SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2021, Vol. 64 212105:1–212105:23
https://doi.org/10.1007/s11432-019-3018-6

State and tendency: an empirical study of deep learning question&answer topics on Stack Overflow

Henghui ZHAO[1], Yanhui LI[1*], Fanwei LIU[1], Xiaoyuan XIE[2] & Lin CHEN[1]

[1]*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093*, China;*
[2]*School of Computer Science, Wuhan University, Wuhan* 430072*, China*

Received 6 December 2019/Revised 18 March 2020/Accepted 7 May 2020/Published online 15 October 2021

Abstract Deep learning has developed rapidly in recent years, attracting the attention of numerous researchers. Since a wide range of topics are covered in this field, we are wondering what topics researchers have concerned about. However, after investigation, we find that very few researchers have paid attention to this demand. In this paper, we conduct a large-scale study to analyze the questions faced by deep learning developers. We use Stack Overflow, one of the largest question&answer sites, as our data source, and extract 32969 posts about deep learning as our studied dataset. After filtering, augmenting and pre-processing the post datasets from Stack Overflow, we use the latent Dirichlet allocation (LDA) topic model to summarize 30 topics based on their text content. In addition, we measure the difficulty and popularity of each topic, compare the different issues faced by different deep learning frameworks, and analyze the development trend of each topic. Our main results are as follows: (1) developers ask a broad spectrum of questions about deep-learning, ranging from Data Shape to Object Detection; (2) Gradient Propagation is the most popular among all the topics and (3) Object Detection is the most difficult; (4) issues of Package Installation, Code Understanding and Method Introduction are common in the current different deep learning frameworks; (5) there are three trends in these topics, e.g., a significant rising trend is found in the number of discussion on Data Shape. Finally, based on our research findings, we make some targeted and valuable suggestions for developers, researchers, educators, and framework providers of deep learning.

Keywords topic model, deep learning, Stack Overflow, question&answer

Citation Zhao H H, Li Y H, Liu F W, et al. State and tendency: an empirical study of deep learning question&answer topics on Stack Overflow. Sci China Inf Sci, 2021, 64(11): 212105, https://doi.org/10.1007/s11432-019-3018-6

1 Introduction

In recent years, deep learning has achieved great success in various fields and has brought a lot of influence to many areas [1]. Deep learning has dramatically improved the state-of-the-art in speech recognition [2], visual object recognition [3], object detection [4], and many other domains such as drug discovery [5] and genomics [6]. At the same time, a large number of deep learning frameworks [7–10][1)] are developed for researchers to make better use of deep learning techniques. However, deep learning involves a complex set of algorithms and methods [11], and some of the current deep learning frameworks are difficult to use [12], which brings challenges to both researchers and developers.

There are some studies that focus on topics in a certain field and explore the popularity and difficulty of the field, such as mobile [13], security [14], and concurrency [15]. However, as far as we know, there is no similar research on the so popular field of deep learning. This paper attempts to analyze the topics of question&answer in the field of deep learning through the topic model and further studies five research questions to help people better understand what topics they are facing and the characteristics (e.g., difficulty and popularity) of each topic.

We conduct our study on the dataset extracted from Stack Overflow, which is a large open community for researchers and developers to post questions and share insights with each other. Until August 2019, Stack Overflow had more than 11000000 registered users and more than 18000000 questions[2)]. What's

* Corresponding author (email: yanhuili@nju.edu.cn)

1) Chollet F. Keras. https://keras.io, 2015.

2) https://stackexchange.com/sites?view=list#users.

more, Stack Overflow provides an open-source dataset, Stack Exchange Data Dump[3)], to facilitate scholars to conduct research, which makes Stack Overflow an ideal platform for large-scale experiments. We download the dataset from the Stack Exchange Data Dump and take the following steps, trying to understand what deep learning developers ask. First, we filter out the posts related to deep learning by the tag information. Then, these posts are further screened and pre-processed to obtain the final dataset. After that, based on the text information of the post, the latent Dirichlet allocation (LDA) topic model [16] is used to divide the post into several topics. Finally, we conduct further analysis of the topics and try to answer the following research questions (RQs).

• RQ1: What deep learning topics do developers ask? Deep learning covers a wide range of topics, and we are interested that which topics developers are discussing on Stack Overflow. Through model training and manual review, we find that developers ask a broad spectrum of questions about deep learning, including 30 topics, such as Data Shape, Loss Calculation, Convolution, and Package Installation.

• RQ2: What topics are more popular than others? Different topics are different in popularity. Knowing that which topics are more popular can help researchers and developers be aware of the current research interests in the deep learning area and keep pace with the development. We find that Package Installation, Gradient Propagation, and Code Understanding are more popular than others, while Object Detection and Batch Size seem to gain less attention.

• RQ3: What topics are more difficult than others? Similar to the previous question, we evaluate the difficulty of each topic to give researchers and developers some insights about what topic is suitable for them and what topic may need more effort. We find that Object Detection is more difficult than others and Data Format seems to be relatively easy to handle.

• RQ4: What are the differences between the issues faced by different deep learning frameworks? There are a variety of excellent deep learning frameworks available for developers to use. We are trying to make a summary of the differences of five currently popular frameworks, including tensorflow, torch, caffe, theano, and keras. On one hand, we find that issues of Package Installation, Code Understanding, and Method Introduction are common in all the five frameworks. On the other hand, issues of API Usage are more common in tensorflow, the same as Code Understanding in torch, Model Implementing in keras, Neural Network in caffe, and Debug in theano.

• RQ5: What is the trend of each topic? We analyze the trends of the topics to help people make reasonable research plans and avoid risks. Some of our findings are that Data Shape, Model Save&Load, Model Implementing, Cloud Computing, and Object Detection have significant rising trends in recent three years, while Gradient Propagation, Method Introduction, and Neural Network have a significant falling trend in these years.

The major contributions of this paper are as follows.

(1) We conduct a large-scale analysis of the deep-learning related issues on Stack Overflow and summarize 30 topics.

(2) We put forward five interesting and valuable research questions, and have an in-depth exploration of the field of deep learning, including the popularity and difficulty of each topic, comparison of different deep learning frameworks, and the development trends of different topics.

(3) We have a discussion of the results of our research and make some targeted suggestions for developers, researchers, educators, and framework providers of deep learning.

The remainder of this paper is organized as follows. In Section 2, we introduce our approach, including two parts, the data processing part and the LDA model analysis part. Section 3 introduces the motivations, approaches, and results of five research questions. In Section 4, we discuss some of the details of our research and provide some targeted suggestions to the developers, researchers, educators, and framework providers of deep learning. Section 5 analyzes the threats that affect the validity of our results. Section 6 introduces some of the related work, and the last section summarizes our work.

2 Approach

The entire framework of our approach is shown in Figure 1. First, we collect the data from Stack Overflow. Then, a series of data processing steps are implemented, including data filtering, data augmentation, and data pre-processing. Finally, the LDA model is used for topic analysis. We introduce our approach in detail through two parts, the data processing part and the LDA modeling part.

3) https://archive.org/details/stackexchange.

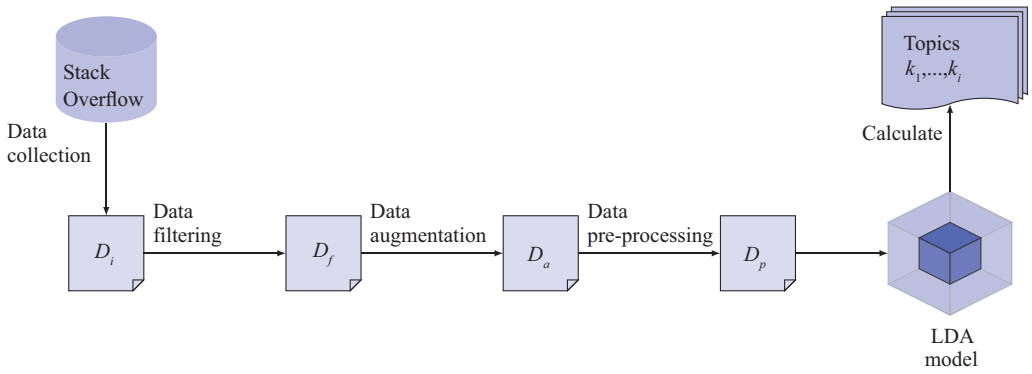

Figure 1 (Color online) The framework of our approach.

Table 1 The detailed structure of the post

Field	Example
Id*	5311671
PostTypeId*	1
AcceptedAnswerId	5312464
CreationDate*	2011-03-15T12:25:39.717
Score*	5
ViewCount*	7197
Body*	What are the best algorithms available to find longest repeating patterns of characters in a string using .net? \n
OwnerUserId*	590278
LastEditorDisplayName	
LastEditorUserId	483620
LastEditDate	2011-03-15T13:43:38.167
LastActivityDate*	2011-03-28T13:44:46.320
Title*	Best algorithm to find a repeating pattern
Tags*	<.net><algorithm><neural-network>
AnswerCount*	2
CommentCount*	5
FavoriteCount	1

* Fields that must be included.

2.1 Data filtering, augmentation and pre-processing

In this subsection, we will detailedly describe how we process the raw data from Stack Overflow. We decide to use the Stack Exchange Data Dump[4)] as our data source, which is an anonymized dump of all user-contributed content on the Stack Exchange network. We download the file 'Post.xml' with 43872992 posts (collected from July 2008 to March 2019) from it as our initial dataset D_i, including 17278709 (about 39.4%) question posts and 26594283 (about 60.6%) answer posts. The detailed structure of question posts is shown in Table 1, where the first column is the field of posts, the second column is an example of the corresponding part of posts, and the fields marked with * are required for all posts.

2.1.1 *Data filtering*

This initial dataset D_i contains the posts from various areas, but here we only need to pay attention to those related to deep learning. Intuitively, we can use the post tags to filter out what we need. The tags can be extracted from the 'Tags' field in the XML file as described in Table 1. Specifically, we first initialize an empty set D_f, and then for each question post, we check its tagset through the field 'Tags'. If the tagset contains a 'deep-learning' tag, we add this post to D_f. In this way, we can get our filtered dataset D_f, which contains 10918 questions.

4) https://archive.org/details/stackexchange.

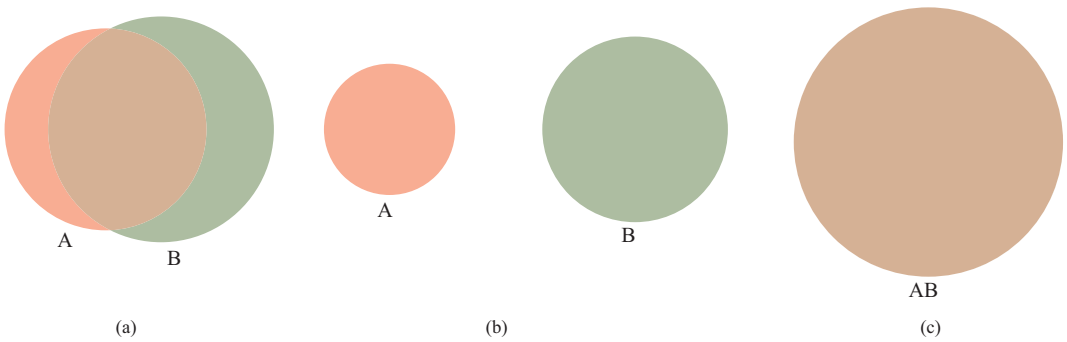

Figure 2 (Color online) Three examples of different m_{t_1,t_2} and n_{t_1,t_2}. A indicates the set of posts tagged with t_1, B indicates the set of posts tagged with t_2. (a) $m_{t_1,t_2} = 0.75, n_{t_1,t_2} = 0.6$; (b) $m_{t_1,t_2} = n_{t_1,t_2} = 0$; (c) $m_{t_1,t_2} = n_{t_1,t_2} = 1$.

2.1.2 *Data augmentation*

However, the number of posts in D_f is not complete, because there are some posts also related to deep learning, but they do not have a 'deep-learning' tag, which reminds us that we can expand our dataset by finding more tags similar to 'deep-learning'.

In order to measure the similarity between two tags t_1 and t_2, we define two indicators, the percentage of overlap in t_1 (m_{t_1,t_2}) and the percentage of overlap in t_2 (n_{t_1,t_2}). The formulas of m_{t_1,t_2} and n_{t_1,t_2} are shown below:

$$m_{t_1,t_2} = \frac{\text{number of posts tagged with } t_1 \text{ and } t_2}{\text{number of posts tagged with } t_1},$$

$$n_{t_1,t_2} = \frac{\text{number of posts tagged with } t_1 \text{ and } t_2}{\text{number of posts tagged with } t_2}.$$

To illustrate the meanings of the two indicators more clearly, we draw a Venn diagram in Figure 2 with different values of m_{t_1,t_2} and n_{t_1,t_2}. As is shown in Figure 2, when $m_{t_1,t_2} = n_{t_1,t_2} = 0$, it means that there is no intersection between posts tagged with t_1 and posts tagged with t_2. When $m_{t_1,t_2} = n_{t_1,t_2} = 1$, it means that once a post is tagged with t_1, it will also be tagged with t_2. In conclusion, m_{t_1,t_2} and n_{t_1,t_2} reflect the correlation between tags t_1 and t_2. The greater m_{t_1,t_2} and m_{t_1,t_2} are, the closer tags t_1 and t_2 are correlated.

Once we are able to calculate the similarity of the two tags, we can get a tagset $\mathcal{T}$ through the steps described in Algorithm 1. After we get $\mathcal{T}$, we find all the posts that contain any tags in $\mathcal{T}$ to form our augmented dataset D_a. The details of the algorithm are as follows. First, we take D_f (we have obtained in Subsection 2.1.1) as input, set two thresholds thre_1, thre_2 and initialize a tagset $\mathcal{T}$ with only one tag 'deep-learning'. Second, for each tag t that have appeared in D_f, we calculate $m_{t,\text{deep-learning}}$ and $n_{t,\text{deep-learning}}$. When $m_{t,\text{deep-learning}}$ is larger than thre_1 and $n_{t,\text{deep-learning}}$ is larger than thre_2, we add t to tagset $\mathcal{T}$.

Algorithm 1 Finding similar tags to 'deep learning'

Input: $D_f \neq \emptyset, 0 \leqslant \text{thre}_1 \leqslant 1, 0 \leqslant \text{thre}_2 \leqslant 1$;
Output: $\mathcal{T}$;
1: $\mathcal{T} \Leftarrow$ {'deep-learning'};
2: **for all** $t \in$ the tagset of D_f **do**
3: calculate $m_{t,\text{deep-learning}}$ and $n_{t,\text{deep-learning}}$;
4: **if** $m_{t,\text{deep-learning}} \geqslant \text{thre}_1 \wedge n_{t,\text{deep-learning}} \geqslant \text{thre}_2$ **then**
5: add t to $\mathcal{T}$;
6: **end if**
7: **end for**

The setting of two thresholds thre_1 and thre_2 is a little tricky. We want tags in $\mathcal{T}$ to be as similar to 'deep-learning' as possible; but at the same time, we also want $\mathcal{T}$ to be as large as possible to extract more related posts. In some prior studies [14, 15], a large number of experiments and manual reviews are used to select the appropriate values of these two thresholds, and the final selected thresholds are about 0.1 and 0.01, respectively. Similarly, we conduct the experiment with different choices of thre_1 and thre_2 around 0.1 and 0.01, respectively, and present the experimental results of the tagset $\mathcal{T}$ in Table 2. As can be seen, when $\text{thre}_1 < 0.11$, some tags, such as 'machine-learning', which are too broad, are

Table 2 The results of different thre_1 and thre_2

thre_1	thre_2	Tagset
0.10	0.01	caffe, rnn, lstm, deep-learning, theano, mnist, torch, machine-learning, keras, keras-layer, pytorch, reinforcement-learning, autoencoder, recurrent-neural-network, pycaffe, convolution, conv-neural-network, neural-network, tensorflow
0.10	0.02	neural-network, keras, pytorch, recurrent-neural-network, theano, deep-learning, caffe, machine-learning, tensorflow, pycaffe, lstm, conv-neural-network
0.11	**0.01**	**keras-layer, rnn, reinforcement-learning, conv-neural-network, mnist, lstm, keras, recurrent-neural-network, pycaffe, caffe, convolution, tensorflow, deep-learning, torch, theano, neural-network, pytorch, autoencoder**
0.11	0.02	conv-neural-network, recurrent-neural-network, pycaffe, tensorflow, pytorch, keras, caffe, deep-learning, lstm, neural-network, theano
0.12	0.01	pycaffe, recurrent-neural-network, neural-network, keras-layer, convolution, pytorch, conv-neural-network, reinforcement-learning, theano, lstm, torch, mnist, autoencoder, caffe, keras, rnn, deep-learning
0.12	0.02	caffe, theano, recurrent-neural-network, neural-network, deep-learning, lstm, pycaffe, keras, conv-neural-network, pytorch

included unexpectedly; when $\text{thre}_1 > 0.11$, tags like 'tensorflow' that obviously belong to deep learning are ignored. Therefore, we decide to set thre_1 as 0.11, and we set thre_2 as 0.01 for the similar reasons.

Further more, to ensure the quality of the dataset and prevent the impact of noise, we decide to use post scores for further filtering. The score of a post can be extracted from the field 'Score' as shown in Table 1, which is defined as the number of upvotes subtracting the number of downvotes in Stack Overflow. We believe that only posts with positive scores make sense, so we exclude posts with non-positive scores.

After calculating the tagset $\mathcal{T}$ and introducing the 'Score' information, we use Algorithm 2 to construct the augmented dataset D_a, which totally contains 32969 posts.

Algorithm 2 Constructing augmented dataset D_a

Input: $D_i, D_f \neq \emptyset, \mathcal{T}$;
Output: D_a;
$D_a = \emptyset$;
for all $d \in D_i$ **do**
  calculate the set $\mathcal{T}_d$ of tags in d;
  extract the 'Score' s of d;
  if $\mathcal{T}_d \cap \mathcal{T} \neq \emptyset$ and $s > 0$ **then**
    add d to D_a;
  end if
end for

2.1.3 *Data pre-processing*

So far, our dataset D_a is still composed of a number of posts in XML form. The data pre-processing step is to extract text information from it to form our final dataset. Specific steps are as follows.

(1) Extract the title and body of each post in our dataset D_a by locating the fields 'Title' and 'Body';

(2) Remove the code snippets (which are enclosed in ⟨code⟩ tag) in each post, since the code language is greatly different from natural language, which may have a negative impact on model training;

(3) Remove all HTML tags such as $\langle p \rangle$ and $\langle /p \rangle$, which appear frequently but do not make any sense.

(4) Remove the stop words, numbers, punctuation marks and other non-alphabetic characters and URLs.

(5) Stem each word by Porter stemming algorithm[5)], which is the data processing method widely used in prior studies [14, 15, 17–19] before LDA, and discard the terms that appear less than 10 times.

After above five steps, we complete the data pre-processing and get our processed dataset D_p for the further analysis.

2.2 Topic modeling with latent Dirichlet allocation

LDA is a kind of topic model, which is mainly used to process text collections and has a wide range of applications in natural language processing and data mining [20–24]. The concept 'topic' in LDA specif-

5) https://tartarus.org/martin/PorterStemmer. Stemming may disturb the analysis of some terminology, but it also has many benefits, such as reducing the impact of plural forms and tense changes of words. Moreover, when the topic words are summarized manually later (as presented in Table 3) in this study, most of the terminology changes (e.g., from "reinforcement" to "reinforc") can be recognized.

ically refers to a series of related words and how often they appear under this concept. For convenience, we use the notations of [25] to present our approach.

- Given a fixed vocabulary V, a collection $D_p = \{d_1, \ldots, d_m\}$ of m posts, the number K of topics.
- The j-th word in the i-th post is denoted as w_{ij}, which is one term belonging to the fixed vocabulary.
- A topic β_k ($\beta_k \in \mathbb{R}^{|V|}$) presents a distribution over V.
- Each post $d_i \in D_p$ is denoted as a vector of topic proportions θ_i ($\theta_i \in \mathbb{R}^K$).
- Each word w_{ij} is assumed to associate with a single topic with the topic assignment z_{ij}.

Algorithm 3 illustrates the details of the generative process of LDA. First, a word distribution β_k of the k-th topic is randomly sampled according to the Dirichlet distribution of the parameter η. Second, a topic distribution θ_i of the i-th post is randomly sampled according to the Dirichlet distribution of the parameter α. Then, a topic z_{ij} is assigned to the current word w_j in post d_i according to θ_i. Finally, the word w_{ij} is randomly sampled according to the word frequency distribution $\theta_{z_{ij}}$ of the topic z_{ij}.

Algorithm 3 LDA [26]

Input: A collection $D_p = \{d_1, \ldots, d_m\}$ of m posts, a topic set $T = \{t_1, \ldots, t_K\}$ of K topics, the prior of topic word distribution η, and the prior of document topic distribution α.

for all $t_k \in T$ **do**
  Draw a word distribution $\beta_k \sim$ Dirichlet(η);
end for
for all $d_i \in D$ **do**
  Draw a topic distribution $\theta_i \sim$ Dirichlet(α);
  for all words w_j in the document d_i **do**
    Draw a topic assignment $z_{ij} \sim$ Multinomial(θ_i);
    Draw a word $w_{ij} \sim$ Multinomial($\theta_{z_{ij}}$);
  end for
end for

There are a variety of libraries that have implemented the LDA algorithm, such as scikit-learning [27], which is a free software machine learning library for the Python programming language and widely used in the fields of data mining and data analysis for software engineering [28–32]. Therefore, in this paper, we train our LDA model with scikit-learning. Specifically, we use the LatentDirichletAllocation[6)] API to train our LDA model with two hyper parameters: the number K of topics and the number of I iterations. Here we set $K = 30$ and $I = 1000$. We will explain in detail the reasons we set these values in Section 4.

Finally, we get our LDA model after around 1.87 hours' training which is conducted on our server with an Intel(R) Core(TM) i7-7700 CPU with 3.60 GHz, 16 GB RAM, and Ubuntu 18.10. Results of our experiments will be presented in Section 3.

3 Experimental results

In this section, we conduct an in-depth investigation to answer the following five RQs.

- RQ1: What deep learning topics do developers ask?
- RQ2: What topics are more popular than others?
- RQ3: What topics are more difficult than others?
- RQ4: What are the differences between the issues faced by different deep learning frameworks?
- RQ5: What is the trend of each topic?

3.1 RQ1: What deep learning topics do developers ask?

After the above experimental steps, we have got the topic structure of our dataset, including 30 topics and two distributions, namely the word distribution of each topic (β_k) and the topic distribution of each post (θ_i). We want to further delve into each topic and give each topic a summation, thus giving developers a deeper insight of the deep-learning area and make them aware of what most developers are concerned about by this RQ.

Before determining the topic name, we give each topic an id (from 1 to 30) for the convenience of description. For each topic, i.e., topic k ($1 \leqslant k \leqslant 30$), we can use the word distribution β_k to sort all the words in descending order, so that we can obtain the top 10 words with the highest frequency as shown in the third column of Table 3.

6) https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html.

Table 3 Topic names and top 10 topic words

Number	Topic name	Topic word
1	Data Shape	shape, input, error, tensor, get, tri, array, float, torch, dimens
2	Loss Calculation	loss, function, calcul, softmax, metric, cross, use, mean, custom, cost
3	Convolution	convolut, filter, channel, input, kernel, size, map, imag, use, output
4	Package Installation	tensorflow, instal, python, version, use, tri, import, window, error, cuda
5	Model Save&Load	model, train, save, use, load, predict, tensorflow, restor, checkpoint, want
6	Visualization	file, graph, tensorboard, tensorflow, use, incept, node, summari, pb, name
7	API Usage	tensorflow, tf, use, graph, function, call, oper, like, session, way,
8	Classification	class, predict, label, classif, featur, classifi, output, use, one, binari
9	Variable Operation	variabl, weight, initi, optim, updat, tensorflow, share, scope, set, global
10	CNN Structure	layer, connect, conv, convolut, fulli, output, pool, cnn, dropout, network
11	Calculation Device	gpu, memori, run, use, cpu, tensorflow, time, devic, process, gb
12	Code Understanding	code, tri, tensorflow, work, get, use, help, follow, exampl, problem
13	Gradient Propagation	gradient, output, input, layer, function, network, hidden, weight, neuron, activ
14	Dataset	data, dataset, use, file, tensorflow, read, estim, tf, input, train
15	Model Transplanting	build, tensorflow, compil, java, android, bazel, librari, use, sourc, project
16	Method Introduction	would, like, use, way, question, one, need, know, look, time
17	Model Implementing	kera, model, use, gener, fit, backend, input, output, function, like
18	Neural Network	network, neural, caff, use, train, net, input, output, matlab, want
19	Reinforce Learning	learn, deep, state, action, algorithm, task, game, reinforc, reward, devic
20	Image Processing	imag, use, cnn, dataset, mnist, train, label, digit, pixel, classifi
21	Cloud Computing	tensorflow, googl, serv, ml, server, cloud, local, machin, cc, request
22	Sequence Prediction	lstm, sequenc, rnn, time, input, state, predict, output, length, cell
23	Package Importing	python, py, file, line, tensorflow, lib, packag, site, user, op
24	Batch Size	batch, size, sampl, number, distribut, thread, paramet, worker, gener, process
25	Word Embedding	word, embed, vector, text, encod, sentenc, use, hot, charact, one
26	Data Format	tensor, matrix, array, tensorflow, numpi, want, column, valu, element, row
27	Learning Rate	valu, learn, result, normal, rate, tri, differ, random, regress, chang
28	Generalization	train, test, data, set, accuraci, epoch, valid, step, dataset, model
29	Debug	error, run, tri, get, code, follow, theano, use, work, problem
30	Object Detection	object, detect, box, api, train, frame, video, tensorflow, use, bound

To better help us make judgments, we also find the top 10 posts under each topic. We assume that a post i belongs to a topic k, if and only if the topic k has the highest percentage of the post's topic distribution θ_i. Therefore, for each post i, we can find the topic k it belongs to and the corresponding percentage. For each topic, we record the posts that belong to it and sort them in descending order according to the percentage of this topic in the posts. In this way, we can take the top 10 posts for each topic. Figure 3 displays the post distribution, where the X-axis shows the topic ids, and the Y-axis indicates the number of posts under each topic as a percentage of the total number of posts. As can be seen, topic 12 contains the most posts (about 11.2%) and topic 9 contains the least (about 0.6%). Table 4 is an example list of the top 10 posts of topic k ($k = 1$), where the second column is the title of the top 10 questions, and the first column is the percentage of topic k in them.

Next, we present the top 10 words and top 10 posts of each topic to the three members of our research group, who all have a deep learning background. Each person tries to assign each topic a topic name based on his/her experience. Afterwards, we combine the results of the three people. If there is a conflict, the three will discuss again until they reach a unified opinion. The final results are presented in the second column of Table 3.

Through our analysis, we find that developers ask a broad spectrum of questions about deep learning, ranging from Data Shape to Object Detection. What's more, there are the most questions (about 11.2%) about Code Understanding and the fewest questions (about 0.6%) about Variable Operation.

3.2 RQ2: What topics are more popular than others?

Deep learning has developed rapidly in recent years, and a large number of researchers have paid attention to it. Researchers in this area often focus on only one or a few of these topics, since deep learning encompasses many topics, as we have summarized in RQ1. We want to know which topic gains more

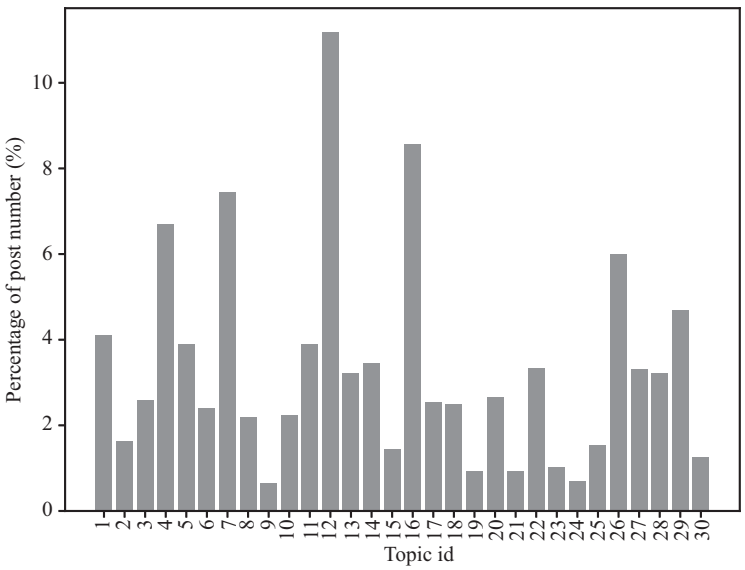

Figure 3 The post distribution under the topics.

Table 4 Top 10 questions of topic k ($k = 1$)

Topic proportion	Question title
0.95	Why does this tensorflow snippet throw an error in feeding?
0.94	Python array reshaping issue to array with shape (None, 192)
0.94	Tensorflow error with ConvLSTMCell: Dimensions of inputs should match
0.93	You must feed a value for placeholder tensor 'Placeholder' with dtype float and shape [2, 2]
0.91	How can I enumerate a tensor with unknown dim in tensorflow?
0.91	ValueError: Error when checking model target: expected dense_4 to have shape (None, 4) but got array with shape (13252, 1)
0.90	ValueError: Error when checking input: expected conv2d_1_input to have 4 dimensions, but got array with shape (120, 1)
0.90	Error when checking model input: expected flatten_input_8 to have 4 dimensions
0.88	ValueError: Cannot feed value of shape (2, 4) for Tensor u'InputData/X:0', which has shape '(?, 2, 4
0.88	ValueError: expected 2D or 3D input (got 1D input) PyTorch

popularity, so we can help researchers and developers aware of the trend in the deep learning area and keep pace with the development.

Intuitively, if a topic has more posts, and the posts of it have a higher average views, favorites, and score, this topic would be considered more popular. Therefore, we employ four indicators, namely average views (P_1), average favorites (P_2), average scores (P_3), and numbers of posts (P_4), to measure the popularity of a topic. Many studies have used the first three indicators (P_1, P_2, and P_3) to measure the popularity [13–15,33]. However, we think that the relationship between the number of posts and the popularity of the topic also cannot be ignored.

The formula of the four indicators are as follows:

$$P_1 = \frac{\sum_{i=0}^{n} v_i}{n}, \quad P_2 = \frac{\sum_{i=0}^{n} f_i}{n}, \quad P_3 = \frac{\sum_{i=0}^{n} s_i}{n}, \quad P_4 = n, \tag{1}$$

where v_i indicates the number of views in post i, f_i indicates the number of favorites in post i, s_i indicates the score of post i, and n indicates the total number of posts for the current topic.

As shown in Table 1, the values of view, favorite, and score can be found in the 'ViewCount', 'FavoriteCount', and 'Score' field in the XML file, respectively. All we need to do is finding the values for each post and averaging the values of posts under the same topic. Results are shown in Table 5, where the first column is the topic name, columns 2 and 3 are the calculation results of the indicator P_1 and the rankings of the topics according to P_1. Similarly, columns 4–9 result from indicators P_2, P_3, and P_4, respectively. In the last column, we make an average of the four rankings and sort the entire table in ascending order according to it.

Table 5 The results of popularity with 4 indicators (P_1–P_4) and their rankings

Topic	P_1	Rank.P_1	P_2	Rank.P_2	P_3	Rank.P_3	P_4	Rank.P_4	Avg.Rank
Gradient Propagation	2121	3	1.679	1	4.144	1	1058	13	4.5
API Usage	1892	6	1.258	6	3.735	3	2454	3	4.5
Method Introduction	1550	16	1.501	2	3.725	4	2820	2	6
Package Installation	4039	1	0.928	20	3.652	5	2208	4	7.5
Convolution	1911	4	1.327	4	3.430	6	854	16	7.5
Model Save&Load	1886	7	1.225	9	3.296	11	1280	9	9
Variable Operation	2430	2	1.240	7	4.058	2	208	30	10.25
Loss Calculation	1895	5	1.229	8	3.420	7	536	22	10.5
Calculation Device	1726	9	0.969	16	3.314	10	1282	8	10.75
Generalization	1722	10	1.089	11	3.330	9	1058	14	11
Cloud Computing	1669	12	1.303	5	3.388	8	304	28	13.25
Neural Network	1885	8	0.991	13	2.897	17	822	18	14
Visualization	1717	11	0.981	14	3.192	13	791	19	14.25
Learning Rate	1547	17	0.948	17	3.227	12	1086	12	14.5
Sequence Prediction	1210	28	1.356	3	3.038	16	1095	11	14.5
Code Understanding	1502	20	0.833	23	2.823	22	3687	1	16.5
Data Format	1546	18	0.599	27	2.827	21	1973	5	17.75
CNN Structure	1422	22	0.981	15	3.177	14	733	20	17.75
Word Embedding	1319	25	1.137	10	3.165	15	504	23	18.25
Debug	1624	15	0.482	28	2.444	26	1542	6	18.75
Data Shape	1632	13	0.460	29	2.294	28	1349	7	19.25
Classification	1525	19	0.941	19	2.842	18	723	21	19.25
Dataset	1249	26	0.895	21	2.794	24	1138	10	20.25
Model Implementing	1350	23	0.727	26	2.830	20	833	17	21.5
Image Processing	1347	24	0.747	25	2.396	27	876	15	22.75
Batch Size	1241	27	1.049	12	2.804	23	225	29	22.75
Model Transplanting	1431	21	0.762	24	2.688	25	474	24	23.5
Reinforce Learning	1030	29	0.879	22	2.836	19	305	27	24.25
Package Importing	1628	14	0.405	30	2.269	30	338	26	25
Object Detection	1029	30	0.942	18	2.274	29	413	25	25.5

As can be seen, Gradient Propagation has the 3rd P_1, the 1st P_2, the 1st P_3, the 13th P_4, and the average ranking of it is 4.5, which is the top one among all the topics. Besides, the same as Gradient Propagation, API Usage also has the most top average ranking, while Object Detection has the most bottom average ranking.

According to the four indicators we introduce, we can draw a conclusion that Gradient Propagation and API Usage are more popular than others, while Object Detection seems to gain less attention.

3.3 RQ3: What topics are more difficult than others?

Awareness of the difficulty of each topic is also significant to researchers and developers in the area. In this case, they can know what topic is suitable for them and what topic may need more effort.

Two indicators are introduced to measure the difficulty of topics [15], namely percentage of questions without accepted answer (D_1) and average time cost to be solved (D_2). We assume that the larger D_1 and D_2 are, the more difficult the topic is. The formulas are as follows:

$$D_1 = \frac{\sum_{i=0}^{n} \mathbb{I}(\text{question } i \text{ has no accepted answer})}{n},$$

$$D_2 = \frac{\sum_{i=0}^{n} \mathbb{I}(\text{question } i \text{ has accepted answer})(\text{st}_i - \text{ct}_i)}{\sum_{i=0}^{n} \mathbb{I}(\text{question } i \text{ has accepted answer})},$$

where n indicates the total number of posts for the topic, $\mathbb{I}(\cdot)$ is the indicator function (if $\cdot$ satisfies, $\mathbb{I}(\cdot)$ returns 1; else it returns 0), st_i is the solving time of question i, and ct_i is the creation time of question i.

Here, we give the detailed implementation of the computing for above two indicators D_1 and D_2. By checking whether the 'AcceptedAnswerId' field in the question is not empty (i.e., contains an answer id),

Table 6 The results of topic difficulty measured by two indicators (D_1 and D_2)

Topic	D_1	Rank.D_1	D_2	Rank.D_2	Avg.Rank
Object Detection	0.753	1	7.860	3	2
Model Transplanting	0.715	2	7.419	4	3
Cloud Computing	0.688	7	8.421	1	4
Visualization	0.702	5	7.368	5	5
Package Importing	0.710	3	7.269	7	5
Word Embedding	0.619	14	7.997	2	8
Sequence Prediction	0.637	12	7.367	6	9
Package Installation	0.689	6	6.630	13	9.5
Dataset	0.643	9	6.638	12	10.5
Calculation Device	0.704	4	6.241	17	10.5
Learning Rate	0.624	13	6.805	10	11.5
Debug	0.676	8	6.459	15	11.5
Image Processing	0.638	11	6.299	16	13.5
Reinforce Learning	0.570	23	6.884	8	15.5
Model Save&Load	0.641	10	5.941	21	15.5
Batch Size	0.582	21	6.703	11	16
Method Introduction	0.591	18	6.476	14	16
Gradient Propagation	0.547	26	6.864	9	17.5
Loss Calculation	0.593	16	5.945	20	18
Generalization	0.608	15	5.864	22	18.5
Neural Network	0.552	24	6.228	18	21
Code Understanding	0.586	20	5.685	23	21.5
API Usage	0.593	17	4.492	28	22.5
Classification	0.589	19	5.260	27	23
CNN Structure	0.538	28	6.161	19	23.5
Model Implementing	0.580	22	5.271	26	24
API Usage	0.547	27	5.613	25	26
Convolution	0.528	29	5.667	24	26.5
Variable Operation	0.548	25	3.682	30	27.5
Data Format	0.497	30	4.397	29	29.5

we can determine if this question has an accepted answer. In addition, the creation time can be obtained by the field 'CreationDate' for both questions and answers. We consider the creation time of the accepted answer as the solving time of the question. Experimental results are shown in Table 6. The first column is the topic name, columns 2 and 4 result from D_1 and D_2, and columns 3 and 5 are the corresponding rankings of the two indicators. Similar to Table 5, we average the two rankings and sort the topics in the last column.

As we can see, Object Detection has the top average ranking of 2 (with the 1-st D_1 and the 3-rd D_2) and Data Format has the lowest of 29.5 (with the 30-th D_1 and the 29-th D_2).

According to the two indicators, we find that different topics have different levels of difficulty. Object Detection is the most difficult among all the topics, while Data Format seems to be the easiest to handle.

3.4 RQ4: What are the differences between the issues faced by different deep learning frameworks?

There are a variety of excellent deep learning frameworks available for developers to choose from, which can greatly improve the efficiency of developers, such as tensorflow, torch, caffe. However, the previous research has shown that some of them are difficult to use [12]. We want to know whether developers of different frameworks are facing the same difficulty, or what their unique difficulty is. In this way, we can make a summary of the common topics of different frameworks and make some reasonable suggestions for the user and developer of deep learning frameworks.

First, we group the posts into five frameworks based on the tags. The tags we use are shown in Table 7, where the first column lists the names of the frameworks, and the second column shows the tags we choose for each framework. For each post, we check whether its tagset contains any tag listed in Table 7. If, for example, the tagset contains the tag 'pytorch', then we think this post is related to torch. In some cases,

Table 7 Tags of each framework

Framework	Tags
tensorflow	tensorflow
torch	torch, pytorch
caffe	caffe, pycaffe
theano	theano
keras	keras, keras-layer

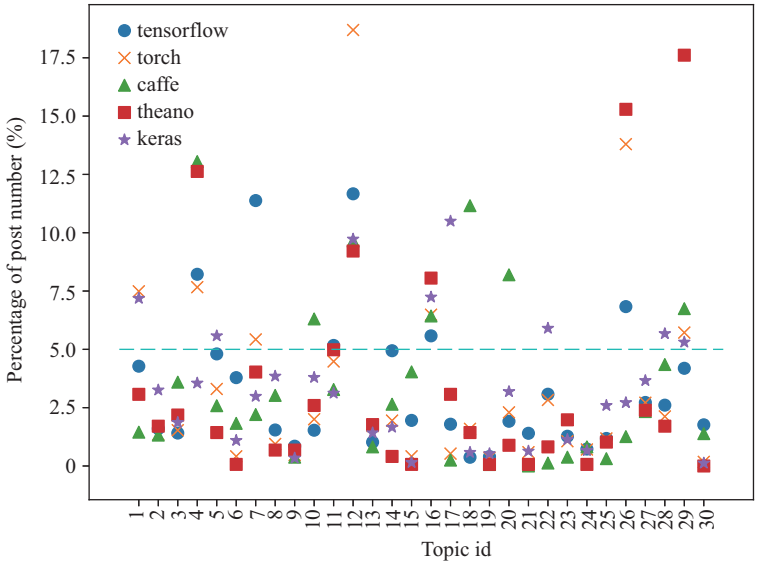

Figure 4 (Color online) The distributions of posts under each framework.

if the tagset also contains the tag 'keras-layer', we think this post is related to keras as well. Certainly, if none of the tags are contained in the tagset, then this post is not related to any framework.

Second, since we know the topics of all the posts, we can draw a post distribution for every framework, which is shown in Figure 4, where the X-axis indicates the topic id which is presented in Table 3, the Y-axis indicates the percentage of posts under each topic, and different frameworks are labeled by different shapes to distinguish them.

From the results, we can find that topic 12 (Code Understanding), as well as topic 16 (Method Introduction), accounts for more than 5% in all five frameworks. In addition, we notice that the proportion of topic 7 (API Usage) in tensorflow is significantly higher than other frameworks, and the same situation also occurs in the proportion of topic 12 (Code Understanding) in torch, topic 17 (Model Implementing) in keras, topic 18 (Neural Network) in caffe, and topic 29 (Debug) in theano.

For one thing, issues of Package Installation, Code Understanding and Method Introduction are common in all the five frameworks. For another, issues of API Usage are more common in tensorflow, the same as Code Understanding in torch, Model Implementing in keras, Neural Network in caffe, and Debug in theano.

3.5 RQ5: What is the trend of each topic?

Deep learning has flourished for many years, and many people are wondering whether this development momentum can continue, and whether it is wise to start research in this field now. We hope that our investigation can make people better aware of the development trend of various topics in this field and help them make reasonable research plans and avoid risks.

First, we count the number of posts created per month in last three years (from January 2016 to February 2019) for each topic. There are two reasons we choose the data of last three years. The first is that, compared with the data before too long, the number of posts in recent years is more representative. Second, some topics have few posts or large fluctuations of numbers before 2016, which will have a negative impact on our results. In addition, from our perspective, it is more reasonable to use the month

Table 8 The results of the trend test of the topics

Topic	p	s	Avg.#posts
Data Shape	**<0.001**	**379**	–
Loss Calculation	0.002	245	0.644
Convolution	0.004	−229	0.849
Package Installation	0.002	−243	2.673
Model Save&Load	**<0.001**	**395**	–
Visualization	0.003	240	0.960
API Usage	0.047	−159	3.045
Classification	0.940	7	0.841
Variable Operation	0.572	−46	0.240
CNN Structure	0.004	−231	0.923
Calculation Device	0.615	−41	1.601
Code Understanding	0.466	−59	4.338
Gradient Propagation	**<0.001**	**−315**	–
Dataset	0.097	133	1.409
Model Transplanting	0.013	−199	0.566
Method Introduction	**<0.001**	**−288**	–
Model Implementing	**<0.001**	**461**	–
Neural Network	**<0.001**	**−521**	–
Reinforce Learning	0.303	83	0.289
Image Processing	0.018	−189	1.047
Cloud Computing	**<0.001**	**339**	–
Sequence Prediction	0.960	−5	1.334
Package Importing	0.011	204	0.398
Batch Size	0.024	−181	0.283
Word Embedding	0.451	−61	0.635
Data Format	0.744	−27	2.362
Learning Rate	0.980	−3	1.242
Generalization	0.033	171	1.217
Debug	0.315	81	1.762
Object Detection	**<0.001**	**318**	–

instead of the day and the year as the time unit, because it will make the data relatively stable and keep the number of data points sufficient for further analysis.

Then, we can conduct a trend test[7)] for each topic by a non-parametric method called Mann-Kendall test [34], which is implemented in R language. The null hypothesis of Mann-Kendall test is that there is no monotonous trend in data, and the alternative hypothesis is that the data follows a monotonic trend. If the p value is less than the significance level, it means that the null hypothesis can be rejected; in other words, the data has a rising or falling trend. Results are shown in Table 8. As the table shows, we set the significance level to 0.001 here to find the trend with more confidence [35] and mark the corresponding line that satisfies the significance level in bold. Further more, whether the trend is rising or falling can be determined by the positive or negative of the s value (the third column in Table 8, which is a statistic in Mann-Kendall test [34]). In this way, we can find the topics with significant rising or falling trends. For the topics with no obvious trend, we calculate their average number of posts per month in these years, which is shown in the last column in Table 8, so we can believe that their numbers of posts will maintain this level in the future with high probability.

Finally, we verify the correctness of our results by observing the curves in Figure 5, which reveals how the number of posts changes over time under each topic. In Figure 5, we use different colors to represent different topics. The X-axis indicates the time, and the Y-axis is the ratio of the number of posts under the topic to the total number of posts. We use three subgraphs (Figures 5(a), (b) and (c)) to draw topics with rising trends, with falling trends and remaining stable, where in third subgraph, we only draw the one with the highest mean and the two with the lowest mean for the convenience of observation.

Data Shape, Model Save&Load, Model Implementing, Cloud Computing, and Object Detection have significant rising trends in recent three years, while Gradient Propagation, Method Introduction, and

7) https://en.wikipedia.org/wiki/Trend_analysis.

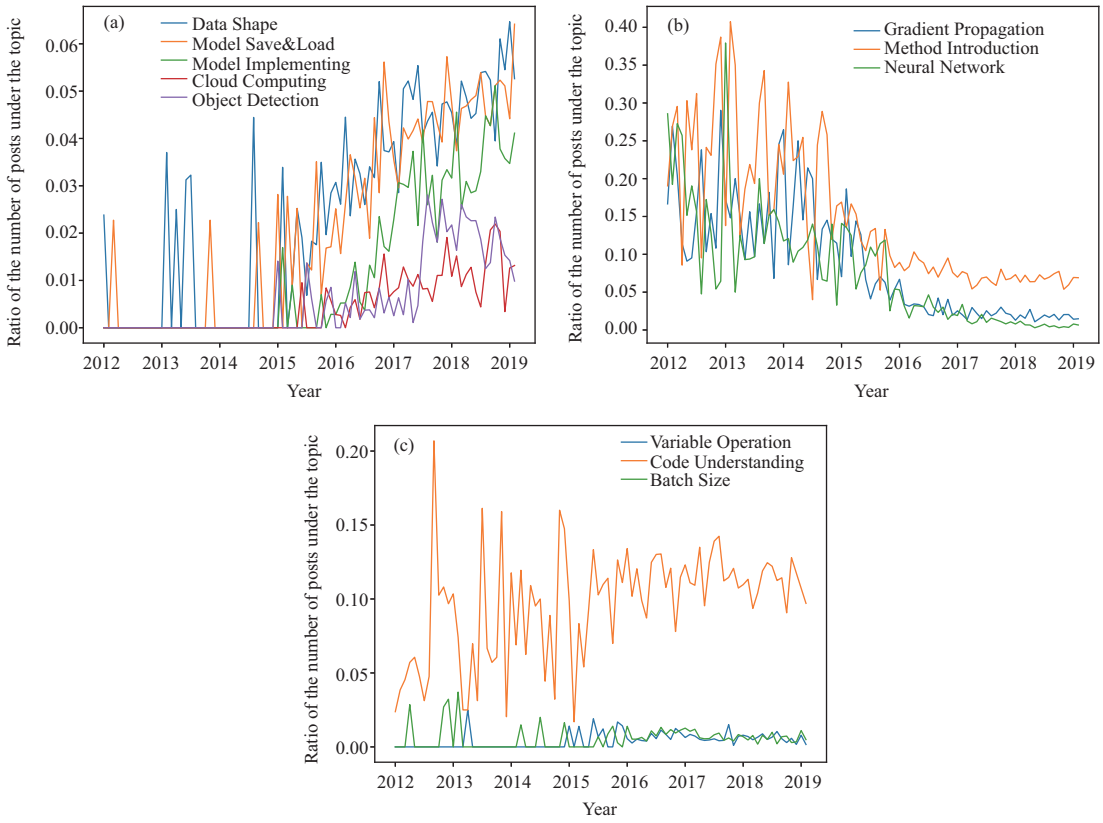

Figure 5 (Color online) Changes of post numbers over years. (a) Rising trend; (b) falling trend; (c) remain stable.

Neural Network have a significant falling trend in these years. What's more, Code Understanding seems to maintain a relatively high number of discussions since the average number of posts is significantly higher than other topics, while the numbers of discussions on Variable Operation and Batch Size are very small.

4 Discussion

In this section, we have a discussion about the details of our research. First, we demonstrate the effectiveness and necessity of our data augmentation steps. Next, we evaluate the impact of stemming by two additional experiments. Then, we explain the selection of parameters during the training of our LDA model, including the topic number K and the iteration number I. Besides, we explore the impact of tag accuracy on the dataset and make a reasonable and effective assessment of the quality of our dataset. In addition, we conduct correlation tests for the indicators we introduced to measure the popularity and difficulty. Moreover, we discuss the characteristics of each topic, and finally, some targeted suggestions are made for developers, researchers, educators, and framework providers of deep learning.

4.1 The effectiveness of data augmentation

As described in Subsection 2.1.2, in order to enrich our dataset, we perform the data augmentation step based on the tag information of the posts. In this subsection, we explore the effectiveness of the data augmentation step.

We divide our whole dataset D_a (mentioned in Subsection 2.1.2, with totally 32969 posts) into two parts D_{a_1} and D_{a_2}, where D_{a_1} is the dataset that we will get if we remove the data augmentation step, and D_{a_2} is the dataset we add through the data augmentation step. Based on this classification, we observe that there are 5527 posts in D_{a_1}, which is equal to the number of posts with non-negative score in D_f (mentioned in Subsection 2.1.1), and 27442 posts in D_{a_2}. That means after data augmentation, we end up with a dataset that is 496.5% larger than the dataset without data augmentation. Therefore, data augmentation not only increases the diversity of the dataset through introducing more tags (the

Table 9 One of the post added by data augmentation steps

Field	Content
Title	Tensorflow class balancing for training multi-class classification network
Body	I am training a classification neural network for multiple classes. I have very imbalanced classes (80:10:5:5 ratio approximately). I want to use some kind of weight balancing in the loss function to prevent the neural network from overly predicting for the majority class. Does anybody know how to do the class balancing in tensorflow? P.S. I cannot solve this by over-sampling the minority classes because I am training a convolutional-deconvolutional neural network that does medical image segmentation. Each pixels is assigned to a distinct class in this task. I cannot over sample pixels in this task. Thanks a lot!:D Than
Tags	tensorflow, conv-neural-network, neural-network, image-segmentation

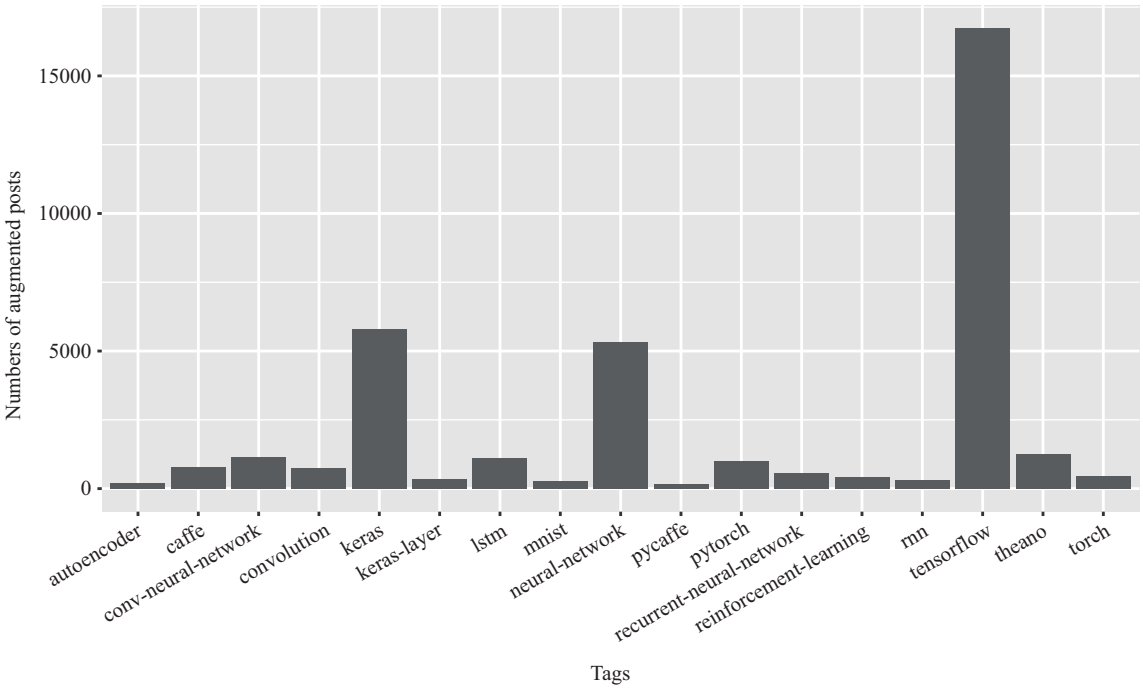

Figure 6 Numbers of posts added in data augmentation.

third row in Table 2), but also greatly increases the size of the dataset, providing solid data support for subsequent analysis.

For example, although the post (presented in Table 9) has not the 'deep-learning' tag, its content is obviously related to deep learning. After the data augmentation step, this post is added to our dataset because it has the 'tensorflow' tag, which increases the size and diversity of our dataset.

In addition, in order to better evaluate the contribution of each tag in data augmentation, we calculate the number of posts added by each tag. Specifically, for each post in D_{a_2}, if it contains a certain tag in the third row of Table 2, this post is considered to be added by this tag. Finally, we get the tag frequency distribution as shown in Figure 6. It can be seen from Figure 6 that each tag plays a significant role in data augmentation. Among them, 'tensorflow' contributes the most posts, reaching 16751, and 'pycaffe' contributes the least, reaching 160. On average, each tag can contribute 1614 posts.

4.2 The impact of stemming

In the step of data pre-processing, we used stemming algorithm [14, 15, 17–19], by which different words may be converted to the same form. For example, 'using' and 'used' will be both stemmed as 'use', thereby reducing the impact of plural forms of words or the impact of tense changes. However, some of the disadvantages of stemming cannot be ignored, such as the interference with the analysis of some terminology. In this subsection, we design two experiments to further discuss the impact of stemming.

On the one hand, we conduct an experiment to compare the difference between the results obtained with and without stemming. In detail, we train a new LDA model under exactly the same experimental settings described in Section 2, where the only difference is that the stemming step is removed. Thus,

Table 10 Top 10 topic words without stemming[a)]

Number	Topic word
1	function, loss, **gradient**, custom, **gradients**, mean, tensorflow, compute, calculate, cost
2	python, py, line, file, lib, tensorflow, packages, site, self, local
3	caffe, tensorboard, build, tensorflow, file, bazel, library, source, compile, error
4	theano, pytorch, torch, attribute, module, function, lua, object, attributeerror, code
5	**variable**, **variables**, tensorflow, graph, session, tf, op, get, run, want
6	**batch**, **size**, number, **batches**, **samples**, input, **sample**, num, mini, **sizes**
7	**training**, loss, **train**, set, epoch, validation, test, data, model, step
8	tf, tensorflow, graph, api, estimator, **use**, **using**, contrib, inference, input
9	time, memory, **run**, **running**, using, code, process, multiple, takes, large
10	google, android, ml, tensorflow, cloud, docker, app, engine, demo, run
11	tensorflow, python, error, version, **installed**, **install**, import, using, windows, run
12	**gpu**, tensorflow, cpu, device, cuda, **gpus**, nvidia, run, use, memory
13	lstm, time, rnn, sequence, input, length, data, series, output, state
14	code, tensorflow, like, example, get, trying, would, using, help, work
15	**network**, neural, **networks**, net, **using**, **training**, **train**, matlab, **use**, **trained**
16	data, dataset, **file**, **training**, **train**, **using**, **use**, read, set, **files**
17	keras, model, **using**, fit, generator, backend, **use**, autoencoder, like, sequential
18	**tensor**, matrix, shape, tensorflow, want, vector, like, **tensors**, dimension, way
19	**images**, **image**, object, detection, using, tensorflow, cnn, train, dataset, api
20	**class**, **classes**, output, **classification**, **labels**, one, **label**, seq, binary, softmax
21	**model**, trained, **save**, tensorflow, using, file, **models**, load, **saved**, weights
22	array, tensor, input, numpy, list, feed, tensorflow, float, function, type
23	accuracy, **results**, learning, different, rate, using, tried, problem, code, **result**
24	error, code, following, shape, trying, **get**, **getting**, input, **got**, problem
25	output, **weights**, layer, hidden, function, input, activation, **weight**, neurons, values
26	**word**, **embedding**, text, java, **words**, **embeddings**, sentence, vec, **vectors**, **vector**
27	**image**, **images**, convolution, using, input, filter, kernel, pixel, output, size
28	learning, deep, algorithm, machine, state, problem, action, would, value, reinforcement
29	would, like, one, way, question, use, two, different, could, know
30	**layer**, **layers**, input, conv, output, caffe, cnn, convolutional, feature, network

a) Different forms of words in the same topic are highlighted in bold.

we have the top 10 topic words of 30 topics similar to Table 3. The results are shown in Table 10, where the second column presents the top 10 topic words. As can be seen, in most (21 out of 30, 70%) cases, different forms of topic words appear in one topic, e.g., 'variable' and 'variables' in topic 5, 'use' and 'using' in topic 8, 'run' and 'running' in topic 9, and 'install' and 'installed' in topic 11. Compared with the result in Table 3, we conclude that stemming can indeed prevent the interference with plural forms and tense changes.

On the other hand, there is a concern that stemming may disturb the analysis of some terminology. To this end, we take the following steps to assess the negative impact of stemming. First, we record each pair of the word and its stem (e.g., using and use) during the stemming process introduced in Subsection 2.1.3 and build a dictionary T according to all these pairs. Specifically, the key of the dictionary T is a stem, and the value is a list which contains the words that can be converted to the corresponding stem by Porter Stemming Algorithm[8)], e.g., with the key 'use', the value list contains 'used', 'uses', and 'using'. Then we collect a total of 172 stems[9)] that have appeared as top 10 topic words in Table 3, and obtain their possible original forms according to T. After a manual review of these words, we find that after stemming, there are indeed some words that will affect the analysis of terminology. For example, 'important' will become 'import' after stemming, but 'import' actually indicates importing packages in Python, which has different meanings of 'important'. However, we observe that the similar case only appears on 5 stems, and the proportion affected is relatively low, i.e., about 3% (5/172).

After the above experiments, we observe that stemming does have a slightly negative impact on our results, and this impact is almost negligible compared to the benefits it brings.

8) https://tartarus.org/martin/PorterStemmer.

9) One stem may appear multiple times in the total of 300 top 10 topic words for 30 topics in Table 3.

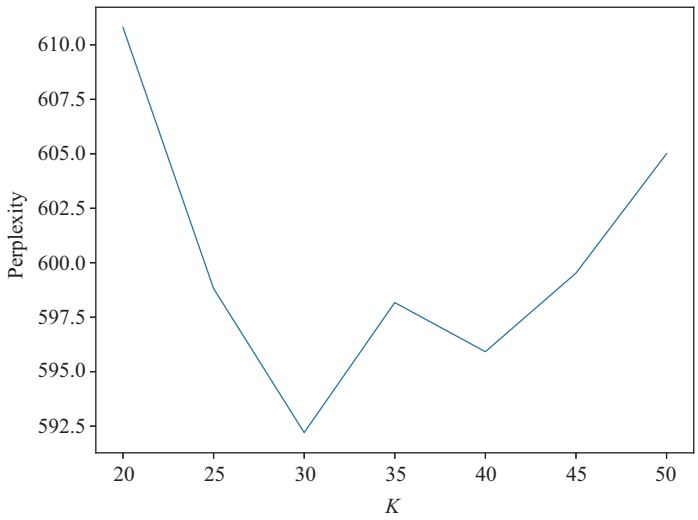

Figure 7 (Color online) Perplexities of models with different setting of topic number K.

4.3 Iteration number I and topic number K

In this subsection, we will detailedly describe the selection of parameters during the training of our LDA model, including the iteration number I and the topic number K.

The choice of the iteration number I is relatively simple, because the bigger the number of iterations is, the better the performance of the model is. We try to choose an I as large as computing resources allow. In general, the number of iterations should be over 500, so we choose the number of iterations I as 1000.

To determine the number of topics K, we conduct a pre-experiment with a broad range of different K ($K \in \{20, 25, 30, 35, 40, 45, 50\}$) to check the performance of perplexity. The perplexity is designed to measure the likelihood of words appearing in the corpus and hence to evaluate the performance of LDA modeling: the lower perplexity value means the better performance [26]. The detailed results of the pre-experiment are shown in Figure 7, where X-axis indicates the models with different numbers of topics, and Y-axis indicates the perplexity of different models[10]. As can be seen, the perplexity value of the model with 30 topics is around 592.5, which is the lowest among all models. What's more, $K = 30$ seems to be the minimal point of the curve (there are a decreasing trend on the left side and an increasing trend on the right side). So we set the value of K as 30 to get a relatively satisfactory result.

Besides, the overlap between different topics is another factor to determine the number of topics. When the number of topics is set inappropriately, different topics may contain large overlaps. To further explore whether $K = 30$ is appropriate, we measure the overlaps of the 30 topics by calculating the numbers of overlapped top 10 words between each two topics. The top 10 topic words are already obtained in Table 3. For each topic, we calculate the number of overlaps of the top 10 words between it and the other 29 topics. In this way, we have the boxplots of overlapped numbers as shown in Figure 8.

It can be seen from Figure 8 that for most cases the numbers of overlapped topic words between two topics are 0 or 1, since the medians of most boxplots are less than 2. Only for one pair (topic 12 and topic 29) out of $30 \times 29/2 = 435$ pairs, the overlap number exceeds 3. It means that the overlaps between most pairs of topics are small.

4.4 Assessment of tag accuracy

Our experimental datasets are selected based on tags of posts. The inaccuracy of tags may have an impact on our research results. In this subsection, we will provide additional analysis about the accuracy of tags: (a) tag accuracy of posts related to deep learning and (b) tag accuracy of posts with different framework tags.

10) The LatentDirichletAllocation API supports the calculation of perplexity.

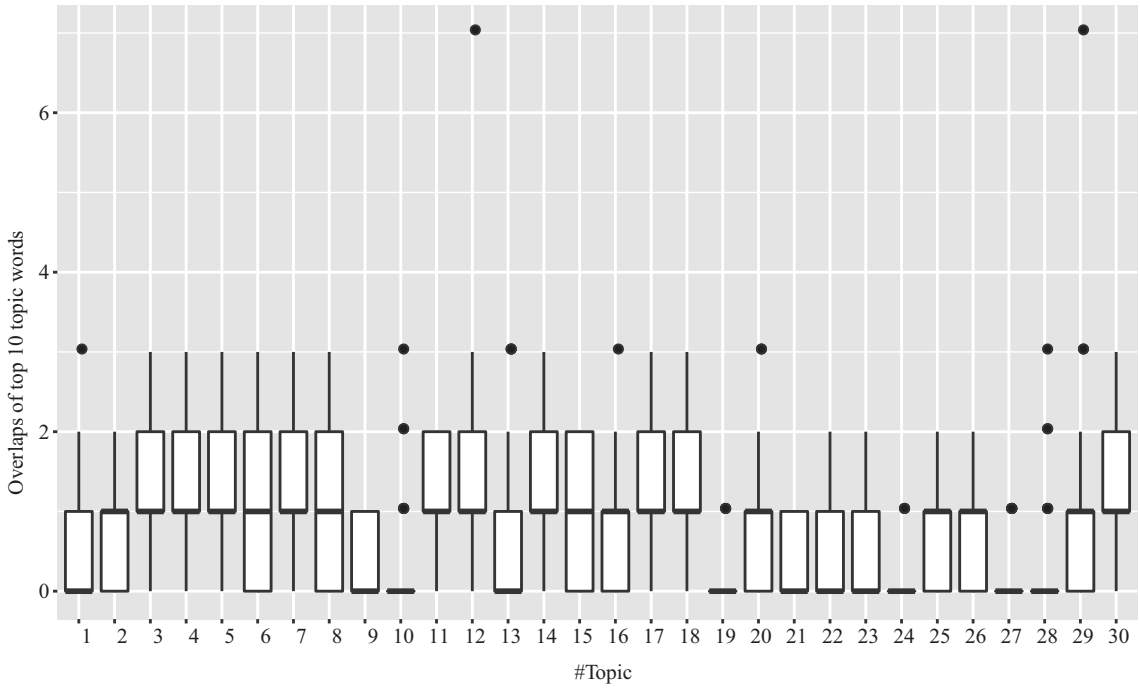

Figure 8 Boxplot of overlaps of 30 topics.

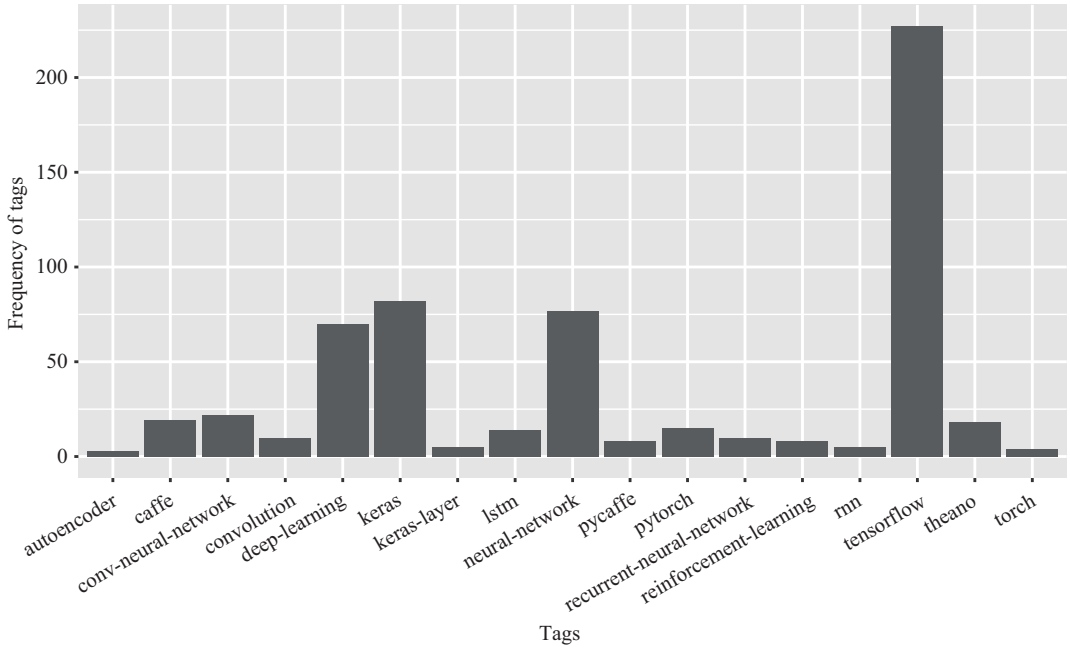

Figure 9 Tag frequency of the sampled data.

4.4.1 *Tag accuracy of posts related to deep learning*

Since our dataset contains a huge number of posts, it is not realistic to review each of them to see whether it is actually related to deep learning. To overcome this deficiency, we randomly sample 400 posts from our dataset to reach a confidence level of 95% [36]. The selected posts cover 17 of all 18 tags (i.e., tags listed in Table 2), and the frequency of each tag is shown in Figure 9. It can be seen that the frequency of 'tensorflow' is the highest and the frequency of 'autoencoder' is the lowest. The tag frequency of the sampled data is basically consistent with our whole dataset.

For these 400 posts, we conduct a manual review to see if the content is related to deep learning. Specifically, three members in our research group with deep learning research background mark each

Table 11 An example of post with inaccurate tags

Field	Content
Title	How to activate VirtualEnv on IDE
Body	I need to Debug my Project using IDE but It requires to be run on a virtualenv. I have created a virtualenv on the IDE Clion using this documentation: But I am not able to activate it. Will it get automatically activated, when I run the project? I have tried that also but it is not running. when I am running it from terminal everything works fine but debugging becomes difficult as I am not much familiar with gdb. How do I activate my virtualenv on IDE?? Any help will be appreciated. Thanks Ashish
Tags	virtualenv, keras, clion

Table 12 Correlations between four indicators (P_1 to P_4) of popularity

	P_1	P_2	P_3	P_4
P_1	–	$r = 0.171, p = 0.367$	$r = 0.545, p = 0.002$	$r = 0.251, p = 0.180$
P_2	–	–	$r = 0.816, p < 0.001$	$r = 0.014, p = 0.942$
P_3	–	–	–	$r = 0.167, p = 0.378$
P_4	–	–	–	–

Table 13 Correlations between two indicators (D_1 and D_2) of difficulty

	D_1	D_2
D_1	–	$r = 0.640, p < 0.001$
D_2	–	–

post to determine whether it is related to deep learning, independently. If the preliminary results of the manual review are different, the final result is determined by voting[11)]. Our results show that a total of 4 posts are marked as "unrelated". Thus, the inaccurate rate of our dataset is 1% (4/400). Table 11 shows one post with inaccurate tags, which has the 'keras' tag. When checking manually at its content, we find that, as the title shows, the post is wondering how to activate the VirtualEnv on IDE, which is not related to deep learning.

As a result, the inaccuracy of tags may cause our dataset to include a very small percentage (i.e., 1%) of posts, which are not related to deep learning.

4.4.2 *Tag accuracy of posts related to different frameworks*

Our analysis of different frameworks (Subsection 3.4) may also be influenced by tag accuracy, since posts that have multiple framework tags may only focus on one of them. To assess the impact of tag accuracy related to different frameworks, we conduct the following experiments.

First, we count all the 27461 posts that are related to at least one framework (using the tags as described in Table 7), and find that a total of 3558 posts are related to two or more than two frameworks. The proportion of multi-framework posts is 13.0% (3558/27461). Second, we extract all 35 posts with multiple framework tags from the 400 posts collected in Subsection 4.4.1 and conduct a manual review. We find that 4 of the 35 posts, their content is only related to the part of the framework tags. The inaccurate rate is 11.4% (4/35).

Therefore, we can estimate an error rate at around 1.5% (11.4%×13%) on the entire dataset when considering the posts with multiple framework tags, which is considered to be acceptable.

4.5 Correlations between the different indicators of popularity/difficulty

In Section 3, we have proposed four indicators to measure the popularity and two indicators to measure the difficulty. We want to know if there are significant correlations between these indicators.

To this end, we conduct a Pearson correlation analysis [37] of these indicators. The results are shown in Tables 12 and 13, where Table 12 represents the correlation among four popularity indicators P_1, P_2, P_3 and P_4, and Table 13 represents the correlation between two difficulty indicators D_1 and D_2. In the correlation test, the r value indicates the degree of correlation, and the p value is used to measure whether it is significant. It is generally considered that when $p < 0.001$, there is a statistically significant high correlation between two indicators [37].

11) For example, if two members vote "related" and one votes "unrelated", we judge the post as "related".

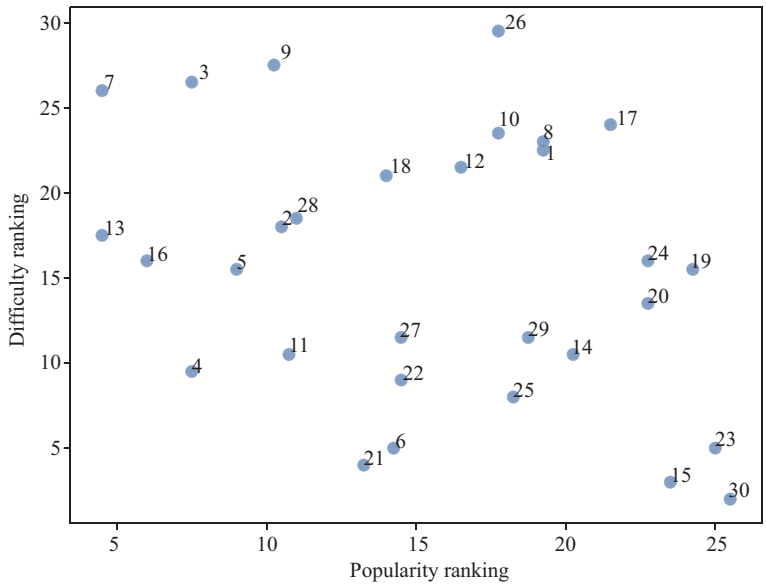

Figure 10 (Color online) Correlation between popularity and difficulty.

As can be seen in Table 12, the correlation coefficient between P_2 (average favorites) and P_3 (average score) is around 0.816 and the p value is far lower than 0.001, which means that these two indicators are highly correlated, and we may only need to focus on one of these two indicators. In addition, there is no significant correlation between P_4 (the number of posts) and the three other indicators (P_1, P_2, P_3), showing that our new proposed indicator P_4 is statistically different from the three commonly used indicators. Therefore, we believe that P_4 is irreplaceable and valuable for measuring popularity.

Similarly, Table 13 shows that the Pearson correlation coefficient r between these two indicators D_1 and D_2 is near 0.640, and the p value is less than 0.001, which means that there is a significant correlation between the two indicators. When we discuss the degree of difficulty, these two indicators may have a certain equivalence.

4.6 Characteristics of the topics

Through the above research, we have summarized 30 topics about deep learning and have a general understanding of the difficulty and popularity of each topic. Here, in order to display the characteristics of each topic more intuitively, we draw Figure 10, where X-axis indicates the average ranking of popularity (the last column of Table 5), and Y-axis indicates the average ranking of difficulty (the last column of Table 6).

As can be seen from the figure above, the topics are scattered in the coordinate plane, and there is no obvious correlation between the popularity and difficulty of the topics. The topics in the lower left corner of the figure are both popular and difficult, such as topic 4 (Package Installation), while the topics in the upper right corner are relatively easy and unpopular, such as topic 17 (Model Implementing). Thus, we can know better about the characteristics of each topic and better advise different types of people in Subsection 4.7.

4.7 Implications for developers, researchers, educators and framework providers

According to conclusion obtained above, we can give some implications to people in the field of deep learning.

Developers. Using Figure 10, novice developers can choose to focus on topics with higher popularity and lower difficulty, such as topic 7 (API Usage), topic 3 (Convolution), and topic 9 (Variable Operation), so they can get help more easily when they are in trouble. Experienced developers can try more challenging topics such as topic 13 (Gradient Propagation), topic 16 (Method Introduction) or topic 5 (Model Save&Load). In addition, developers should be more concerned with the trend of the topics, pay more attention to topics with rising trends, and should be cautious for topics showing a falling trend. From this perspective, as Figure 5 shows, topics like Cloud Computing, Object Detection and so on are

recommended for all the developers, and topics like Neural Network and Gradient Propagation are not recommended.

Besides, the results of our observation may bring new directions of the development of automation tools in the deep learning field. For example, the popularity of API Usage and Package Installation issues is relatively high. If developers can develop some automatic API analysis tools and package installation tools, these two hot issues will be well solved. In addition, developing automatic model migration tools and visualization tools will be very helpful to solve some relatively more difficult problems such as Model Transplanting and Visualization. Finally, if there is an automatic tool that can help save and load deep learning models, the problems of topic Model Save&Load that have a rising trend will be alleviated.

Researchers. Researchers should pay more attention to topics that are more difficult and less popular because such topics would be more challenging and interesting, and in the meantime, more likely to make contributions owing to less attention. As shown in Figure 10, topic 30 (Object Detection), topic 23 (Package Importing), and topic 15 (Model Transplanting) may be the suitable topics.

Educators. Figure 10 can facilitate individualized education and help educators schedule their courses more reasonably. For beginners, educators can teach some topics easy to start, such as topic 26 (Data Format) and topic 9 (Variable Operation). For students with a wealth of knowledge, some difficult topics are recommended, such as topic 11 (Calculation Device), topic 27 (Learning Rate) or topic 29 (Debug).

Framework providers. From the conclusion we presented in RQ4, we can give some suggestions to each framework that tensorflow should pay more attention to improve their performance on API Usage, as well as Code Understanding in torch, Model Implementing in keras, Neural Network in caffe, and Debug in theano. In addition, a new framework that wants to stand out from current frameworks needs to address some common problems in topics such as Package Installation, Code Understanding, and Method Introduction.

5 Threats to validity

There are a number of factors that may pose a potential threat to our research. In this section, we discuss these threats from both internal and external perspectives.

Internal threats. One of the internal threats is that our dataset may not contain all the post about deep learning. To mitigate the impact of this threat, we did not only use the 'deep-learning' tag, but also expanded the tag set as much as possible with the method we described in Subsection 2.2.

Besides, the limitation of LDA methods might be another internal threat, especially because it requires a fixed number of topics K. In order to choose a relatively appropriate K, we made a preliminary experiment with different K and determine $K = 30$ according to perplexity. Under the determined $K = 30$, we conducted an additional analysis to evaluate the overlaps of the 30 topics in Subsection 4.3, which shows that the overlaps between different topics are small.

In addition, the topic name is assigned manually, which can also be considered one of the threats. To minimize this impact, for one thing, we found the top 10 words and top 10 posts of each topic to facilitate this process. For another, members of our research group with deep learning background discussed repeatedly until a consensus was reached.

As stated in Subsection 2.1.2, we determined the thresholds by manual observation, which may be another threat. In the future, we will consider refining it by introducing a more objective method.

Finally, the measurements of popularity and difficulty can also be a threat. To alleviate this threat, we decided to use four indicators for popularity and two indicators for difficulty and conducted a statistical analysis of the indicators to learn their relevance.

External threats. Using Stack Overflow as the only source of data could be one of our main external threats. Although Stack Overflow is a widely used and popular Q&A website, it does not cover all questions about deep learning. In the future, we may consider more data sources and increase the diversity of data to mitigate this threat.

Besides, as we discussed in Subsection 4.4, the tag accuracy has a certain impact on the quality of our dataset. Although this impact is relatively low, it still cannot be ignored. In the future, we hope to further mitigate this impact by combining some text analysis methods.

Data quality of Stack Overflow itself is also a threat for us. To minimize this threat, we excluded the posts with non-positive scores, which guarantees the quality of our dataset to some extent.

6 Related work

6.1 Empirical studies on Stack Overflow

There are a number of studies using Stack Overflow as their data source and conducting research through text analysis. Allamanis and Sutton [17] used topic model to associate programming concepts and identifiers with particular types of questions on Stack Overflow. Barua et al. [18] applied LDA on Stack Overflow and analyzed the discovered topics, as well as their relationships and trends over time, to gain insights into the development community. Rosen et al. [13] employed latent Dirichlet allocation-based topic models to summarize the mobile-related questions from Stack Overflow, as well as determined what popular mobile-related issues were the most difficult, explored platform specific issues, and investigated the types of questions mobile developers asked. Bajaj et al. [19] used LDA to categorize the questions and defined a ranking algorithm to rank all the Stack Overflow questions based on their importance. Beyer et al. [38] presented a manual categorization of 450 Android related posts of Stack Overflow concerning their question and problem types and found dependencies between certain problems and question types to get better insights into issues of Android App development. Yang et al. [14] used LDA tuned using genetic algorithm (GA) to cluster different security-related questions, summarized all the topics into five main categories, and investigated the popularity and difficulty of different topics. Ahmed et al. [15] conducted a large-scale study on the textual content of the entirety of Stack Overflow with LDA to understand the interests and difficulties of concurrency developers. Han et al. [39] applied LDA topic modeling techniques to derive the discussion topics related to three popular deep learning frameworks on Stack Overflow and Github.

However, in this paper, we take deep learning as the research focus, which is not covered by previous studies. In addition, we have carried out an extensive study, including topic assignments, analysis of popularity and difficulty of the questions, comparison between different deep learning frameworks, and trend analysis of topics. Based on our research results, many valuable suggestions have been made to the practitioners of deep learning.

6.2 LDA and other topic models

LDA is a kind of topic model that is widely used in natural language processing and data mining. Many studies chose to use LDA to model document collections [15, 18, 19]. However, the LDA model has its limitations. For example, the results of LDA are closely related to the choice of the number of topics, and LDA is biased towards larger sized corpora. Therefore, some studies have improved the limitations of the LDA model and proposed other models. Wan et al. [40] proposed an approach that combined balanced LDA (which ensured that the topics were balanced across a domain) with the reference architecture of a domain to capture and compare the popularity and impact of discussion topics across the Stack Exchange communities. Huang et al. [41] proposed a novel emerging topics tracking method, which aligned emerging word detection from temporal perspective with coherent topic mining from spatial perspective to address the problem of detecting emerging topics early and monitor evolving topics over time effectively. Zhu et al. [42] proposed a time-aware topic modeling perspective to model user emotions for online news with respect to time spans. Specifically, they first developed two models named emotion-topic over time (eToT) and mixed emotion-topic over time (meToT) to uncover the latent relationship among news, emotion and time directly. Then they further developed another model named emotion-based dynamic topic model (eDTM) to capture the evolution of topics. Xu et al. [43] built a job skill network and developed a novel skill popularity based topic model (SPTM) for modeling the generation of the skill network in the job market.

7 Conclusion

In this paper, we conducted a large-scale experiment using data of posts from Stack Overflow to analyze the topics of problems encountered by deep-learning developers. After data filtering, augmentation, and pre-processing, we inferred 30 topics of deep learning from the posts with the help of LDA topic model, including Data Shape, Object Detection and so on. In addition, we measured the difficulty and popularity of each topic separately and found that Gradient Propagation was the most popular topic and Object Detection was the most difficult. Moreover, we studied the distribution of topics in different deep

learning frameworks and found that issues of Package Installation, Code Understanding, and Method Introduction were common in all of them. Through trend analysis of different topics, we found that there were three kinds of trends in the 30 topics, namely rising trend, falling trend, and remaining stable. Finally, we discussed some details of our experiment, e.g., correlations between the different indicators of popularity/difficulty and characteristics of the topics. Based on our findings, we gave developers, researchers, educators and framework providers some practical implications.

In the future, we will introduce more data sources, take some other models into account, and seek a more general and reliable way to study these research questions on a larger scale.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2018YFB1003901) and National Natural Science Foundation of China (Grant Nos. 61872177, 61772259, 61972289, 61832009). We thank the anonymous referees for their helpful comments on this paper.

References

1 Wan Z, Xia X, Lo D, et al. How does machine learning change software development practices? IEEE Trans Software Eng, 2020. doi: 10.1109/TSE.2019.2937083
2 Graves A, Mohamed A, Hinton G E. Speech recognition with deep recurrent neural networks. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing, 2013. 6645–6649
3 Ba J, Mnih V, Kavukcuoglu K. Multiple object recognition with visual attention. 2015. ArXiv: 1412.7755
4 Redmon J, Divvala S K, Girshick R B, et al. You only look once: unified, real-time object detection. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, 2016. 779–788
5 Gawehn E, Hiss J A, Schneider G. Deep learning in drug discovery. Mol Inf, 2016, 35: 3–14
6 Park Y, Kellis M. Deep learning for regulatory genomics. Nat Biotechnol, 2015, 33: 825–826
7 Abadi M, Barham P, Chen J, et al. Tensorflow: a system for large-scale machine learning. In: Proceedings of 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, 2016. 265–283
8 Collobert R, Kavukcuoglu K, Farabet C. Torch7: a matlab-like environment for machine learning. In: Proceedings of Neural Information Processing Systems, 2011
9 Jia Y, Shelhamer E, Donahue J, et al. Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM International Conference on Multimedia 2014. 675–678
10 Theano Development Team. Theano: a Python framework for fast computation of mathematical expressions. 2016. ArXiv:1605.02688
11 Schmidhuber J. Deep learning in neural networks: an overview. Neural Netw, 2015, 61: 85–117
12 Erickson B J, Korfiatis P, Akkus Z, et al. Toolkits and libraries for deep learning. J Digit Imag, 2017, 30: 400–405
13 Rosen C, Shihab E. What are mobile developers asking about? A large scale study using stack overflow. Empir Softw Eng, 2016, 21: 1192–1223
14 Yang X L, Lo D, Xia X, et al. What security questions do developers ask? A large-scale study of stack overflow posts. J Comput Sci Technol, 2016, 31: 910–924
15 Ahmed S, Bagherzadeh M. What do concurrency developers ask about? A large-scale study using stack overflow. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Oulu, 2018. 1–10
16 Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation. J Mach Learn Res, 2012, 3: 993–1022
17 Allamanis M, Sutton C. Why, when, and what: analyzing stack overflow questions by topic, type, and code. In: Proceedings of the 10th Working Conference on Mining Software Repositories, Piscataway, 2013. 53–56
18 Barua A, Thomas S W, Hassan A E. What are developers talking about? An analysis of topics and trends in Stack Overflow. Empir Softw Eng, 2014, 19: 619–654
19 Bajaj K, Pattabiraman K, Mesbah A. Mining questions asked by web developers. In: Proceedings of the 11th Working Conference on Mining Software Repositories, Hyderabad, 2014. 112–121
20 Rama G M, Sarkar S, Heafield K. Mining business topics in source code using latent dirichlet allocation. In: Proceedings of the 1st Annual India Software Engineering Conference, Hyderabad, 2008. 113–120
21 Arora R, Ravindran B. Latent dirichlet allocation based multi-document summarization. In: Proceedings of the 2nd Workshop on Analytics for Noisy Unstructured Text Data, New York, 2008. 91–97
22 Bolelli L, Ertekin Ş, Giles C L. Topic and trend detection in text collections using latent dirichlet allocation. In: Advances in Information Retrieval. Berlin: Springer, 2009. 776–780
23 Tirunillai S, Tellis G J. Mining marketing meaning from online chatter: strategic brand analysis of big data using latent dirichlet allocation. J Marketing Res, 2014, 51: 463–479
24 Guo Y, Barnes S J, Jia Q. Mining meaning from online ratings and reviews: tourist satisfaction analysis using latent dirichlet allocation. Tourism Manage, 2017, 59: 467–483
25 Hoffman M D, Blei D M, Wang C, et al. Stochastic variational inference. J Mach Learn Res, 2013, 14: 1303–1347
26 Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation. J Mach Learn Res, 2003, 3: 993–1022
27 Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in Python. J Mach Learn Res, 2011, 12: 2825–2830
28 Chen Z F, Ma W W Y, Lin W, et al. A study on the changes of dynamic feature code when fixing bugs: towards the benefits and costs of Python dynamic features. Sci China Inf Sci, 2018, 61: 012107
29 Chen L, Wu D, Ma W, et al. How C++ templates are used for generic programming. ACM Trans Softw Eng Methodol, 2020, 29: 1–49
30 Chen Z, Chen L, Ma W, et al. Understanding metric-based detectable smells in Python software: a comparative study. Inf Softw Tech, 2018, 94: 14–29
31 Guo Z, Li Y, Ma W, et al. Boosting crash-inducing change localization with rank-performance-based feature subset selection. Empir Softw Eng, 2020, 25: 1905–1950
32 Wang C, Li Y, Chen L, et al. Examining the effects of developer familiarity on bug fixing. J Syst Softw, 2020, 169: 110667
33 Nadi S, Krüger S, Mezini M, et al. Jumping through hoops: why do Java developers struggle with cryptography APIs? In: Proceedings of the 38th International Conference on Software Engineering, Hannover, 2017. 935–946

34 Pohlert T. Trend: non-parametric trend tests and change-point detection. 2018. R Package Version 1.1.1
35 Labovitz S. Criteria for selecting a significance level: a note on the sacredness of .05. The American Sociologist, 1968, 3: 220–222
36 Boslaugh S, Watters P A. Statistics in a Nutshell: a Desktop Quick Reference. Sebastopol: O'Reilly Media, 2008
37 Benesty J, Chen J, Huang Y, et al. Pearson Correlation Coefficient. Berlin: Springer, 2009
38 Beyer S, Pinzger M. A manual categorization of android app development issues on stack overflow. In: Proceedings of the 30th IEEE International Conference on Software Maintenance and Evolution, Victoria, 2014. 531–535
39 Han J, Shihab E, Wan Z, et al. What do programmers discuss about deep learning frameworks. Empir Softw Eng, 2020, 25: 2694–2747
40 Wan Z, Xia X, Hassan A E. What is discussed about blockchain? A case study on the use of balanced LDA and the reference architecture of a domain to capture online discussions about blockchain platforms across the stack exchange communities. IEEE Trans Softw Eng, 2019. doi: 10.1109/TSE.2019.2921343
41 Huang J, Peng M, Wang H, et al. A probabilistic method for emerging topic tracking in Microblog stream. World Wide Web, 2017, 20: 325–350
42 Zhu C, Zhu H, Ge Y, et al. Tracking the evolution of social emotions with topic models. Knowl Inf Syst, 2016, 47: 517–544
43 Xu T, Zhu H, Zhu C, et al. Measuring the popularity of job skills in recruitment market: a multi-criteria approach. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, 2018. 2572–2579