The Influence of Loss Function Usage at SIAMESE Network in Measuring Text Similarity

Suprapto[1], Joseph A. Polela[2]
Department of Computer Science and Electronics
Universitas Gadjah Mada
Yogyakarta, Indonesia

***Abstract*—In a text matching similarity task, a model takes two sequence of text as an input and predicts a category or scale value to show their relationship. A developed model is to measure the similarity - one of relationship between those two text. The model is SIAMESE network that implement two copies of same network of CNN, it takes $text\_1$ and $text\_2$ as the inputs respectively for two CNN networks. The output of each CNN network is features vector of the corresponding text input, both outputs are then fed by a loss function to calculate the value of loss (i.e. similarity). This research implemented two types of loss functions, i.e. Triplet loss and Contrastive loss. The usage purpose of these two types of loss functions was to see the influence toward the measurement results of similarity between two text being compared. The metrices used for this comparison are $precision$, $recall$, and F1-$score$. Based on the experimental results done on 1500 pairs of sentences, and varied on the epoch value starting from 10 until 200 with an increment of 10, showed the best result was for epoch value of 180 with $precision$ 0.8004, $recall$ 0.6780, and F1-$score$ 0.6713 for Triplet loss function; and epoch value of 160 with $precision$ 0.6463, $recall$ 0.6440, and F1-$score$ 0.6451 for Contrastive loss function gave the best performance. So that, the Triplet loss function gave better influence than Contrastive loss function in measuring similarity between two given sentences.**

***Keywords*—Sentence; similarity; triplet; contrastive; CNN; Siamese; dataframe**

I. Introduction

The very fast growth of information nowdays causes a particular problem, such as an overwhelming of information [21]. It is very likely among those collections of huge of information found some similar ones, so that, they can be grouped into several classes based on their similarity. In order to overcome the overwhelming of information, each class will only be represented by a single information. Obviously, to identify the similar information requires a process called similarity measurement. A text similarity measurements is one of text mining approach that capable of coping with the information overwhelming. This process begins with finding similar word for sentece, then paragraph, and finally document [6]. Text similarity approach will ease people to find relevance information. It has a great support in successness for text mining operations such as, searching and information retrieval (IR), text classification, information extraction (IE), document clustering [8], sentiment analysis [4] [10] [16][3] [13], machine translation, text summarization, and natural language processing (NLP). Text similarity measurement may be done by comparing text - text matching. In text comparison tasks, a model takes two texts as inputs and predicts a category or a scale value indicates the relationship between those two texts. A big number of varieties of tasks such as, natural language inference [2] [11], paraphrase identification [17], answer selection [19] could be consider as special form of text matching problems. Recently, deep neural network is the most popular choice for text mining. Semantic alignment and comparison of two sequence of texts are the key of neural text matching. Most of previous deep neural network contain single inter-sequence alignment layer. In order to make the alignment process fully used, model must take many external syntaxtical features or aligment as additional inputs at alignment layer [5] [7], adopt a complex alignment mechanism [17], or build a big number of post-process layers to analyze alignment results [7].

This research proposed two models to compute similarity value between two texts using SIAMESE network in which each uses two different types of loss function - Triplet loss and Contrastive loss. The model consists of two copies of same CNN networks, where each recieves text (or sentence) as input. Subsequently, each CNN network results feature vector of each recieved text, and finally fed by loss function to compute the similirity value. Each model will be tested using the same dataset **Quora Question Pair similarity** taken from https://www.kaggle.com/c/quora-question-pairs. In order to see the influence of using these two types of loss function, the three metrics: $precision$, $recall$, and F1-$score$ will be computed.

II. Related Works

Deep neural network is very dominant in text matching area. While semantic alignment and comparison between two text sequences is the core of text matching [17]. The very beginning task explores encoding of each sequence individuallly into a vector and then bulds a neural network classifier over the two vectors. In this paradigm, recurrence [2], recursive [12], convolutional network [20] were used as sequence encoder. In this model, where encoding from a sequence independent with other sequence caused the last classifier had difficulty in modeling complex relations. Therefore, the subseqeunt tasks adopt framework matching aggregation to match two sequences at the lower level and aggregate results based on attention mechanism. DecomAtt used a simple form from attention for alignment and aggregate representations aligned by feed-foward network [15]. ESIM used a similar attention mechanism and implement bidirectional LSTMs as encoders and aggregators [5]. In order to improve model's performance, the researcher adopted three main paradigms. The first paradigm used syntacs that richer hand-writing features. HIM used syntactic parse tree [5]. The many usages of POS were found in previous tasks, some of them in [12]

[7]. The exact match with lemmatized tokens was reported as a powerfull binary features [7][12]. The second way was adding the complexity to alignment computation. BiMPM exploited an advanced multi-perspective matching operation [17], and MwAN implemented multi heterogeneous attention functions to calculate the alignment's results. The third way to improve model is by building heavy post-processing layers for alignment results. CAFE extracted additional indicators from allignment process using alignment factorization. DIIN adopted DenseNet as a deep convolutional feature extractor to filter information from alignment results [7]. The more effective models could be built when inter-sequence matching was allowed to be done more than once. CSRAN performed multi-level attention refinement with dense connections among many levels [12]. DRCN stacked encoding and alignment layers [12]. DRCN concatenated all previous alignment results and must've used autoencoder to cope with features space explotion. SAN exploited recurrent networks to combine many alignment results [14]. An architecture of deep based on new way relating contiguous blocks called by augmented residual connections to filter previous aligned information roling as inmportant features for text matching [18].

III. Methodology

This section presents the building of two Siamese networks each with **triplet loss** and **contrastive loss** functions.The training model for Siamese network with triplet loss function consists of three copies of same network of CNN, it takes $text\_1$, $text\_2$ and $text\_3$ as the inputs, while one with contrastive loss function consists of two copies only, it takes $text\_1$, and $text\_2$ as the inputs. However, the testing model for Siamese network both with triplet and contrastive loss function consist of two copies of same network of CNN, it takes $text\_1$, and $text\_2$ to be calculated their similarity. The dataset used to do training and testing was **Quora Question Pair similarity** taken from https://www.kaggle.com/c/quora-question-pairs. At the end, the three metrics: precision, recall, and F1-score were computed to see the influence of loss function's usage in each network.

A. Learning Model

Based on the objective of the research, the model was built with three main components: twin network, similarity function, and output layer.

1) **Twin Network**: most feature's extraction for text take place in this network. It has two copies of same networks, two networks share set of same weights. It capable of receiving two different inputs - two sequence of text. Each network in it is a convolutional text encoder. CNN network is used twice before performing backpropagation.
2) **Similarity function**: two outputs from twin netork representing features learned automatically from two compared text was fed by a layer. Subsequently, distance formula was used to compare the similarity between two text in *n*-dimensional space.
3) **Output layer**: last layer whose single neuron connecting *n* neurons from the results of previous similarity function. The role of this part of the model was to decide the probability of tested text to be a member the text used as reference. The probability was computed using **Sigmoid** function as in equation (1).

$$p = \sigma(\Sigma_j \sigma_j |h_j^1 - h_j^2|) \tag{1}$$

where h_j^1 and h_j^2 are values of j-th neuron of first twin network and second twin network respectively, σ_j is weight between neuron output and j-th neuron in similarity layer.

The structure of **Siamese** network is shown in Fig. 1 [23].

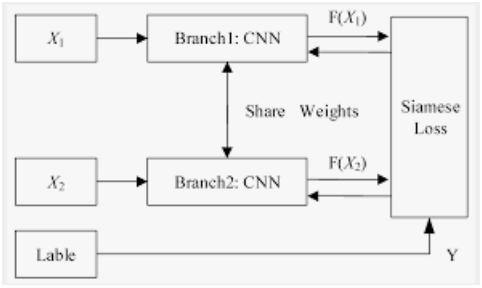

Fig. 1. The Structure of **Siamese** Network

The two types of loss function are implemented in the research, namely, **triplet** and **contrastive**. The aim of this implementation is to find the influence of loss function usage in similarity value computed between two text.

a. **Triplet Network**
A **triplet** network is comprised of three instances of the same feedforward network (with shared parameters). When fed with three samples, the network outputs two intermediate values - the L_2 distances between the embedded representation of two of its inputs from the representation of the third. The three inputs will be denoted as x, x^+ and x^-, and the embedded representation of the network as $Net(x)$. In words, this encodes the pair of distances between each of x^+ and x^-, against the reference x, i.e., $||Net(x) - Net(x^-)||_2$ and $||Net(x) - Net(x^+)||_2$. The distance between reference input x and positive input x^+ was minimized, on the other hand, the distance between reference input x and negative input x^- was maximized. The structure of **triplet** network is shown in Fig. 2 [9].

b. **Contrastive Network**
The **contrastive** loss function takes output from network for positive sample and computes the distance to an example from the same class and contrast it with distance to negative examples. In other word, the value of loss is low if positive samples are encoded to similar representations (closer), and negative examples are encoded to different representations (farer). The formula used to compute distance was **cosine similarity** distance as shown in equation (2).

$$Loss(x_p, x_q, y) = y * ||x_p - x_q||^2 + (1 - y) * max(0, m^2 - ||x_p - x_q||^2) \tag{2}$$

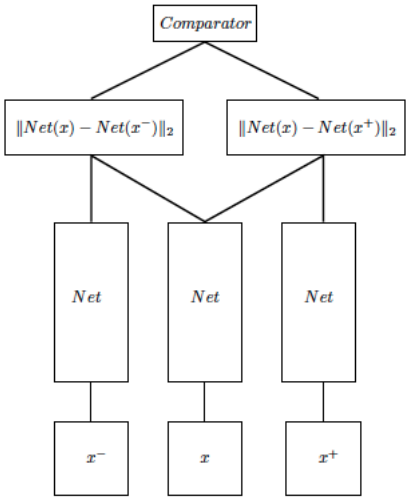

Fig. 2. The Structure of **Triplet** Network

	id	qid1	qid2	sent_1	sent_2	is_duplicate
337503	337503	18052	6721	How can the ban of 500 and 1000 rupee notes in...	What are the economic implications of banning ...	1

distance	tokenized_sent_1	tokenized_sent_2	tokenized_sent_3
0	[how, can, the, ban, of, and, rupee, notes, in...	[what, are, the, economic, implications, of, b...	[what, does, bigly, mean]

Fig. 3. The Format of Training Dataframe

	id	qid1	qid2	sent_1	sent_2
314821	314821	439623	439624	Which exam is more tough NDA or IIT?	How do I crack tough exams?

is_duplicate	distance	tokenized_sent_1	tokenized_sent_2
0	1	[which, exam, is, more, tough, nda, or, iit]	[how, do, i, crack, tough, exams]

Fig. 4. The Format of Testing Dataframe

IV. Results and Discussion

As previously mentioned, the dataset used in the research was public. It was taken from https://www.kaggle.com/c/quora-question-pairs. The dataset was in CSV file containing 5000 pairs of Quora questions. The dataset was read to generate dataframe, and then the generated dataframe was devided into two: 70% for training and the other 30% for testing. The next process was building Word2Vec model using Gensim for embedding layer of deep learning. Then define function to transform sentence into vector containing index of Word2Vec's vocabularies. Another two important functions to be defined were **triplet** and **contrastive** loss, continued to define the CNN Siamese network with both **triplet** and **contrastive** loss functions, and then trained it.

A. Dataframe

The dataframe was generated in several steps: first applying simple process both to question 1 ($sent\_1$) and question 2 ($sent\_2$), to do this there was a function $simple\_process$ taken from https://radimrehurek.com/gensim/utils.html #gensim.utils.simple_preprocess for tokenization. This process was done fao all pairs of sentences in the dataset. From these all tokenized pairs of sentence, the number of tokens from the longest sentence was calculated based on the *mean* and *deviation standard*. This value was used to do padding, to made all tokenized pairs of sentences had the same length. In order to train the Siamese CNN model with triplet loss function, a negative sentence (the third tokenized sentence) must have been added to each pair of tokenized sententece (i.e. tokenized_sent_1 and tokenized_sent_2) for training dataframe. The format of training dataframe is shown in Fig. 3.

However, in the testing process for Siamese CNN model even with triplet loss function only need two tokenized sentences (tokenized_sent_1 and tokenized_sent_2). The dataframe format for testing was shown in Fig. **??**.

B. Word2Vec Model

Before building the CNN **Siamese** network, the Word2Vec model was built using Gensim for embedding layer in deep learning. It was built with CBOW (https://iksinc.online/tag/continuous-bag-of-words-cbow/) with 20 iterations, vector length of 100, and window size was 5. The Word2Vec was trained once using string of words (tokenozed sentence as a result of concatenation between tokenized of sentence 1 and tokenized of sentence 2), and saved it. During the process of training the model, there was a function required to convert a sentence into a vector containing Word2Vec vocabulary indices. For the model using **triplet** loss function, the converter changed three tokenized sentences (i.e. tokenized_sent_1, tokenized_sent_2, tokenized_sent_3) into three vectors containing indices of Word2Vec vocabularies (i.e. sent_1_ids, sent_2_ids, sent_3_ids), and two tokenized sentences (i.e. tokenized_sent_1, tokenized_sent_2) into two vectors containing indices of Word2Vec vocabulary (i.e. sent_1_ids, sent_2_ids) respectively in training and testing processes. However, for the model using **contrastive** loss function, the converter changed two tokenized sentences (i.e. tokenized_sent_1, tokenized_sent_2) into two vectors containing indices of Word2Vec vocabularies (i.e. sent_1_ids, sent_2_ids) in both training and testing processes. The sample of conversion results was shown in Fig. 5.

While the definition of the function that convert list of tokens to list of indices was shown in Fig. 6.

C. Triplet Loss

The **Triplet** loss function takes three inputs: baseline (an anchor sentence), positive (true sentence - one closest to an anchor), and negative (false sentence - one farest to an anchor). Therefore, the objective of this function is to minimize the

	is_duplicate	distance	tokenized_sent_1	tokenized_sent_2	sent_1_ids	sent_2_ids
0	1		[why, don, t, we, use, conveyor, belts, to, sh...	[how, likely, is, that, the, us, or, russia, w...	[15, 106, 43, 52, 75, 14103, 10068, 6, 18462, ...	[4, 852, 2, 28, 0, 105, 22, 626, 36, 3063, 77,...
1	0		[is, it, a, good, idea, to, buy, a, used, car,...	[what, are, the, pros, and, cons, of, buying, ...	[2, 16, 5, 42, 422, 6, 123, 5, 129, 231, 33, 5...	[1, 10, 0, 622, 11, 624, 9, 937, 5, 129, 231, ...

Fig. 5. The Conversion Result Sample

```
def tokens2ids(tokens, vec_len):
    padding_idx = word2vec.wv.vocab['_pad'].index
    ids = [padding_idx] * vec_len

    for i, token in enumerate(tokens):
        if token in word2vec.wv.vocab:
            ids[i] = word2vec.wv.vocab[token].index

    return ids
```

Fig. 6. The Definition of Converter Function

distance between an anchor sentence with a positive sentence, on the other hand, maximize the distance between an anchor sentence with a negative sentence. The implementation of the **triplet** function was shown in Fig. 7.

```
class TripletLoss(nn.Module):
    def __init__(self, margin=1.0):
        super(TripletLoss, self).__init__()
        self.margin = margin

    def forward(self, output1, output2, output3):
        loss = torch.pow(\
            F.pairwise_distance(output1,output2), 2)+\
            torch.pow((F.pairwise_distance(output1,output2)-\
                       F.pairwise_distance(output1,output3)+\
                       self.margin), 2)
        return loss.sum()
```

Fig. 7. The Implementation of Triplet Loss Function

D. Contrastive Loss

Unlike the **triplet**, the **contrastive** loss function only takes two inputs: positive sample and negative example. The objective is to contrast the distance between positive sample and an example from the same class with the distance between positive sample and negative example.

The implementation of the **triplet** function was shown in Fig. 8.

```
class ContrastiveLoss(nn.Module):
    def __init__(self, margin=2.0):
        super(ContrastiveLoss, self).__init__()
        self.margin = margin

    def forward(self, output1, output2, y):
        distance = F.pairwise_distance(output1,output2)
        loss = ((1 - y) + torch.pow(distance, 2) +\
                torch.pow(torch.clamp(\
                    self.margin-distance,min=0.0,2)*y)/2
        return loss.sum()
```

Fig. 8. The Implementation of Contrastive Loss Function

These two functions were only used during the training process to update the weights of network in order to converge; the **triplet** loss used two distances, and the **contrastive** loss only used one distance. While during the testing, no loss function needed, but the distance between two sentences. Because the model simply used the final weights resulted from training process.

E. CNN Siamese Network

The CNN **Siamese** network was built with the following specifications:

1) The embedding layer was the Word2Vec model that had already been trained.
2) Three 2-dimensional convolutional layers with configurable hyperparameters.
3) For every convolutional layer used $tanh$ as an activation function, dropout, and maxpool layer.

The architecture of CNN that was implemented in the CNN Siamese or just Siamese model was shown in Fig. 9 [22].

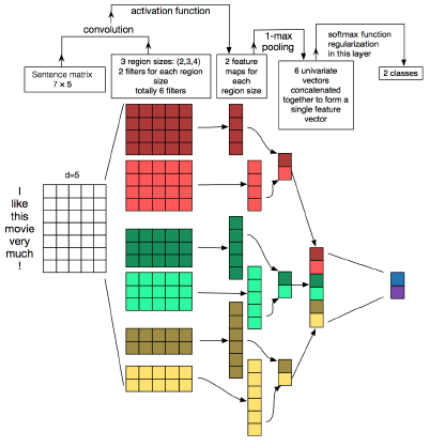

Fig. 9. The Architecture of Implemented CNN

The hyperparameter of the CNN **Siamese** network was shown in Fig. 10.

```
BATCH_SIZE = 64
EMBEDDING_DIM = word2vec.wv.vectors.shape[1]
EPOCHS = 200
#(in_channels, out_channels, kernel_size, dilation, padding)
CONVS_PARAMS = (
    (1, 100, (1, EMBEDDING_DIM), 1, (0, 0),),
    (1, 100, (3, EMBEDDING_DIM), 1, (1, 0),),
    (1, 100, (5, EMBEDDING_DIM), 1, (2, 0),)
)
LEARNING_RATE = 0.0001
MARGIN = 1
THRESHOLD = 1
```

Fig. 10. The CNN **Siamese**'s Hyperparameter

The CNN **Siamese** model with **triplet** loss function consisting of three exact same models of CNN was trained using triple tokenized sentences (i.e. tokenized_sent_1, tokenized_sent_2, and tokenized_sent_3) from the dataframe training. The output of each CNN model was fed by **Triplet** loss function to calculate the loss value. On the other hand, the CNN **Siamese** model with **contrastive** loss function consisting of two exact same models of CNN was trained

using pair of tokenized sentences (i.e., tokenized_sent_1, and tokenized_sent_2) from the dataframe training. The output of each CNN model was fed by **contrastive** loss function to calculate the loss value. Finally, both model of CNN **Siamese** with **triplet** and **contrastive** loss function respectively were tested by feeding two tokenized sentences to be calculated their similirity value. Both testing were conducted using the same number of 1500 pairs of sentences (Quora questions), and varied for the values of EPOCH starting from 10 until 200 with an increment of 10 to obtain the values of three metrics: precision, recall and F1-score. The values of three matrics for each CNN **Siamese** model were shown respectively in Table I and Table II.

TABLE I. THE PRECISIONS, RECALLS AND F1-SCORES OF THE CNN SIAMESE WITH TRIPLET

Epoch	Training Loss	Precision	Recall	F1-Score
10	64.3868	0.3762	0.6133	0.4663
20	19.4053	0.3762	0.6133	0.4663
30	8.6268	0.6277	0.6187	0.4896
40	4.9022	0.6469	0.6447	0.5753
50	3.2373	0.6762	0.6827	0.6602
60	2.3511	0.7073	0.7127	0.7074
70	1.8006	0.7254	0.7207	0.7224
80	1.4510	0.7433	0.7253	0.7288
90	1.2047	0.7526	0.7160	0.7194
100	1.0458	0.7520	0.6980	0.7003
110	0.9170	0.7562	0.6873	0.6880
120	0.8245	0.7746	0.6960	0.6956
130	0.7459	0.7774	0.6860	0.6838
140	0.6954	0.7834	0.6807	0.6768
150	0.6398	0.7892	0.6800	0.6752
160	0.6074	0.7885	0.6707	0.6642
170	0.5824	0.7907	0.6707	0.6639
180	0.5589	0.8004	0.6780	0.6713
190	0.5404	0.7973	0.6700	0.6621
200	0.5203	0.7986	0.6733	0.6659

TABLE II. THE PRECISIONS, RECALLS AND F1-SCORES OF THE CNN SIAMESE WITH CONTRASTIVE

Epoch	Training Loss	Precision	Recall	F1-Score
10	16.2760	0.3762	0.6133	0.4663
20	5.8481	0.3762	0.6133	0.4663
30	2.6012	0.3760	0.6127	0.4660
40	1.4245	0.6290	0.6193	0.4921
50	0.8705	0.6370	0.6333	0.5434
60	0.5853	0.5921	0.6200	0.5508
70	0.4222	0.6068	0.6287	0.5824
80	0.3151	0.6113	0.6313	0.5976
90	0.2358	0.6071	0.6260	0.6030
100	0.1795	0.6199	0.6347	0.6190
110	0.1423	0.6246	0.6367	0.6258
120	0.1149	0.6283	0.6367	0.6305
130	0.0934	0.6332	0.6400	0.6354
140	0.0775	0.6331	0.6360	0.6344
150	0.0653	0.6422	0.6427	0.6424
160	0.0566	0.6463	0.6440	0.6451
170	0.0498	0.6336	0.6300	0.6315
180	0.0449	0.6436	0.6387	0.6407
190	0.0404	0.6299	0.6227	0.6254
200	0.0370	0.6440	0.6353	0.6384

In accordance with the testing results, it seemed that an ordinary CNN did not fit enough for this dataset even though each trained epoch showed its convergence, but the validation results were not good. This was caused either by the difference between distribution of words in the training data and the one in testing data or an ordinary CNN model found difficulty to distinguish two sentences with very similar structures, but in fact they were different semantically. For instance, "What's your favorite political bumper sticker?" with "What was your favorite bumper sticker in the 2000s?". These two sentences was taged "not similar", even their structures were similar, and there were some others. Conceptually, a CNN usually performs convolution with kernel. The kernel size used in this model was [1, 3, 5] multiplied by the size of embedding yielded by Word2Vec. If there are similar sentence structures, the result of the convolution will only be different in some elements of convolution's results matrix. So that, the similarity between matrices of convolution's results from two sentence inputs will be high (i.e., two sentences are similar). Even though dropout and regularization had been implemented, this problem would still happened, because there were pair of sentences with similar structures would be considered either similar or not similar. In order to solve the problem it needs a language model and a more complex network. In this experiment, Word2Vec yielded vector of word based on surrounding words. It was not contextual, it means that one word would be represented by a vector that always the same.

From both Table I and Table II was derived chart comparing the two loss functions (triplet and contrastive) each for metric *precision*, *recall* and F1-*score*. Fig. 11, 12 and 13 show the chart of comparing metric between two loss functions, respectively for *precision*, *recall* and F1-*score*.

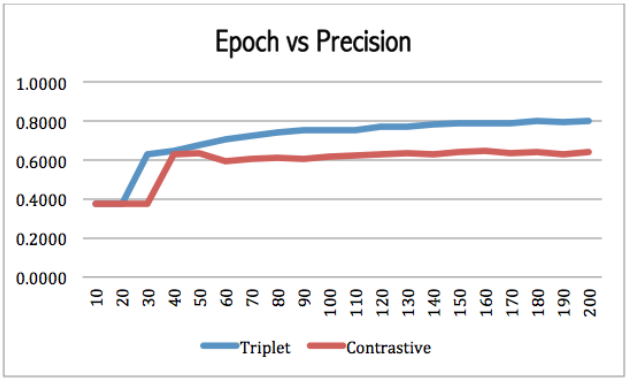

Fig. 11. The Chart of *Epoch* versus *Precision* for Triplet and Contrastive Loss Function

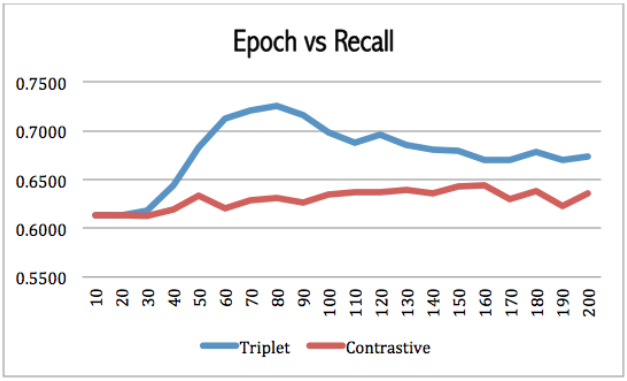

Fig. 12. The Chart of *Epoch* versus *Recall* for Triplet and Contrastive Loss Function

The three metric, *precision*, *recall* and F1-*score* were

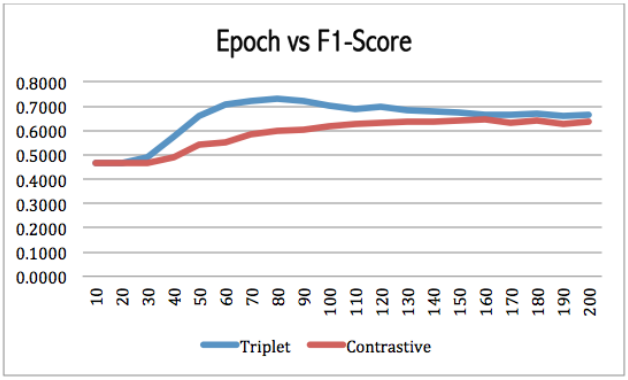

Fig. 13. The Chart of *Epoch* versus F1-*Score* for Triplet and Contrastive Loss Function

computed using the formula derived from a confusion matrix.

V. Conclusions

The two CNN **Siamese** models had been succesfully built, one with **triplet** loss function and the other with **contrastive** loss functions. The size of dataset used for training and testing were 3500 and 1500 respectively for the total of 5000. Both models were treated equally either in training or testing processes. In the training process, the model with **triplet** loss function consisted of three exact the same CNN models, while one with **contrastive** loss function only consisted of two exact the same CNN models. From the results of the model testing, it was seen that among 20 values of epoch there was one with epoch value of 180 with $precision$ 0.8004, $recall$ 0.6780, and F1-$score$ 0.6713 for **triplet** loss function; and epoch value of 160 with $precision$ 0.6463, $recall$ 0.6440, and F1-$score$ 0.6451 for **contrastive** loss function gave the best performance. So that, the **triplet** loss function gave better influence than **contrastive** loss function in measuring similarity between two given sentences.

References

[1] Bird, S., at.al, 2009, Natural Language Processing with Python, Published by O'Reilly Media, Inc., 105 Gravenstein Highway North, Sebastopol, CA 95472.

[2] Bowman, S. R., Angeli, G., Potts, C. and Manning, C. D., 2015, "A large annotated corpus for learning natural language inference". In Proceeding of the 2015 on Empirical Methods in Natural Language Processing, pages 632 - 642, Lisbon, Portugal. Association for Computational Linguistics.

[3] Bravo-Marquez, F., Mendoza, M., and Poblete, B., Meta-level sentiment models for big social data analysis, Knowl.-Based Syst. 69 (2014) 86–99, http://dx.doi. org/10.1016/j.knosys.2014.05.016.

[4] Chauhan, V. K., Bansal, A. and Goel, A., Twitter Sentiment Analysis Using Vader, International Journal of Advance Research, Ideas and Innovations in Technology. vol. 4, 1 (2018), 485-489.

[5] Chen, Q., Zhu, X., Ling, Z. H., Wei, S., Jiang, H. and Inkpen, D., 2017, "Enhanced LSTM for Natural Language Inference". In Proceedings of The 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages: 1657 - 1668; Vancouver, Canada. Association for Computational Linguistics.

[6] Gomma, W. H. and Fahmy, A. A., "A Survey of Text Similarity Approaches," Int. J. Comput. Appl., vol. 68, no. 13, 2013, doi: https://doi.org/10.5120/11638 - 7118.

[7] Gong, Y., Luo, H. and Zhang, J., 2018, "Natural Language Inference over Interaction Space", In Proceedings of the 6th International Conference on Learning Representations.

[8] Hidayat, E. Y., Firdausillah, F., Hastuti, K., Dewi, I. N. and Azhari, A., "Automatic Text Summarization Using Latent Drichlet Allocation (LDA) for Document Clustering," Int. J. Adv. Intell. Informatics, vol. 1, no. 3, p. 132, Dec. 2015, doi: https://doi.org/10.26555/ijain.v1i3.43.

[9] Hoffer, E., and Ailon, N., "Deep Metric Learning using Triplet Network" Accepted as a workshop contribution at ICLR 2015.

[10] Hutto, C. J. and Gilbert, E., VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Copyright © 2014, Association for the Advancement of Artificial Intelli-gence (www.aaai.org).

[11] Khot, T., Sabharwal and Clark, P., 2018. SciTail: A Textual Entailment Dataset from Science Question Answering. In Proceedings of AAAI.

[12] Kim, S., Hong, J. H., Kang, I. and Kwak, N., 2018, "Semantic Sentence Matching with Densely-connected Recurrent and Co-attentive Information". Computing Research Repository, arXiv: 1805.11360. Version 2.

[13] Liu, B.and Zhang, L., A survey of opinion mining and sentiment analysis, in: Mining Text Data, Springer, 2012, pp. 415–463, http://dx.doi.org/10.1007/978-1- 4614-3223-4_13.

[14] Liu, X., Duh, K. and Gao, J., 2018, "Stocastic Answer Networks for Natural Language Inference". Computing Reseach Repository, arXiv: 1804.07888.

[15] Parikh, A., T., Tackstrom, O., Das, D. and Uszkoreit, J., 2016, "A Decomposable Attention Model for Natural Language Inference". In Proceedings of the 2016 Conference of Empirical Methods in Natural Language Processing (EMNLP), pages 1532 - 1543, Doha, Qatar. Association for Computational Linguistics.

[16] Ravi, K., and Ravi, V., A survey on opinion mining and sentiment analysis: Tasks, approaches and applications, Knowl.-Based Syst. 89 (2015) 14–46, http://dx. doi.org/10.1016/j.knosys.2015.06.015.

[17] Wang, S. and Jiang, J., 2017. "A Compare-Aggregate Model for Matching Text Sequences". In Proceedings of the 5th International Conference on Learning Representations.

[18] Yang, R., Zhang, J., Gao, X., Ji, F. and Chen, H., 2019, "Simple and Efective Text Matching with Richer Alignment Features". In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4699 - 4709, Florence, Italy, July 28 - August 2, 2019. Association for Computational Linguistics.

[19] Yang, Y., Yih, W. and Meek, C., 2015, "WikiQA: A Chalenge Dataset for Open-domain Question Answering." In Proceeding of The 2015 Conference on Empirical Methods in Natural Language Processing", pages 2013 - 2018, Lisbon, Portugal. Association for Computational Linguistics.

[20] Yu, L., Hermann, K. M., Blunsom, P. and Pulman, S., 2014, "Deep Learning for Answer Sentence Selection". In NIPS Deep Learning and Representation Learning Work shop, Montreal.

[21] Yunianta, A., Barukah, O.M., Yusof, N., Dengen, N., Haviluddin, H. and Othman, S., "Semantic data mapping technology to solve semantic data problem on heterogeneity aspect," Int. J. Adv. Intell. Informatics, vol. 3, no. 3, pp. 161 - 172, Dec. 2017, doi: https://doi.org/10.26555/ijain.v3i3.131.

[22] https://miro.medium.com/max/2800/0*0efgxnFIaLTZ2qkY, "The Architecture of CNN".

[23] https://www.researchgate.net/publication/337578683_A_Survey_of_Vehicle_Re-Identification_Based_on_Deep_Learning/figures?lo=1, "Schematic diagram of the Siamese Network structure".