EVOLUTIONARY COMBINATORIAL OPTIMIZATION FOR WORD EMBEDDING (ECOWE) IN SENTIMENT CLASSIFICATION

Thineswaran Gunasegaran[1*] and Yu-N Cheah[2]

[1,2]School of Computer Sciences
Universiti Sains Malaysia
11800 USM Penang, Malaysia

Email: thineswaran@gmail.com[1]*(corresponding author), yncheah@usm.my[2]

DOI: https://doi.org/10.22452/mjcs.sp2019no3.3

ABSTRACT

Word embedding is widely used in various natural language processing (NLP) tasks, especially sentiment classification. Huge computational costs for training new embeddings from scratch have made pretrained word embeddings such as Word2Vec and Glove popular for reuse of word vectors. Inadequacy of a single embedding necessitated newer techniques to combine two embeddings. However, the combined embeddings proposed in existing works are static and only provide a one-size-fits-all solution regardless of the problem and dataset at hand. Optimization is a more promising technique to overcome the limitations of simplistic techniques in existing works related to combined word embeddings because optimization provides unique and optimal solutions according to the problem and dataset at hand. In this paper, a new genetic algorithm based combinatorial optimization algorithm called Evolutionary Combinatorial Optimization for Word Embedding (ECOWE) is proposed to produce combinations of word embeddings, which yield optimal accuracy for the specific sentiment classification dataset that is used. Results show that absolute percentages of improvement ranging from 1.7% to 12.9%, averaging around 5.5% and relative percentages of improvement ranging from 2.4% to 19.5%, averaging around 8.1% have been achieved over the benchmark model accuracy values for all datasets. The ECOWE accuracy values for all datasets have also been found to be statistically significant compared to benchmark accuracy values with a z-score of -2.2014 using two-tailed Wilcoxon signed rank test with 5% significance level.

Keywords: Evolutionary Computation, Genetic Algorithm, Combinatorial Optimization, Word Embedding, Sentiment Analysis, Sentiment Classification

1.0 INTRODUCTION

Sentiment classification is a subarea under sentiment analysis, which also consists of many other subareas such as subjectivity detection and emotion detection from text [1, 2]. Sentiment classification is particularly focused on classifying text inputs in a document according to the document's polarity, i.e. the ratio of positive sentiment terms to negative sentiment terms [3]. Therefore, sentiment classification is predominantly a binary classification task, though multiclass sentiment labels are also considered in certain cases. The most high level classification of sentiment analysis or sentiment classification consists of lexicon based approach, machine learning based approach and hybrid approach [4]. An inherent characteristic feature of a machine learning model is that the model is capable of making more accurate predictions when it is fed with more training data. Social media platforms with loads of opinionated text data on sites like Twitter provide a fertile ground for training sentiment classification models. Unfortunately, modeling such data using machine learning algorithms has always been a chore because handcrafted features had to be extracted and selected from the raw data before training the model. Deep learning had completely changed the scenario with its ability to automatically learn salient features from raw image and text data. This characteristic feature of the deep learning paradigm is known as representation learning, i.e. learning representations or features from raw input data before proceeding to learn the patterns between these features [5]. Representation learning involves multiple processing units or layers of different abstraction levels in a hierarchical manner, where input at the current processing layer produces output to be served as input for the next layer that is more abstract. The single biggest strength of the deep learning paradigm is the ability to learn representation from input data in such a way that features are automatically extracted and the most optimal feature subsets are selected for training.

Word embedding had produced astoundingly great accuracies for various NLP tasks including sentiment classification, thus making it a very popular choice among practitioners [6, 7]. Using a word embedding is as simple as plugging in the continuous word vectors for every word in the dataset's vocabulary, so that the entire raw text input gets converted into a set of feature vectors. Hence, pretrained word embedding is preferred because it is

available as a separate component, which can be used for various NLP tasks such as named entity recognition (NER), part-of-speech (POS) tagging, text classification and sentiment classification. As stated earlier, it is very easy to use a word embedding once it is has been trained because the continuous word vectors can just be mapped into the dataset's vocabulary to convert words to real valued vectors. This is what makes pretrained word embeddings such as Word2Vec and Glove to be popularly used for a wide variety of NLP tasks. Unfortunately, practitioners are often puzzled when it comes to choosing the best embedding for the particular task at hand mainly because of the variety that exists. Interestingly, recent works have proposed to use combinations of more than one pretrained word embedding for a particular task [8, 9]. However, the manner in which these combinations work are rather simplistic because the combined word embedding is produced merely by stacking two embeddings such as Word2Vec and Glove. Such techniques only produce a one-size-fits-all combined word embedding that would remain the same, regardless of the task and dataset for which it is used. Sentiment classification datasets vary in terms of average length of sentences, vocabulary size, the number of sentiment labels (binary or multiclass) and also the type of domain from which the dataset was generated from. The one-size-fits-all combined word embedding approach proposed in existing works offer no scope for optimizing the combination of embeddings in such a way that optimal results could be produced for different sentiment classification datasets.

In this paper, an evolutionary combinatorial optimization algorithm is proposed to find the most optimal combination of word embeddings for sentiment classification. The optimal combination is obtained by combining two pretrained word embeddings so that individual word vectors for a specific sentiment classification dataset come from an optimized combination of pretrained word embeddings. This method is assumed to be better than previous works related to improving word vectors for sentiment classification mainly because it enables optimization according to the task and dataset at hand, in stark contrast to the one-size-fits-all combined word embeddings produced in previous works. Forthcoming sections of this paper are organized as follows: Related Work section is a review of previous works related to the application of word embedding for sentiment classification, specially focusing on gaps in those works. Methodology section details intricacies of the methodology of the ECOWE algorithm. Evaluation section presents results for experimental evaluation of the algorithm. Discussion section focuses on analytical discussion of the evaluation results. Lastly, the Conclusion section contains concluding remarks about the work presented in this paper and hints at possible directions for future work.

2.0 RELATED WORK

Word embedding is a representation technique, which embeds or projects words as real valued vectors on a low dimensional continuous vector space. Word vectors are produced in the form of a distributed representation where words with similar meanings appear closer to each other on the vector space. An important point to be noted is that word embedding projects words to a low dimensional vector space. Therefore, dimensionality of the embedding can always be controlled to ensure that it does not result in a square embedding matrix of |V| x |V|, where |V| is the vocabulary size of a language model. This had always been a major issue with traditional language models such as bag-of-words (BOW) because words are represented using one hot encoding. In one hot encoding, if the vocabulary size is |V|, then the bit-string used to represent each word would also be of length |V|. This results in the BOW model having a |V|x|V| square matrix representation, which is extremely vulnerable to the curse of dimensionality. For a *d*-dimensional word embedding, the embedding matrix would always be a |V| x *d* matrix with *d* being always less than |V|, thus alleviating the curse of dimensionality. Hence, word embedding solves some of the core issues imposed by traditional language modeling techniques. An example of word embedding visualization is shown in Fig. 1.

Many pretrained word embeddings are available on the Internet, even in multilingual versions thanks to the rapid adoption of word embedding in NLP areas like machine translation. Training models with supervised learning algorithms is the most common approach for sentiment classification, which involves the usage of a classifier and labelled data with pretrained word embeddings. Using this approach, raw text data can be converted into feature vectors because the word embedding algorithm acts as a feature extractor. Recurrent neural network (RNN), long short term memory (LSTM) and gated recurrent unit (GRU) with their ability of learning long range dependencies have been proven to be very useful for sentiment classification just like other text classification tasks because sentiment classification is also a special type of text classification task [1]. These language modeling algorithms can also be used to produce new word embeddings out of raw text data if pretrained word embeddings are not available. However, reuse of pretrained word embeddings is more preferred given the computational costs and overhead for training new embeddings from scratch, especially using deep learning algorithms [10, 11].

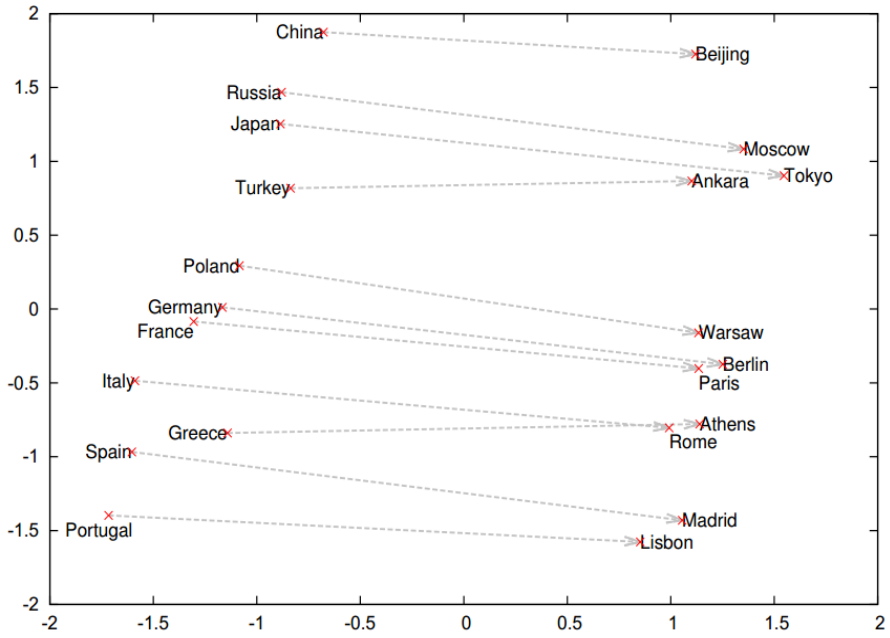

Fig. 1: Two-dimensional projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities [7]

Apart from using just a single embedding, works related to combination of embeddings have also been pursued to produce improved word vectors (IWV) [8]. This approach combines three techniques of word vector generation which include:

- Reuse of pretrained Word2Vec and Glove embeddings: 300 dimensional embeddings are stacked to form a single combined embedding.
- POS2Vec: Part-of-speech (POS) tag denotes syntactic categories of words such as nouns, verbs, adjectives and adverbs. This information can be used to represent similarities and dissimilarities between words. The POS2Vec technique converts every word based on its POS tag to 50-dimensional vectors. These vectors are concatenated with the 300-dimensional combination of Word2Vec and GloVe vectors to enrich those vectors with syntactic information of words. The result is a 350 dimensional embedding.
- Lexicon2Vec: Similar to POS2Vec, Lexicon2Vec uses six sentiment and emotion lexicons to assign a six dimensional vector to every word. These vectors are concatenated with the 350 dimensional embedding to form the final 356 dimensional word embedding.

IWV was evaluated on three sentiment classification datasets namely Movie Reviews (MR) [12], Customer Reviews (CR) [13] and SST-2 (binary) [14] datasets. IWV had outperformed baseline models which used only Word2Vec or Glove and also several other related models. This hints at a promising research direction, where existing pretrained word embeddings can be improved further before being used for the specific task at hand.

Apart from the survey in [9], more recent works which have used pretrained word embeddings for sentiment classification include Emo2Vec [15] and Self-Attention Networks (SAN) [16]. Both Emo2Vec and SAN were evaluated using six sentiment classification datasets that are in common with those used for the survey in [9]. Results in [9] show that the JOINT, LSTM and BiLSTM models produce the best accuracies. The BiLSTM model works best for all three multiclass sentiment classification datasets, i.e. OpeNER [17], SemEval 2013 [18] and SST-1 (multiclass) [14] datasets.

Emo2Vec embeddings are 100 dimensional embeddings pretrained from a Twitter hashtag corpus using Convolutional Neural Network (CNN) model. In addition to gains in accuracy achieved by Emo2Vec alone, the concatenated embedding of Glove and Emo2Vec (Glove+Emo2Vec) also produced better accuracy using a logistic regression classifier. Similar to IWV, Glove+Emo2Vec also shows that improved pretrained embeddings can produce better results with simple machine learning algorithms.

SAN is based on attention mechanism, which was proposed to improve RNN encoder decoder sequence-to-sequence architecture for neural machine translation (NMT). Self-attention applies the attention mechanism to every position of the source sequence of words to create query-key-value (QKV) vectors. In RNN encoder decoder models, the key-value pairs are the encoder hidden states, while the query comes from the decoder's previous output. Hence, the SAN model converts words into QKV vectors like how word embedding models convert words into word vectors. The SAN model was also evaluated on the same set of sentiment classification datasets in [9] and [15] using pretrained word embeddings of various dimensions. SAN had produced better accuracy values than the best accuracy values reported in [9] and [15].

Recent works increasingly show a trend towards improving pretrained word embeddings in various ways rather than just using the original embedding straightaway [8, 9, 15, 16]. This also means that training new embeddings from scratch for every new problem is not feasible, hence making the research on improving word embeddings more significant. Optimization of pretrained word embeddings remains an unexplored area. Optimization differs from improvement of word embedding such as IWV because optimization could create optimal word embeddings according to the problem at hand, unlike improved word embeddings, which provide a one-size-fits-all solution ignoring specific details of the problem. Improving and optimizing word embeddings ultimately aims to create high quality feature vectors. This task closely resembles feature subset selection, which had been widely explored even before the emergence of deep learning. Ironically, optimization of pretrained word embeddings had not been explored, though a strong basis for it exists through the myriad of works related to feature subset selection. Popularity of evolutionary computation based feature subset selection also substantiates the potential for applying evolutionary computation to optimize word embeddings.

Apart from that, most of the existing works related to sentiment classification using pretrained word embeddings are too concentrated around deep learning models [10, 11]. The main purpose of reusing pretrained word embeddings is to create feature vectors for text data by mapping words to word vectors. Once the feature vectors have been created, any machine learning algorithm can be used for modeling. This research focuses on optimization of word vectors in pretrained word embeddings using basic machine learning algorithms such as support vector machine (SVM), thus shifting away from the usual trend of using deep learning algorithms. Shifting away from deep learning is required to keep the spotlight on optimized word embeddings because the actual significance of an optimized word embedding can only be assessed fairly using machine learning algorithms. Improved accuracy in the absence of deep learning models could be fully attributed to the optimal word embedding alone if a machine learning algorithm is used for modeling. Practically, reusing pretrained word embeddings with machine learning algorithms is the most sensible thing to do because if an optimized word embedding is truly optimal enough, complex deep learning models would not be required as prescribed by the Occam's razor principle [19].

Considering the gaps in existing works and the promising research direction that lies ahead, this research is focused on applying evolutionary computation based optimization to produce optimal word embeddings for sentiment classification. The next section details the methodology of this research.

3.0 METHODOLOGY

Optimization of pretrained word embeddings is similar to feature selection, a prominent task in machine learning prior to the emergence of deep learning. Metaheuristics based optimization methods such as evolutionary computation (EC) and swarm intelligence (SI) have been widely used for feature selection, thus proving the effectiveness of those methods for solving this task [20]. Since optimization of pretrained word embeddings is also a task that closely resembles feature selection, the same metaheuristics based optimization methods can also be used for this task.

Inspired by Darwin’s theory of evolution, EC algorithms were designed to mimic the way living beings evolve by adapting to their environment. A population of individuals (solutions) evolve over generations (iterations) as they undergo modifications through recombination and mutation operators. During every generation, the fitness of all individuals in the population will be evaluated using a fitness function (objective function). Individuals with high fitness get selected as parents to produce offsprings via recombination. Then, the newly generated offsprings are mutated to produce variants of these offsprings. A certain number of these individuals will be selected based on the population size to form the new population in the next generation. Algorithm's execution ends once the termination condition had been fulfilled. Generally, the termination condition is based on a predefined maximum number of generations or fitness evaluations. EC algorithms (a.k.a. evolutionary algorithms (EAs)) consist of different variants

such as genetic algorithm (GA), evolution strategies (ES), evolutionary programming (EP) and genetic programming (GP). Fig. 2 shows the core components and typical flow for EAs.

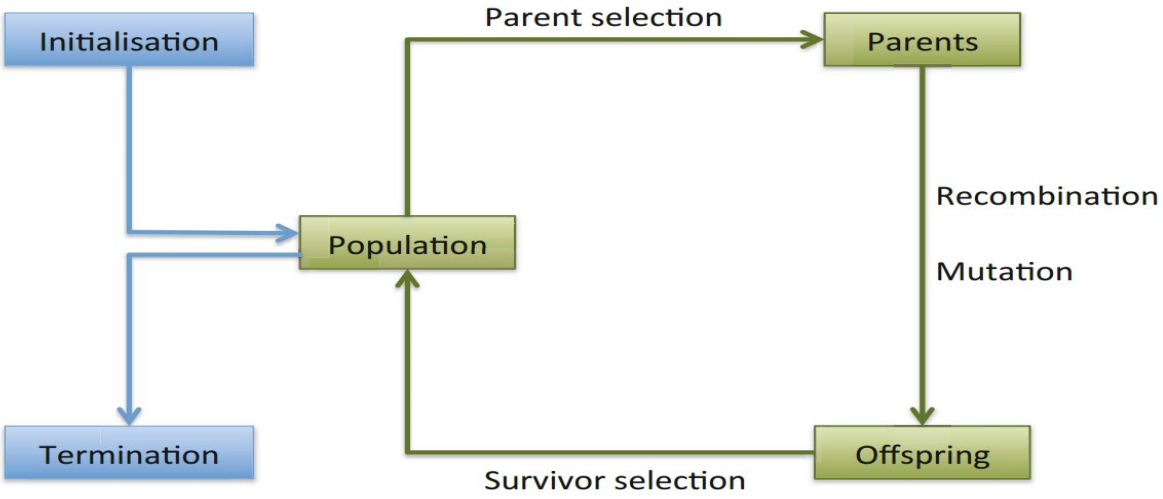

Fig. 2: General scheme for the flow of EA processes [21]

A combined word embedding could be optimized according to the problem at hand by generating different combinations of word vectors for the problem domain's vocabulary. NLP tasks involve text or speech datasets with a certain vocabulary for the problem's dataset. For example, the OpeNER sentiment classification dataset has a vocabulary of 2393 distinct words. The basic approach of using only one word embedding such as Word2Vec or Glove means that all 2393 words would be assigned either with Word2Vec or Glove vectors only. Works on improved word vectors suggest combining Word2Vec and Glove vectors to produce a combined word embedding, which would certainly be better than using either Word2Vec or Glove alone. However, these combined word vectors are static and therefore, the same set of word vectors would be used for all words in a dataset's vocabulary, regardless of the vocabulary size. This is the major flaw in all existing works that propose to generate improved word embeddings merely by combining two pretrained word embeddings. Even an improved word embedding that is produced by combining two pretrained word embeddings is likely to show only limited improvement if the same set of word vectors is used for all datasets by completely ignoring important aspects such as data dimensionality (vocabulary size). There is significant room for improvement in this issue by applying optimization.

Combinatorial optimization of two pretrained word embeddings is very similar to feature subset selection. A feature subset selection problem with n features is an n-dimensional problem with a finite search space of size 2^n because candidate solutions are represented as bit strings of length n with every bit denoting either the presence or absence of the feature at the corresponding bit's position. Similarly, candidate solutions for distinct combinations of two pretrained word embeddings can also be represented as bit strings of length |V|, where |V| is the vocabulary size of the dataset at hand. For example, the OpeNER sentiment classification dataset's vocabulary size is 2393 and therefore, the bit string's length would be 2393. Bits can be used to generate combinations of two pretrained word embeddings, but integer values will have to be used if there are more than two embeddings involved. Combinatorial optimization of Word2Vec and Glove embeddings can be generated using bit strings because bit 0 can be used to denote a Word2Vec vector and bit 1 can be used to denote a Glove vector or vice versa. Small vocabulary size will be considered for the purpose of discussion. Given a dataset with 10 words (labelled as w_1 to w_{10}) in the vocabulary, candidate solutions can be encoded as 10-bit strings as shown in Fig. 3.

0	0	1	0	1	0	1	0	1	1
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}

Fig. 3: Candidate solution bit string for ECOWE (Genotype)

Assuming that bit 0 is fixed for choosing word vectors from Word2Vec and bit 1 for choosing word vectors from Glove, the first and second bits refer to the choice of word vectors from Word2Vec followed by the third bit referring to the choice of word vector from Glove and so on. Hence, the combined embedding would contain a mix of Word2Vec and Glove vectors for all 10 words in the vocabulary as shown in Fig. 3. This is just one possible combination represented by one candidate solution. A total of $2^{|V|}$ combinations can be generated using exhaustive search methods. Since |V| is different for every dataset, the ECOWE search space would also be different according to the problem's dataset. This reinforces the benefit of optimization in improving word vectors by combining pretrained word embeddings. Instead of having just one static combined word embedding by stacking a Word2Vec embedding with a Glove embedding, combinatorial optimization allows for generating various possible embeddings that are likely to produce different levels of accuracy for different problems. During fitness evaluation, every candidate solution's genotype is converted into its phenotype equivalent to generate the combined word embedding as shown in Fig. 4. The genotype in Fig. 3 is an example of a possible combination of word vectors for a vocabulary of 10 words. Its phenotype would be the combined word embedding shown in Fig. 4, where every bit 0 is replaced by the Word2Vec vector and every bit 1 is replaced by the Glove vector for each word at the corresponding index position in the vocabulary.

Word2Vec vector for w_1	Word2Vec vector for w_2	Glove vector for w_3	Word2Vec vector for w_4	Glove vector for w_5	Word2Vec vector for w_6	Glove vector for w_7	Word2Vec vector for w_8	Glove vector for w_9	Glove vector for w_{10}

Fig. 4: Candidate solution for ECOWE (Phenotype)

ECOWE algorithm is based on GA, similar to most evolutionary feature subset selection algorithms. Fitness of every candidate solution or combination of word embeddings can be evaluated based on the specific problem domain. For example, if the OpeNER sentiment classification dataset is used as input for the ECOWE algorithm, then the algorithm would generate 2393-bit strings to evolve combinations of word embeddings for all words in the dataset's vocabulary. Each candidate solution can be evaluated by training a sentiment classification model using the combined word embedding and its utility or fitness would be represented by its test accuracy. In a GA, this fitness value would guide the search process to evolve towards finding the most optimal combination of Word2Vec and Glove embeddings for the specific dataset's vocabulary. Hence, optimization of pretrained word embeddings is a more economic way to produce new word embeddings than employing deep learning based models to train new embeddings out of a corpus from scratch. Training new word embeddings from scratch is not just computationally very costly and time consuming, but domain specific corpora may have to be considered for different problems. Optimization avoids this issue by optimizing the pretrained word embeddings according to the specific problem's dataset vocabulary to achieve optimal accuracy. Table 1 shows components and parameters of the ECOWE algorithm. Fig. 5 shows the flowchart for the ECOWE algorithm's fitness function.

Table 1: Components and parameters of ECOWE algorithm

Parameter	**Value**
Representation	Bit string
Population Size	1000
Number of Generations	10000
Recombination	Two-point crossover
Crossover Rate	0.95
Mutation	Bit flip
Mutation Rate	0.01
Parent Selection	Tournament selection (tournament size=3)
Survivor Selection	Tournament selection (tournament size=3)
Termination Condition	100000 fitness evaluations

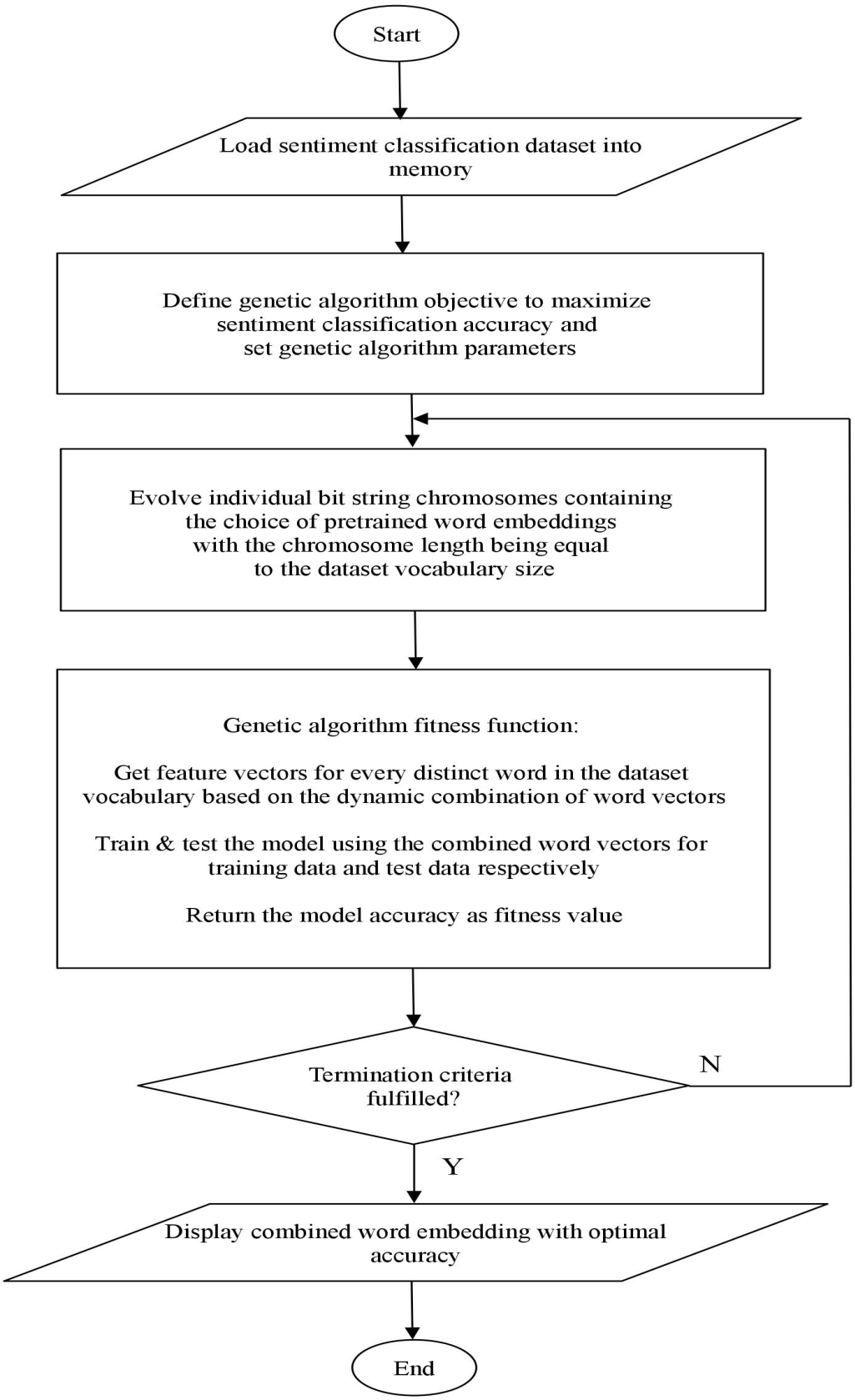

Fig. 5: Flowchart for the ECOWE algorithm's fitness function

4.0 EVALUATION

Six sentiment classification datasets used in [9] namely OpeNER, SemEval, SST-binary, SST-fine, SenTube-A and Sentube-T have been used for evaluation. Details about these datasets are shown in Table 2.

Table 2: Benchmark datasets for sentiment classification

Dataset	Training Set	Development Set	Test Set	Vocabulary Size	Number of Classes
OpeNER [17]	2780	186	743	2393	4
SemEval [18]	6021	890	2376	21169	3
SST-binary [14]	6920	872	1821	17540	2
SST-fine [14]	8544	1101	2210	19501	5
SenTube-A [22]	3381	225	903	10609	2
SenTube-T [22]	4997	333	1334	10904	2

Baseline models are SVMs trained using the original pretrained Word2Vec and Glove embeddings. Results in Table 3 show that Word2Vec appears to be better than Glove for the baseline model of all datasets.

Table 3: Baseline (BL) results

Dataset	Word2Vec	Glove	Mean	Max
OpeNER	76.0	67.0	71.5	76.0
SemEval	66.2	56.8	61.5	66.2
SST-binary	81.8	67.7	74.8	81.8
SST-fine	43.4	33.2	38.3	43.4
SenTube-A	64.3	61.4	62.9	64.3
SenTube-T	66.6	60.9	63.8	66.6

Apart from baselines, benchmark accuracy values have also been taken for the six datasets from the three most related works [9, 15, 16]. The benchmark accuracy values are shown in Table 4.

Table 4: Benchmark (BM) results

Dataset	BM 1 [9]	BM 2 [15]	BM 3 [16]	Mean	Max
OpeNER	82.5	78.1	84.2	81.6	84.2
SemEval	68.5	69.5	72.2	70.1	72.2
SST-binary	83.1	82.3	84.2	83.2	84.2
SST-fine	45.6	43.6	48.1	45.8	48.1
SenTube-A	66.2	66.0	64.1	65.4	66.2
SenTube-T	68.1	68.4	68.8	68.4	68.8

SAN [16] had recorded the highest benchmark accuracy for all datasets, except SenTube-A. Mean accuracy values for the baseline and benchmark models are considered just as rough indications of the common accuracy levels for every dataset. However, since the maximum accuracy values would certainly be higher than the mean accuracy values, only the maximum accuracy values are considered for benchmarking comparisons and evaluation of the ECOWE algorithm.

As shown in Table 5, benchmark accuracy values are higher than the baseline accuracy values for all datasets.

Table 5: Comparison between the best baseline and benchmark results

Dataset	MAX BL	MAX BM	MAX
OpeNER	76.0	84.2	84.2
SemEval	66.2	72.2	72.2
SST-binary	81.8	84.2	84.2
SST-fine	43.4	48.1	48.1
SenTube-A	64.3	66.2	66.2
SenTube-T	66.6	68.8	68.8

Table 6 shows results of the ECOWE algorithm's evaluation using the six sentiment classification datasets. ECOWE is based on GA and GA is a stochastic algorithm, i.e. the algorithm operates on the basis of randomness and it would produce slightly different results when the program execution is repeated using the same inputs. Therefore, five runs of the ECOWE algorithm were conducted for every dataset to get the mean accuracy as a more reliable estimate.

Table 6: Results for ECOWE algorithm with five runs on each dataset

Dataset	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Standard Deviation	Max
OpeNER	88.0	88.8	87.3	88.3	87.5	88.0	0.6	88.8
SemEval	73.9	74.7	73.5	74.4	73.4	73.9	0.6	74.7
SST-binary	87.9	86.1	86.9	87.5	87.3	87.1	0.7	87.9
SST-fine	52.0	50.3	50.3	51.0	51.8	51.1	0.8	52.0
SenTube-A	79.8	79.2	78.7	78.1	79.8	79.1	0.8	79.8
SenTube-T	76.8	78.6	77.6	77.1	77.1	77.4	0.7	78.6

Table 7 shows the comparison between ECOWE mean accuracy values against the benchmark accuracy values. Average of absolute percentages of improvement is 5.5% and average relative percentages of improvement is 8.1%. The ECOWE mean accuracy values are statistically significant compared to benchmark accuracy values with a z-score of -2.2014 using two-tailed Wilcoxon signed rank test with 5% significance level.

Table 7: Comparison of ECOWE mean accuracy values against benchmark accuracy values

Dataset	ECOWE Mean Accuracy	Benchmark Accuracy	Absolute Percentage of Improvement (%)	Relative Percentage of Improvement (%)
OpeNER	88.0	84.2	3.8	4.5
SemEval	73.9	72.2	1.7	2.4
SST-binary	87.1	84.2	2.9	3.4
SST-fine	51.1	48.1	3	6.2
SenTube-A	79.1	66.2	12.9	19.5
SenTube-T	77.4	68.8	8.6	12.5

5.0 DISCUSSION

A first glance at the baseline model accuracy values show that pretrained word embeddings have yielded much lower accuracy values compared to benchmark values from related works and the optimal ECOWE accuracy values, thus proving the inadequacy of pretrained word embeddings when used in isolation for extrinsic evaluation or downstream NLP tasks like sentiment classification. This is despite the fact that these embeddings were found to perform well for intrinsic evaluation tasks like word similarity and analogical reasoning of words. The inadequacy of pretrained word embeddings when used in isolation reaffirms and justifies the need for focusing on creating improved word vectors via combination of embeddings. Accuracy values of baseline models clearly show that the Word2Vec embedding is a much more high quality embedding compared to Glove for all the sentiment classification datasets. The huge gap in accuracy between both baseline models suggests that the Glove embedding is of very poor quality compared to Word2Vec and therefore, even the mean accuracy values of both baseline models appear to be quite low. However, the ECOWE algorithm had clearly outperformed both the Word2Vec and Glove baseline models for all the datasets. Interestingly, the optimal ECOWE accuracy values are way much higher than the mean accuracy values of the baseline models. This shows that the combined embeddings of ECOWE are not too simplistic and the ECOWE word vectors do not merely create an average of the Word2Vec and Glove embeddings.

Since the ECOWE algorithm conducts combinatorial optimization, the share of Word2Vec and Glove word vectors in the combined word embedding may not necessarily be equal to a 50:50 ratio, i.e. given a vocabulary of 100 words, the combined word embedding would not always contain 50 Word2Vec vectors and 50 Glove vectors because bit strings which represent candidate solutions in a genetic algorithm are generated randomly. Therefore, not every bit string would contain an equal number of 0 and 1 bits. This explains why the ECOWE accuracy values are not similar to the mean values of the baseline models' accuracy. Another interesting observation is that the gap between the mean accuracy values of baseline models and the ECOWE optimal accuracy values is also huge, with ECOWE accuracies being much higher than the mean accuracies of the baseline models. This corroborates the true contribution of the ECOWE algorithm by moving beyond simplistic operations for combining word embeddings like stacking two word embeddings or retrofitting the embeddings with lexicon data. The combinatorial optimization process searches through a very large search space where there are many possible combinations of word vectors, though every word by itself could only take either the Word2Vec or Glove word vector. Diverse combinations of word vectors obtained from the huge space of combined embeddings on a continous vector space creates many possible representations (candidate solutions) for embedding text inputs from the sentiment classification datasets. Hence, the evolutionary search process of the ECOWE algorithm could search for optimal embeddings that provided high quality representations for raw text inputs in every sentiment classification dataset, as intended.

Despite the fact that great advances in deep learning research had led to the creation of pretrained word embeddings like Word2Vec and Glove, which are being hailed as manifestations of the power of text embedding, benchmark results from related works also pale in comparison to the optimal ECOWE accuracy values. This is another interesting observation because some of the models, which had produced these benchmark results have used state-of-the-art deep learning models such as LSTM and BiLSTM, while the ECOWE algorithm had merely used a linear SVM classifier. Hence, the results from this work show that text representation or feature vectors obtained by optimizing pretrained word embeddings can make a crucial difference even when a shallow machine learning algorithm such as SVM is used to train the model. In short, the ECOWE algorithm had achieved its purpose as evident from the significant leaps gained in terms of accuracy compared to the baselines and benchmarks.

6.0 CONCLUSION

In this paper, a genetic algorithm based combinatorial optimization algorithm called ECOWE had been proposed to produce optimal combinations of pretrained word embeddings such as Word2Vec and Glove. The algorithm had been evaluated using six benchmark sentiment classification datasets and results clearly indicate that the ECOWE algorithm had yielded significant improvements compared to the baselines and benchmarks.

Future work would be focused on further optimization of the optimal ECOWE word embeddings to obtain more improved accuracy values than those reported in this work. This is indeed a promising research direction to be pursued given the fact that further optimization of word embeddings on a continuous vector space involves an infinite search space with unlimited possibilities unlike the finite search space of the ECOWE algorithm.

ACKNOWLEDGMENT

Authors would like to thank the reviewers for their careful review of this paper and insightful feedback.

REFERENCES

[1] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis", *Foundations and Trends in Information Retrieval*, Vol. 2 No. 1–2, 2008, pp. 1-135.

[2] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: Tasks, approaches and applications", *Knowledge-Based Systems*, Vol. 89, 2015, pp. 14-46.

[3] D. Maynard and A. Funk, "Automatic Detection of Political Opinions in Tweets", in *Extended Semantic Web Conference*, Springer, Berlin, Heidelberg, 2011, pp. 88-99.

[4] W. Medhat, A. Hassan and H. Korashy, "Sentiment analysis algorithms and applications: A survey", *Ain Shams Engineering Journal*, Vol. 5 No. 4, 2014, pp. 1093-1113.

[5] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", *Nature*, Vol. 521, 2015, pp. 436-444.

[6] A.L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng and C. Potts, "Learning Word Vectors for Sentiment Analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies,* Vol. 1, Association for Computational Linguistics, 2011, pp. 142-150.

[7] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space", in *Proceedings Of Workshop At ICLR*, 2013.

[8] S.M. Rezaeinia, A. Ghodsi and R. Rahmani, "Improving the Accuracy of Pre-trained Word Embeddings for Sentiment Analysis", *CoRR, vol. Abs/1711.08609*, 2017.

[9] J. Barnes, R. Klinger and S. S. I. Walde, "Assessing State-of-the-Art Sentiment Models on State-of-the-Art Sentiment Datasets", in *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Copenhagen, Denmark, Association for Computational Linguistics, September 2017, pp. 2–12.

[10] K. S. Tai, R. Socher and C. D. Manning, "Improved Semantic Representations from Tree-Structured Long Short-Term Memory Networks", *CoRR, vol. Abs/1503.00075*, 2015.

[11] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu and M. Zhou, "Sentiment Embeddings with Applications to Sentiment Analysis", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28 No. 2, 2015, pp. 496-509.

[12] B. Pang and L. Lee, "Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with respect to Rating Scales", in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, June 2005, pp. 115-124.

[13] M. Hu and B. Liu, "Mining and Summarizing Customer Reviews", in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, August 2004, pp. 168-177.

[14] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng and C. Potts, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank", in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1631-1642.

[15] P. Xu, A. Madotto, C. S. Wu, J. H. Park and P. Fung, "Emo2Vec: Learning Generalized Emotion Representation by Multi-task Training" in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Association for Computational Linguistics, 2018, pp. 292–298.

[16] A. Ambartsoumian and F. Popowich, "Self-Attention: A Better Building Block for Sentiment Analysis Neural Network Classifiers", in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Association for Computational Linguistics, 2018, pp. 130–139.

[17] R. Agerri, M. Cuadros, S. Gaines and G. Rigau, "OpeNER: Open Polarity Enhanced Named Entity Recognition", in *Procesamiento del Lenguaje Natural*, (51), 2013.

[18] T. Wilson, Z. Kozareva, P. Nakov, S. Rosenthal, V. Stoyanov and A. Ritter, "SemEval-2013 Task 2: Sentiment Analysis in Twitter", in *Proceedings of the International Workshop on Semantic Evaluation*, SemEval, Vol. 13, June 2013.

[19] C. E. Rasmussen and Z. Ghahramani, "Occam's Razor", in *Advances in Neural Information Processing Systems*, 2001, pp. 294-300.

[20] B. Xue, M. Zhang, W. N. Browne and X. Yao, "A Survey on Evolutionary Computation Approaches to Feature Selection", *IEEE Transactions on Evolutionary Computation*, Vol. 20 No. 4, 2015, pp. 606-626.

[21] A. E. Eiben and J. E. Smith, "Introduction to Evolutionary Computing", *Springer, Berlin*, Vol. 53, 2003.

[22] O. Uryupina, B. Plank, A. Severyn, A. Rotondi and A. Moschitti, "SenTube: A Corpus for Sentiment Analysis on YouTube Social Media", in *LREC*, 2014, pp. 4244-4249.