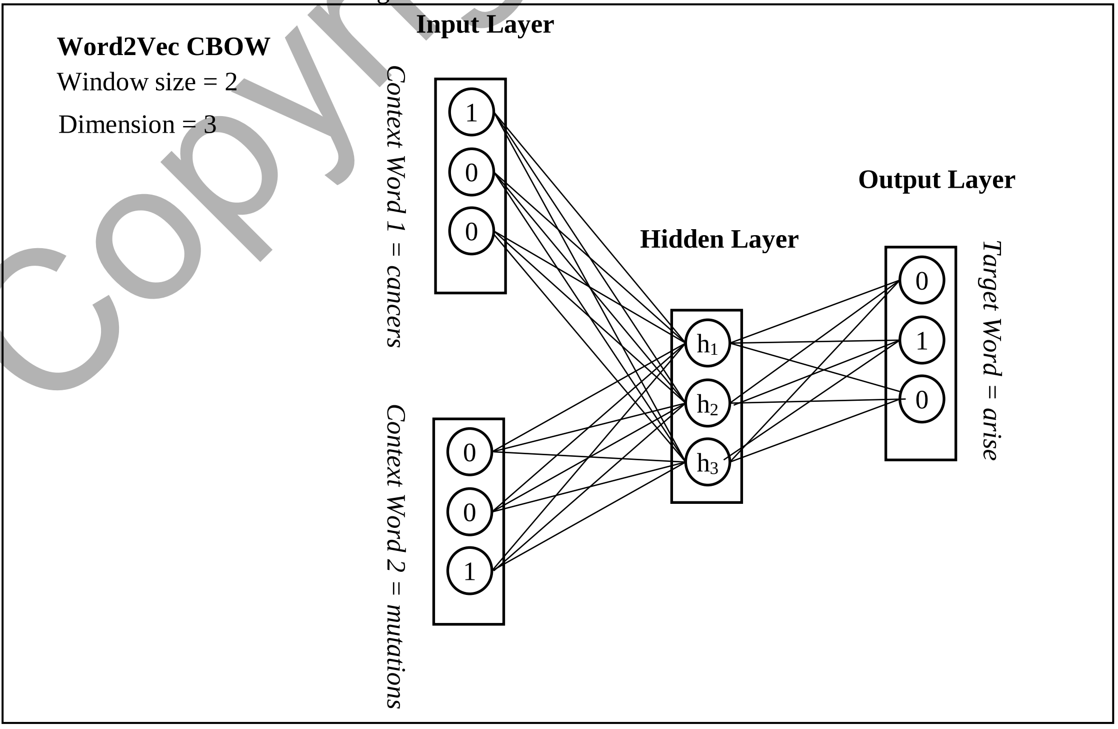


Figure 3 Word2Vec (CBOW) architecture

As shown in Figure 3.3, both context words have been processed as input and the target word has been set as output. Note that, a hidden layer that matches the size of each word (i.e. 3) has been considered. The size of each word corresponds to the dimension where three neurons have been articulated for each word along with the hidden layer. On the other hand, the window size refers to the number of words that are being considered in the input layer. In our case, the number was 2 which corresponds to the two context words.

Consequentially, as in any neural network architecture, input-to-hidden weights Wh = {Wh1, Wh2, …, Whn} are being initiated with random values. Then, the hidden neurons values hi can be obtained by calculating the following equation:

$$\boldsymbol{h_i = \sum W_{hi} \times Input_i} \quad \textbf{(1)}$$

Where Input refers to the value of neurons in input layers. After acquiring the hidden values, the hidden-to-output weights W_o = {W_{o1}, W_{o2}, …, W_{on}} will be initiated with random values. Then, the output neuron values O_i can be obtained by calculating the following equation:

$$\boldsymbol{O_i = \sum W_{oi} \times h_i} \quad \textbf{(2)}$$

After acquiring the resulted output values, a comparison will be made with the actual output values, if there is a relative variance, a training process will take a place where the random weights in both input-to-hidden and hidden-to-output will be re-adjusted. This process will continue until the resulted output values match the actual output values. Once, the matching is obtained, the values of hidden neurons will be considered as embedding vector of the target word.

On the other hand, the second way of processing the one-hot vectors of sentence S is by using Skip-gram architecture. Such architecture is the opposite of CBOW where the aim is to input a target word and attempt to output the context words as in Figure 4.

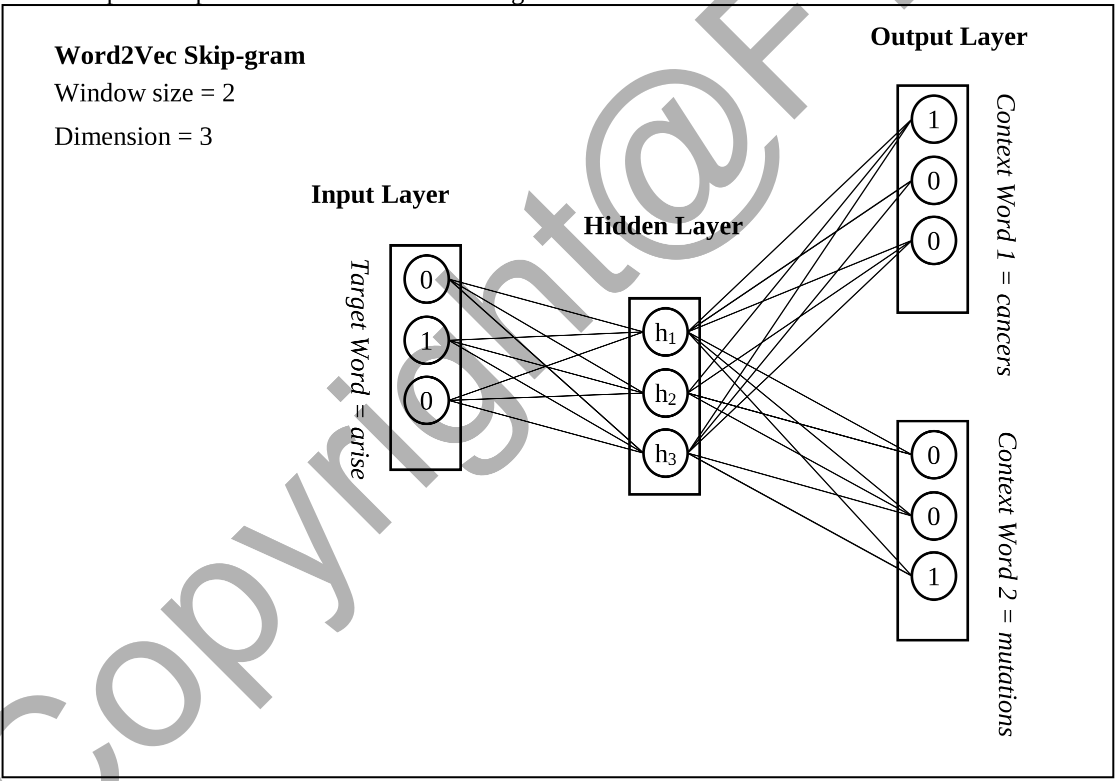


Figure 4 Word2Vec (Skip-gram) architecture

Similar to the CBOW, Skip-gram contains a window size of 2 which refers to the number of context words that are intended to be predicted. As well as, the dimension is 3 which corresponds to the number of neurons of each word's vector.

On the other hand, the same procedures of training and weight adjusting that discussed in CBOW, will be repeated in Skip-gram until the resulted output values match the actual output values. At that moment, the hidden neurons' values will be considered as the embedding for the target word.

Table 5 depicts an arbitrary example of resulted embedding for each word whether by CBOW or Skip-gram.

Table 5 Arbitrary example of resulted embedding of Word2Vec

	Embedding (Dimension = 3)		
cancers	-0.6345	0.9371	0.2649
arise	-0.6754	0.9366	0.2679
mutations	-0.6492	0.9299	0.2813

Word Embedding Using FastText

With the emergence of Word2Vec architecture, numerous applications have been utilizing such architecture for different tasks such as NER or other text classification problems. Yet, a remarkable problem has been arisen with the use of Word2Vec. Such problem is known as 'out-of-vocabulary' which refers to the cases where the model of Word2Vec would not have an embedding vector for specific word. This absence of embedding simply means that the Word2Vec never seen such word within the training.

This problem has been resolved by some normalization mechanisms where the unseen word can be replaced with a vector of zeros. However, sometimes such normalized vector would loss the meaning of significant words. For example, the word 'cancers' that has been trained in the previous section has an embedding shown in Table 3.7, but if such words have different derivational inflection such as the word 'cancer' that did not occur in the training and occurred in the testing. In this case, the normalized vector of zeros that would be given to such unseen word, will mislead the machine learning classification model. This is because the machine learning model would see such word equivalent to other insignificant and unseen terms.

For this purpose, the FastText architecture has been introduce by Facebook to solve the aforementioned problem. FastText aims at processing the possible character N-gram of each term and take the average vector of them as a final embedding for the term (Pylieva, *et al.*, 2018). To understand such mechanism, let consider the word 'cancers' as an example. Similar to Word2Vec, FastText will generate a one-hot encoding matrix. However, instead of considering the surrounding words of 'cancers', FastText will examine the possible character N-gram of the word itself as shown in Table 6.

Table 6 One-hot encoding for possible N-gram of word 'Cancers'

	ca	an	nc	ce	er	rs
ca	1	0	0	0	0	0
an	0	1	0	0	0	0
nc	0	0	1	0	0	0
ce	0	0	0	1	0	0
er	0	0	0	0	1	0
rs	0	0	0	0	0	1

To process the vectors of each possible N-gram, FastText contains the same two approaches of CBOW and Skip-gram. The first architecture intends to input multiple context character N-gram vectors and output a target character N-gram vector. Whereas, the second architecture intends to input a target character N-gram vector and output the context character N-gram vectors. Figure 5 and Figure 6 depicts the CBOW and Skip-gram architectures of FastText respectively.

As shown in Figure 3.5 and Figure 3.6, the dimension is 6 which corresponds to the total number of possible character N-gram of the word 'Cancers' (i.e. 'ca', 'an', 'nc', 'ce', 'er', 'rs'). Whereas, the window size is 5 which corresponds to the number of context character N-gram vectors where the sixth will be the target.

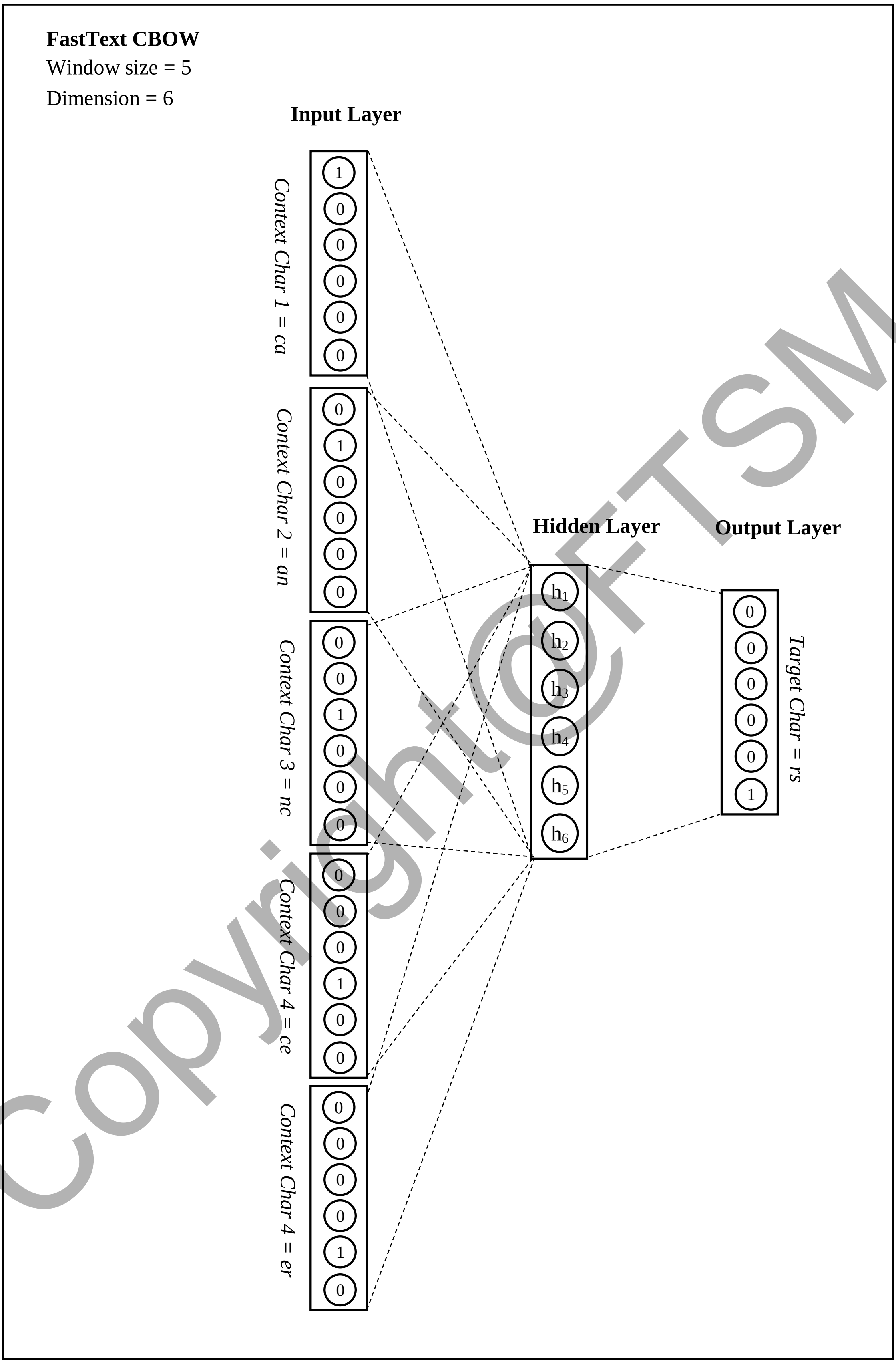


Figure 5 FastText (CBOW) architecture

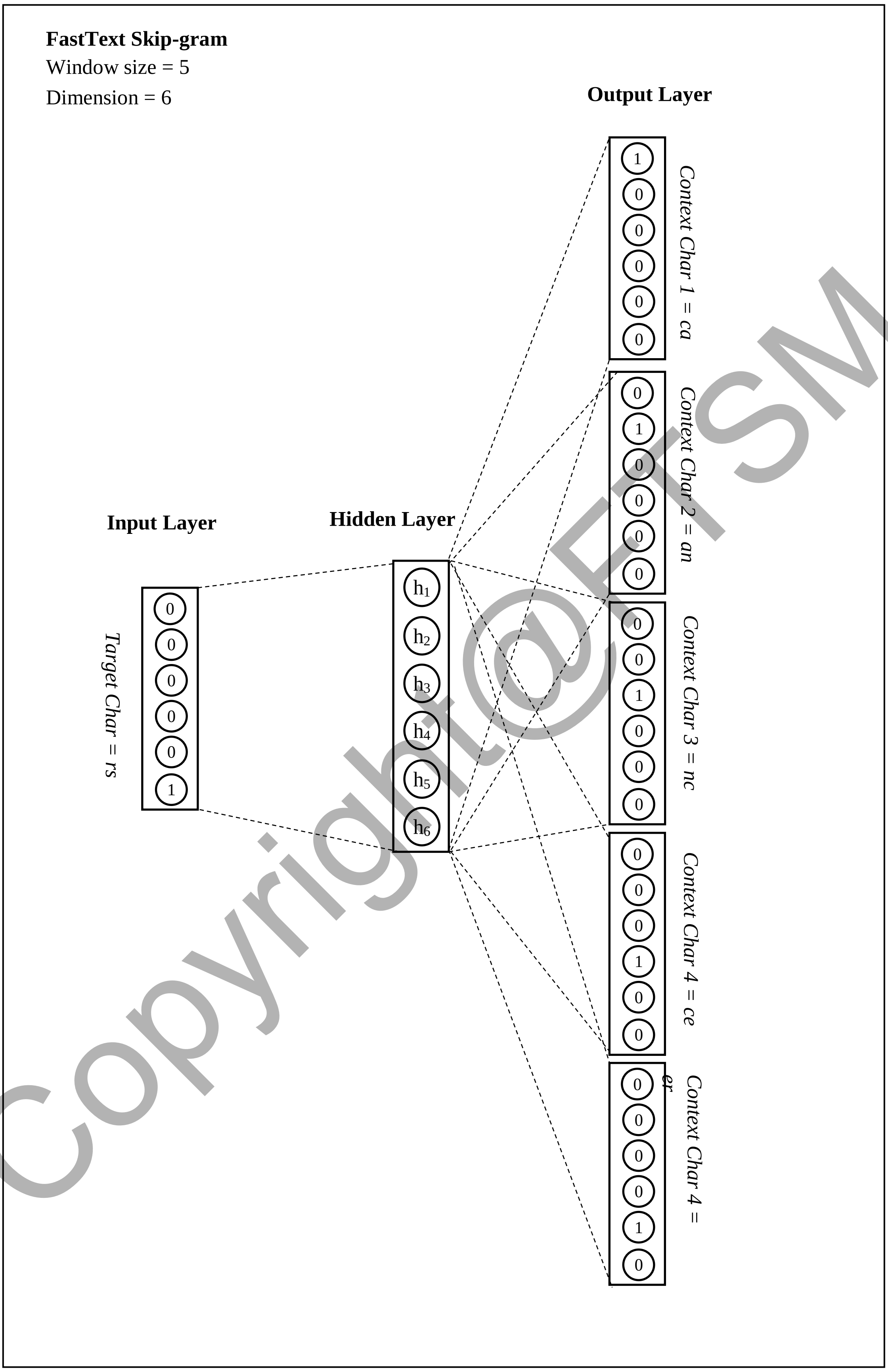


Figure 6 FastText (Skip-gram) architecture

Similar to the Word2Vec, the FastText architecture will also utilize the weight adjustment in order to match the calculated output and the actual output. Once such matching is occurred, the hidden neurons' values will consider to be the embedding vector for the target character N-gram. Let assume that the FastText (whether using CBOW or Skip-gram) finishes the training and produce embedding for each character N-gram as shown in Table 7.

Table 7 Arbitrary example of resulted embedding of FastText

	Embedding (Dimension = 6)					
ca	0.4345	0.9371	0.2649	0.9371	0.2649	0.2649
an	0.4754	0.9366	0.2679	0.9366	0.2679	0.2679
nc	0.4492	0.9299	0.2813	0.9299	0.2813	0.2813
ce	0.4345	0.9371	0.2649	0.9371	0.2649	0.2649
er	0.4754	0.9366	0.2679	0.9366	0.2679	0.2679
rs	0.4492	0.9299	0.2813	0.9299	0.2813	0.2813

Now, in order to get the embedding of the word 'Cancers', FastText will take the mean average of the six embedding vectors in Table 8. Table 9 depicts such process.

Table 8 Final word embedding by FastText

	Embedding (Dimension = 6)					
ca	0.4345	0.9371	0.2649	0.9371	0.2649	0.2649
an	0.4754	0.9366	0.2679	0.9366	0.2679	0.2679
nc	0.4492	0.9299	0.2813	0.9299	0.2813	0.2813
ce	0.4345	0.9371	0.2649	0.9371	0.2649	0.2649
er	0.4754	0.9366	0.2679	0.9366	0.2679	0.2679
rs	0.4492	0.9299	0.2813	0.9299	0.2813	0.2813
Cancers (Average)	**0.453033**	**0.934533**	**0.271367**	**0.934533**	**0.271367**	**0.271367**

In this case, FastText will give both 'Cancer' and 'Cancers' similar embedding vector and the machine learning would learn the relationship between them.

Logistic Regression (LR)

After building both Word2Vec and FastText models for both datasets, this section will take a place where an LR classifier is being used to classify the data into BNEs or not. However, prior to the classification, it is important to replace every word in the two datasets with its matching embedding from the models. To do so, the original datasets (i.e. not preprocessed) of NCBI and BioCreative-II will be brought. Consequentially, two replacement tasks will be applied; the first replacement aims at substituting each word within the two datasets by its equivalent embedding from Word2Vec model, while the second replacement aims at substituting each word within the two datasets by its equivalent embedding from FastText model. Table 9 and Table 10 depict the replacement for both NCBI and BioCreative-II datasets respectively.

Table 9 Example of embedding replacement for NCBI

Original NCBI		**NCBI replaced by Word2Vec**					**NCBI replaced by FastText**				
Word	**Class**	**Word2Vec Embedding**				**Class**	**FastText Embedding**				**Class**
		(V1	**V2**	**..**	**Vn)**		**(V1**	**V2**	**..**	**Vn)**	
A	O	0.64	0.33	..	0.51	O	0.98	0.75	..	0.11	O
common	O	0.12	0.15	..	0.46	O	0.63	0.38	..	0.41	O
human	O	0.11	0.13	..	0.39	O	0.16	0.19	..	0.54	O
skin	I-UN	0.97	0.96	..	0.63	I-UN	0.44	0.65	..	0.63	I-UN
.	.	.	.	..	.	.	.	.	..	.	.
.	.	.	.	..	.	.	.	.	..	.	.

Table 10 Example of embedding replacement for BioCreative

Original BioCreative-II		**BioCreative-II replaced by Word2Vec**					**BioCreative-II replaced by FastText**				
Word	**Class**	**Word2Vec Embedding**				**Class**	**FastText Embedding**				**Class**
		(V1	**V2**	**..**	**Vn)**		**(V1**	**V2**	**..**	**Vn)**	
SIP1	I-UN	0.96	0.76	..	0.22	O	0.63	0.32	..	0.73	O
(	O	0.66	0.26	..	0.43	O	0.55	0.23	..	0.34	O
Smad	I-UN	0.27	0.55	..	0.78	O	0.56	0.22	..	0.66	O
interacting	I-UN	0.39	0.46	..	0.52	I-UN	0.23	0.46	..	0.77	I-UN
.	.	.	.	..	.	.	.	.	..	.	.
.	.	.	.	..	.	.	.	.	..	.	.

As shown in Table 9 and Table 10, every word will be replaced with its embedding either from Word2Vec or FastText models. The embedding of brought from each model has a length of n which corresponds to the dimension of embedding.

Now, the LR classifier can train on a subset of the data and tested on the rest of the data. The reason behind using LR is that it considered as a straightforward classification where the relationship between the dependent variable (i.e. class label) and the independent variables (i.e. embedding vectors) can be accurately identified (Goldberg and Levy, 2014). The equation of learning function used by LR can be shown as follow (Khammassi and Krichen, 2017):

$$\boldsymbol{\ell = \log \frac{p}{1-p} = \beta x_1 + \beta x_2} \quad \textbf{(3)}$$

where ℓ is the log-odds of the class labels, p is the class label (i.e. BNE or not), β is a constant parameter of LR, x_1 and x_2 are the embedding vectors.

RESULT

To investigate the actual difference between using Word2Vec and FastText as a word embedding model, this section is intended to accommodate a comparison between the two models. For this purpose, only f-measure will be considered within the comparison since it is considered the harmony of precision and recall. In addition, both datasets are considered within the comparison. Table 11 and Figure 7 show the results of the comparison.

Table 11 Comparison based on F-measure

Dataset	**Word2Vec**	**FastText**
NCBI	0.73516	0.96616
BioCreative-II	0.49463	0.97464

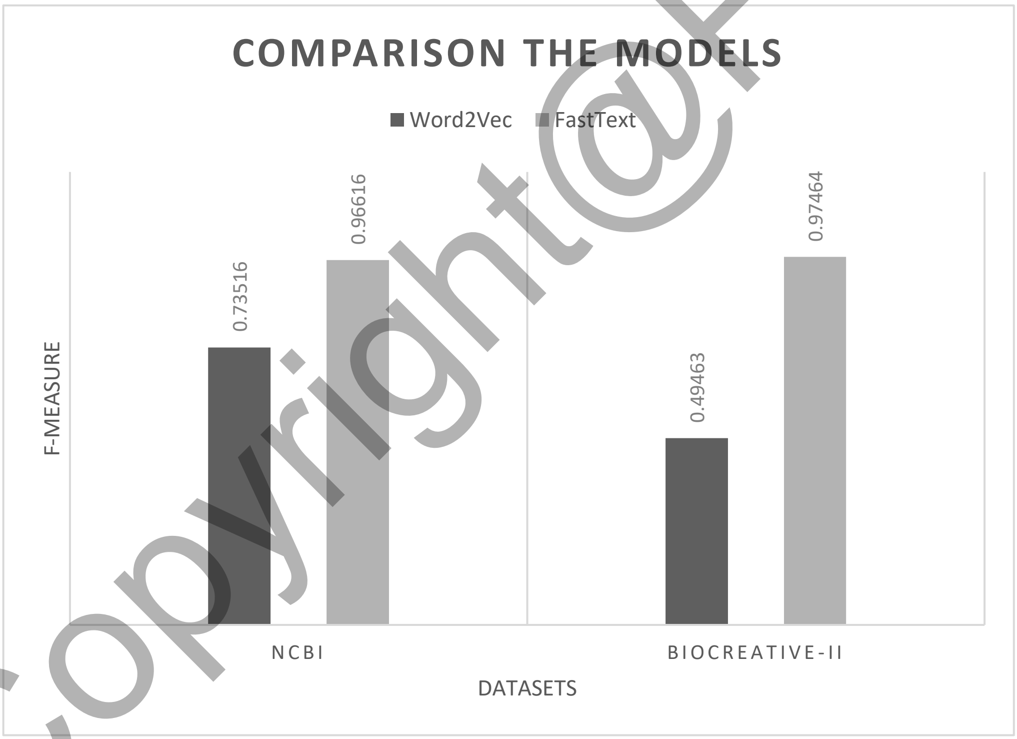


Figure 7 Performances of the two models on BNER dataset

As shown in Table 11, for NCBI dataset, the results of F-measure indicate the superiority of FastText over the Word2Vec where F-measure was 0.96 compared to 0.73 acquired by Word2Vec.

Similarly, for the BioCreative-II dataset, the results of F-measure showed the same superiority has been depicted by FastText over the Word2Vec where F -measure was 0.97 compared to 0.49 acquired by Word2Vec.

In fact, such outperformance depicted by FastText a main reason of the ultimate minimization of out-of-vocabulary instances. Since Word2Vec is concentrating on the word as context and target terms where the only the word unit is considered thus, it makes sense that there will be numerous words that do not yield any embedding. This is because simply the Word2Vec would not determine the inflectional derivations that might occurred to the terms (e.g. ‘cancer’ and ‘cancers’).

In contrast, the FastText depends on the character N-gram of the words in order to generate the embedding. Therefore, the inflectional derivations will be resolved by the model since at least one of the derivations would have an embedding within the model.

CONCLUSION

This study aims to propose an efficient word embedding using FastText model for the BNER task. Such model has the ability to consider the character N-gram of each word in order to produce its embedding. This would contribute toward minimizing the out-of-vocabulary words. In particular, this study utilizes two benchmark biomedical datasets; the first for disease names, and the second for gene expressions. In addition, an LR classifier has been utilized to accommodate the classification after getting the embedding of FastText. Results of applying the proposed embedding showed an outperformance compared to the baseline where an F-measure of 96.61% has been acquired from the first dataset, and an F-measure of 97.46% has been acquired from the second dataset. These results demonstrate the effectiveness of using FastText as a word embedding technique for BNER. For future researches, incorporation additional information to the FastText embedding vectors such as the term frequency would contribute toward improving the classification of BNEs.

REFERENCES

Alshaikhdeeb and Ahmad. 2016. Biomedical Named Entity Recognition: A Review, International Journal on Advanced Science, Engineering and Information Technology, Vol. 6 No. 6, pp. 889-895 (Access 2016

Bhasuran, et al. 2016. Stacked Ensemble Combined with Fuzzy Matching for Biomedical Named Entity Recognition of Diseases, Journal of biomedical informatics, Vol. 64 No. 2016, pp. 1-9 (Access 2016

Chen, et al. 2019. Named entity recognition from Chinese adverse drug event reports with lexical feature based BiLSTM-CRF and tri-training, Journal of biomedical informatics, Vol. 96, pp. 103252 (Access 2019

Cho and Lee. 2019. Biomedical named entity recognition using deep neural networks with contextual information, BMC bioinformatics, Vol. 20 No. 1, pp. 735 https://doi.org/10.1186/s12859-019-3321-4 (Access 2019

Doğan, et al. 2014. NCBI disease corpus: a resource for disease name recognition and concept normalization, Journal of biomedical informatics, Vol. 47, pp. 1-10 (Access 2014

Goldberg and Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method, arXiv preprint arXiv:1402.3722, (Access 2014

Gridach. 2017. Character-level neural network for biomedical named entity recognition, Journal of biomedical informatics, Vol. 70, pp. 85-91 http://www.sciencedirect.com/science/article/pii/S1532046417300977 (Access 2017

Khammassi and Krichen. 2017. A GA-LR wrapper approach for feature selection in network intrusion detection, Computers & Security, Vol. 70, pp. 255-277 http://www.sciencedirect.com/science/article/pii/S0167404817301244 (Access 2017

Levy and Goldberg. 2014. Dependency-based word embeddings, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 302-308.

Li and Jiang. 2017. Biomedical named entity recognition based on the two channels and sentence-level reading control conditioned LSTM-CRF, *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 380-385.

Pylieva, et al. 2018. Improving automatic categorization of technical vs. Laymen medical words using FastText word embeddings.

Seger. 2018. An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing.
Smith, et al. 2008. Overview of BioCreative II gene mention recognition, Genome biology, Vol. 9 No. Suppl 2, pp. S2 (Access 2008
Zhu, et al. 2017. GRAM-CNN: a deep learning approach with local context for named entity recognition in biomedical text, Bioinformatics, Vol. 34 No. 9, pp. 1547-1554 (Access 2017