ANAS: Sentence Similarity Calculation Based on Automatic Neural Architecture Search

Dong-Sheng Wang, Cui-Ping Zhao, Qi Wang, Kwame Dwomoh Ofori, Bin Han, Ga-Qiong Liu, Shi Wang, Hua-Yu Wang, Jing Zhang

ANAS： Sentence similarity calculation based on automatic neural architecture search

WANG Dong-Sheng*[a],ZHAO Cui-Ping[a],WANG Qi[a],Ofori[a],HAN Bin[a],LIU Ga-Qiong[a],WANG Shi[b],WANG Hua-Yu[c]， ZHANG Jing[a]

[a](School of Computer Science, Jiangsu university of Science of Technology, Zhenjiang, Jiangsu 212001, China)

[b](Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190 ,China)

[C](School of Army Engineering University, Nanjing, Jiangsu 210007, China)

ABSTRACT

Sentence similarity calculation is one of several research topics in natural language processing. It has been widely used in information retrieval, intelligent question answering and other fields. Traditional machine learning methods generally extract sentence features through manually defined feature templates and then perform similarity calculations. This type of method usually requires more human intervention and constantly has to deal with domain migration problems. Automatically extracting sentence features using certain deep learning algorithms helps to solve domain migration problems. The training data is of the domain is required to complete the extraction process. However, the neural network structure design in the deep learning model usually requires experienced experts to carry out multiple rounds of the tuning design phase. Using an automatic neural architecture search (ANAS) technology deals with all the network structure design problems. This paper proposes a sentence similarity calculation method based on neural architecture search. This method uses a combination of grid search and random search. The method in this paper is tested on the Quora "Question-Pairs" data set and has an accuracy of 81.8%. The experimental results show that the method proposed in this paper can efficiently and automatically learn the network structure of the deep learning model to achieve high accuracy.

Keywords: Natural language processing, Deep learning, Neural architecture search, Sentence similarity calculation

1. INTRODUCTION

Sentence similarity calculation is a branch of artificial intelligence in natural language processing. Traditional machine and deep learning methods have proven to be highly efficient when dealing with problems encountered in sentence similarity calculation. The neural network structure design in a deep learning model usually requires experienced experts to carry out multiple rounds of tuning and design. Adjusting the network structure to adapt to the characteristics of the field then becomes necessary.

At present, research done in automatic searches under neural architecture search have produced promising results. This paper proposes a sentence similarity calculation model based on neural architecture search. The model uses padding and label encoding technology to convert text inputs into a real number vector. It applies the GloVe model to get the word embedding representation next and finally returns a numerical output between 0 and 1. Adam algorithm is then used for intensive training. This algorithm ascertains the similarity between the two sentences according to the numerical results output by the model. The proposed model uses neural architecture search technology to initiate the model generation process and strategy. It then applies the random search method to iteratively find a better model combination. The main purpose of this paper is to design a deep neural network model using an automatic neural architecture search with minimal human intervention. This way, the domain migration ability increases hence the efficiency of the model in general is improved.

2. RELATED WORK

Many researchers have conducted extensive research in sentence similarity calculation. Zhai Sheping [12] fully considered sentence structure and semantic information. He proposed a method for calculating semantic similarity of sentences based on multi-feature fusion of semantic distance. Liu Jiming et al. [13] combined smooth inverse frequency (SIF) with dependent syntax to calculate sentence similarity. Ji Mingyu et al. [14] calculated sentence similarity by reducing and normalizing the difference features between sentence vectors. He Yinggang [15] proposed a sentence similarity calculation method based on word vectors and LSTM... There are also numerous studies done on neural architecture search. Google [16] proposed neural architecture search for convolutional neural networks. Qi Fei et al. fitted a multivariate Gaussian process function to construct an acquisition function for optimized search based on the trained neural network architecture map and the corresponding evaluation value... Although the above method was highly efficient, there are recurring problems such as low accuracy and time-consuming search process. Therefore, this paper proposes an automated neural architecture search technology which greatly improves efficiency.

3. OUR APPROACH

This paper can be divided into two aspects: the first part is the research and realization of automated neural architecture search technology; the second is the implementation of a deep learning model for sentence similarity calculation based on neural architecture search.

3.1 Neural architecture search technology

With the increasing demand of deep learning models, designing a complex network structure requires more time and effort. Therefore, neural architecture search is used to automate the construction of neural networks. This section describes the three key components of ANAS: search space, search strategy and search performance evaluation strategy.

Search space. There are two common search space types in the current research results which are the chain structure neural network space and the unit search space. The NAS search architecture used in this article refers to the idea of unit search space. By pre-defining some neural network structural units and then using NAS to search the stacking mode of these network units to build a complete neural network. The basic network block is created as the main search object and ultimately generates good models.

Search strategy. Although the search space is theoretically limited, it requires a certain search strategy to balance exploration and utilization: on one hand, it is necessary to search for a good-performance architecture rapidly, on the other hand, it should be avoided due to the repercussion of early convergence.

Many different search strategies like Bayesian optimization and evolutionary algorithms are used to explore the neural architecture space. Restricted by experimental conditions, this paper uses a search strategy that combines grid search and random search commonly used in machine learning. Firstly, we use grid search on the framework parameter and then perform random search on other parameters. Next, we combine them into a complete model for training and evaluation. Finally, we keep the model parameters with the highest accuracy on the test set.

The ANAS search strategy algorithm used in this paper is as follows:

Algorithm ANAS Strategy Based on Random Search of Network Blocks algorithm

Input: training data set, search space
Output: good model in search space
Begin
Define search space, including 1earning rate space, batch size space, framework space, dense space, dropout space, random space.
(2)For i=1 to 1ength(framework space) do
(3) For j=1 to 1ength(random space) do
(3) Get model block parameters in search space randomly.
(4) Generate a complete model by model block parameters in (3).
(5) Training model in data set, get the result
(6) If the results are better now than last time.
then record and cover model block parameters.
Else
continue.
(7) End;

(8) End;
(9) Get good model parameters.
(10)End.
End.

The input of the algorithm is the training set and the search space. The output is a set of network parameters. Using this set of parameters, a complete model can be generated. The model performance outperforms other models with same functions.

In the above analysis, the data set used is a subset of the target data set. The specific determination of the search space depends on the outcome of search space experiments carried out. The search space parameters used in this paper are shown in Table 1 below:

Table 1. Margins and print area specifications.

ANAS parameters	Value range
Learning rate space	（0.0001, 0.1）
Batch size space	（50, 10000）
Framework space	（0, 20）
Dense space	（10, 500）
Dropout space	（0, 0.5）
Random search times	100

The parameters in Table 1 are mainly defined based on the experience of designing deep learning models in the past. The upper limit of Random search times is the number of all permutations and combinations in the search space, but subject to the computing power of the experimental platform. We set this value to 100 and eventually generate 2000 models. The model was trained and tested, and it took more than 60 consecutive hours to get the final result.

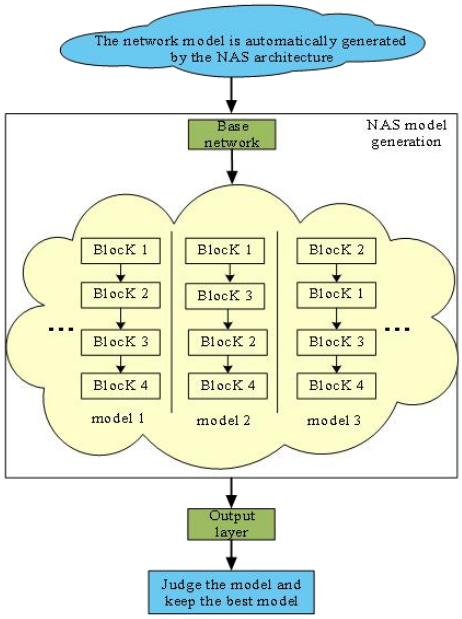

Figure 1. ANAS implementation diagram

The neural architecture search technology proposed in this paper is based on the backbone network. It is also based on the network block stacking and internal parameter changes. It randomly searches for network combinations that perform well. The ANAS implementation is shown in Figure 1.

Performance Evaluation Strategy. The goal of ANAS is to find a model architecture that achieves high predictive performance on unknown data, Therefore, effective performance evaluation strategies need to be developed. Considering

the computational cost, only standard training and verification of 2000 generated models were performed. Under the premise of GPU acceleration, it took more than 60 hours. In the subsequent research, measures need to be taken to improve the efficiency of ANAS search.

3.2 Model implementation and evaluation optimization

Model construction. This paper uses TensorFlow and Keras to implement the construction, training, and prediction of deep learning models. The deep learning model structure used in this paper is shown in Figure 2:

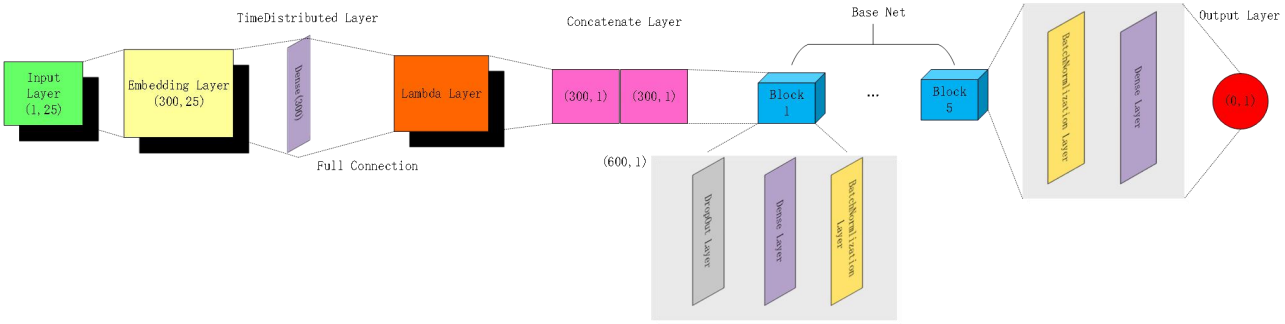

Figure 2. Discriminant model neural network architecture

The input sentence is vectorized using the Glove word embedding of feature engineering and transformed into a real number matrix of shape (1, 25), which respectively represents two sentences to be differentiated. After the Embedding layer is processed, the matrix becomes (300, 25) and each vocabulary is transformed into a word embedding vector with a length of 300 dimensions. After calculation, a result of (1, 1) size is output and the value ranges between 0~1. The next task is to determine if the intent of the two sentences are the same according to the magnitude of the value. The model is implemented using TensorFlow-based Keras. Embedding is the advanced neural network layer that comes with Keras. The input layer serves as the initial occupancy layer of the Keras network and is used to agree on the matrix shape of the input data. The Embedding layer implements the mapping of data to the word embedding vector. The weight of the word embedding vector is set through the weight's parameter. The trainable parameter is usually set to False so that the training can proceed normally. The TimeDistributed layer is an advanced encapsulator. The application in this model is equivalent to independently applying a fully connected layer with 300 neurons on each vocabulary output by the Embedding layer. The Lambda layer is used to encapsulate any expression into the Layer object of the network model. In this model, the Lambda layer selects the word embedding vector with the largest activation value of each word in the output of the TimeDistributed layer. This is done to improve the nonlinear learning ability of the model. The Concatenate layer is used to connect two matrices together. In this model, the (300, 1) matrix output by the two Lambda layers is connected into a (600, 1) matrix. The Dropout layer receives dropout parameters between 0 and 1 and randomly discards the output results of the previous layer according to the parameter size. This process only takes effect during network training which then reduces the overfitting of the model. The dense layer is a fully connected layer. After setting the number of neurons, Keras will automatically establish weight connections between all neurons in this layer and all outputs of the previous layer. BatchNormlization will normalize the calculation data of the current batch which reduces the over-fitting of the model.

Model training. The Adam algorithm is used to train the model with the help of the fit method of the Keras framework. Using Adam algorithm can speed up the training speed, making the model converge in less epoch training. TensorFlow and Keras are used directly in the training, "*model.fit([x1, x2], y, epochs=50, validation_split=0.1,batch_size=4096)*" can be executed to achieve model training.

Part of the information during the model training process is shown in Table 2 below:

Table 2. Part of the information during model training

epoch	time cost	loss	acc	val_loss	val_acc
1/50	6s 18us	0.6028	0.6781	0.5384	0.7247
2/50	4s 11us	0.5047	0.7484	0.5259	0.7264

3/50	4s 11us	0.4644	0.7728	0.4968	0.7537
4/50	4s 11us	0.4332	0.7913	0.4567	0.7797
5/50	4s 11us	0.4042	0.8080	0.4461	0.7877
6/50	4s 11us	0.3810	0.8219	0.4299	0.7937

In Table 2, val_loss and val_acc respectively represent the loss function value and accuracy of the current training batch of the model in the validation set.

Model evaluation and optimization. There are many ways to optimize models. This article only considers optimizing the accuracy of the model.

Underfitting. The word embedding of the GloVe model and TimeDistribute layer are used to improve the network structure. After embedding is output, a fully connected layer of the same 300 dimensions is applied to the representation of each word to make up for the insufficiency of the embedding layer trainable=False setting and improve the accuracy of the model. In order to deal with under-fitting, the neural network structure is deepened and the basic network. After the block, the neural architecture search mechanism determines how deep the network needs to be stacked.

Overfitting. By reducing the number of network parameters or the Dropout method, the Dropout layer of Keras receives the parameter rate and its value should be set between 0 and 1. This indicates the proportion of discarding. Use BatchNormlization layer to reduce model overfitting and speed up training.

4. EXPERIMENT

4.1 Datasets

The training set data uses a data set published by Quora in the form of a competition on the Kaggle platform with a size of about 55MB and a total of about 400,000 training samples. The format of the original data set is a text format with tabs separating different columns and line breaks separating different rows and each row represents a labeled sample.

Table 3. Example of training data

index	question1	question2	is_duplicate
1	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
2	What does manipulation mean?	What does manipulation mean?	1
3	Why are so many Quora users posting questions ...	Why do people ask Quora questions which can be...	1

As shown in Table 3, mark 0 means that the intent of question 1 and question 2 are inconsistent, and mark 1 means that the schematic diagram is consistent.

To train and evaluate the model, the labeled dataset is divided into three subsets: training set, validation set and test set. There is no strict basis for the division ratio of the data set. In this work, we divide the data set into 3 parts. The test set (10%) , the validation set (5%) and the training set (85%).

4.2 Evaluation index

The focus of this paper is to calculate the accuracy of the model but also to consider other evaluation indicators.

Model accuracy. The research focuses on the accuracy of the model. The following is the accuracy calculation formula:

$$A=\frac{R}{S}$$

A is the accuracy rate, R is the number of samples the model predicts correctly in the data set and S is the number of all samples in the data set.

The following figure shows the changes in accuracy on training set and validation set during the training process of the model:

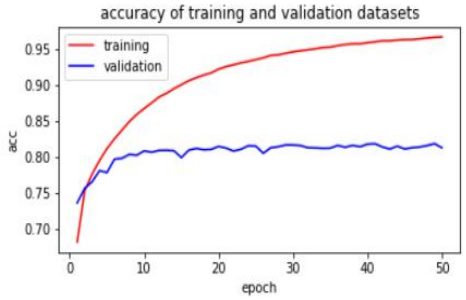

Figure 3. Changes in the accuracy of the model in the training set

Table 4 shows the accuracy results of the three data sets after the model training is completed.

Table 4. Accuracy result after model training is completed

	Training set	Validation set	Test set
Accuracy	0.9640	0.8185	0.8180

Training time and prediction time-consuming. Model training time-consuming is mainly related to model structure, optimization algorithm, experimental software and hardware environment. GPU acceleration is used in the experimental environment of this paper. Table5 shows whether GPU acceleration is used to train the model influences:

Table 5. The impact of GPU acceleration on training time

	Use GPU	Without GPU
1 training batch time	4.04s	54.70s
Total training time	202s	2735s
rate	1	13.54

Table 5 shows that in the experimental environment of this article, keeping other configurations unchanged and using GPU when training the model can be 13.54 times faster than not using GPU. In actual application, how long the model needs to be used to obtain the prediction result is also a factor that needs to be considered. This is slightly different from the factors that affect the time-consuming model training. It is related to the structure of the model and the experimental software and hardware environment, but is related to the use The optimization algorithm has nothing to do.

Table 6. The impact of GPU on model prediction time

	Use GPU	Without GPU
Time-consuming prediction of 1 sample	5ms	17ms
Time-consuming prediction of 10 samples	5ms	17ms
Time-consuming prediction of 100 samples	11ms	39ms

It is reflected from Table 6 that the prediction time is very short, in the order of milliseconds. However, the time consumption of predicting 1 sample and predicting 10 samples or even predicting 50 samples has not changed much in the two cases of using GPU acceleration and not using GPU acceleration. The reason is the design of the model in this article and the characteristics of TensorFlow. It is decided that the model can calculate multiple samples at the same time, which is the result of automatically calling multi-threaded calculations behind TensorFlow. Therefore, within a certain range of predictions, the prediction results can be quickly obtained without being affected by the number of samples.

4.3 Experimental results and analysis

This chapter compares and analyzes the training results of the deep learning model proposed mainly considering the accuracy and time-consuming aspects of the model. The models participating in the comparison include random forest algorithm, gradient boosting tree algorithm (GBDT), and support vector machine (SVM) model.

Model accuracy. Table 7 shows the accuracy of each model.

Table 7. Accuracy performance of each model

	Acc. on the training set	Acc. on the test set
Random forest	0.9459	0.7450
GBDT	0.9843	0.7706
SVM	0.7875	0.6799
ANAS	0.9640	0.8180

The ANAS in Table 7 refers to the model used in this paper. The results in the table cannot directly judge the pros and cons of different model algorithms. Because this paper does not perform feature engineering on the data suitable for the specific algorithm, the data results does not reflect the best performance of the model on this data set.

Comparison of prediction time-consuming models. Table 8 shows the prediction time-consuming results of each model.

Table 8. Forecast time-consuming results of each model

	Prediction 1 sample	Prediction 10 samples	Prediction 100 samples
Random forest	110ms	115ms	270ms
GBDT	3ms	3ms	9ms
SVM	95ms	98ms	250ms
ANAS	5ms	5ms	11ms

As shown in Table 8, the prediction time is greatly affected by the implementation framework. This paper uses the random forest algorithm and support vector machine implemented by Sklearn. It also uses the LightGBM provided by Microsoft to implement the gradient boosting tree algorithm. A deep learning model implemented by TensorFlow and Keras is also used. Sklearn is mainly used in the data preprocessing stage and did not do calculation optimization work. Microsoft's LightGBM and Google's TensorFlow are commercial open source applications that have made a lot of optimizations on computing performance. Therefore, the time-consuming is significantly better than the model implemented by Sklearn.

5. CONCLUSION

This paper mainly focuses on the problem of sentence similarity calculation. In order to use the deep learning method, the deep learning model and neural architecture search are discussed and studied. Feature engineering chooses GloVe word embedding and this choice is also confirmed by the results. The research of neural architecture search mainly realizes the search based on the unit search space, creating the backbone network and the basic network block. We combine grid search and random search strategies using network generators and neural architecture search engines. Finally, a relatively good deep learning model is formed. The key points of the deep learning model are evaluated and analyzed. The basic structure and function principle and optimization direction of the deep learning model are described and summarized. The model obtained in this paper scores an accuracy of 81.8% on the test set. The results indicate that using the ensemble learning techniques commonly used in machine learning competitions, combined with multiple model methods should be further improved.

ACKNOWLEDGEMENT

This paper is funded by the National Natural Science Foundation of China "Research on Automatic Learning Method of Constrained Semantic Grammar for Short Text Understanding" (61702234).

REFERENCES

[1] Gabor Angeli and Christopher D Manning. Naturalli: Natural logic inference for common sense reasoning. In EMNLP, 2014.
[2] Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (2017)
[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In ICLR, 2015.
[4] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In Machine Learning Challenges. Lecture Notes in Computer Science, volume 3944, pages 177–190. Springer,2006.
[5] Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: AutoAugment: Learning Augmentation Policies from Data. In: arXiv:1805.09501 (May 2018)
[6] Karl Moritz Hermann, Toma´sKocisky´, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In NIPS, 2015.
[7] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In NIPS, 2015.
[8] Ahmed,K.,Torresani, L.: Maskconnect: Connectivity learning by gradient descent. In: European Conference on Computer Vision (ECCV) (2018)
[9] Huang, G., Liu, Z., Weinberger, K.Q.: Densely Connected Convolutional Networks. In: Conference on Computer Vision and Pattern Recognition (2017)
[10] Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions (2016)
[11] Chen, T., Goodfellow, I.J., Shlens, J.: Net2net: Accelerating learning via knowledge transfer. In: International Conference on Learning Representations (2016)
[12] ZHAI She-ping, LI Zhao-zhao, DUAN Hong-yu, et al. Sentence semantic similarity method based on multifeatured fusion [J]. Computer Engineering and Design,2019. (In Chinese)
[13] LIU Ji-ming, TAN Yun-dan, YUAN Ye. Calculation Method of Sentence Similarity Based on Smooth Inverse Frequency and Dependency Parsing. Chongqing University of Posts and Telecommunications, 2019. (In Chinese)
[14] JI Mingyu, WANG Chenlong, AN Xiang, et al. Method of sentence similarity calculation for intelligent customer service.Computer Engineering and Applications, 2019, 55（13）：123-128. (In Chinese)
[15] HE Ying-gang, WANG Yu. Method of Sentence Similarity Calculation for Intelligent Customer Service [J]. Journal of Yangtze University (Self Science Edition),2019. (In Chinese)
[16] Google. Neural architecture search for convolutional neural networks:CN201880022762.8[P].2019-11-19. (In Chinese)