Empirical Software Engineering
https://doi.org/10.1007/s10664-020-09819-6

What do Programmers Discuss about Deep Learning Frameworks

Junxiao Han[1] · Emad Shihab[2] · Zhiyuan Wan[1] · Shuiguang Deng[1] · Xin Xia[3]

Published online: 24 April 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Deep learning has gained tremendous traction from the developer and researcher communities. It plays an increasingly significant role in a number of application domains. Deep learning frameworks are proposed to help developers and researchers easily leverage deep learning technologies, and they attract a great number of discussions on popular platforms, i.e., Stack Overflow and GitHub. To understand and compare the insights from these two platforms, we mine the topics of interests from these two platforms. Specifically, we apply Latent Dirichlet Allocation (LDA) topic modeling techniques to derive the discussion topics related to three popular deep learning frameworks, namely, Tensorflow, PyTorch and Theano. Within each platform, we compare the topics across the three deep learning frameworks. Moreover, we make a comparison of topics between the two platforms. Our observations include 1) a wide range of topics that are discussed about the three deep learning frameworks on both platforms, and the most popular workflow stages are Model Training and Preliminary Preparation. 2) the topic distributions at the workflow level and topic category level on Tensorflow and PyTorch are always similar while the topic distribution pattern on Theano is quite different. In addition, the topic trends at the workflow level and topic category level of the three deep learning frameworks are quite different. 3) the topics at the workflow level show different trends across the two platforms. e.g., the trend of the Preliminary Preparation stage topic on Stack Overflow comes to be relatively stable after 2016, while the trend of it on GitHub shows a stronger upward trend after 2016. Besides, the Model Training stage topic still achieves the highest impact scores across two platforms. Based on the findings, we also discuss implications for practitioners and researchers.

Keywords Empirical study · Deep learning frameworks · Tensorflow · Pytorch · Theano · LDA topic model

Communicated by: Filippo Lanubile

✉ Shuiguang Deng
dengsg@zju.edu.cn

Extended author information available on the last page of the article.

1 Introduction

Deep learning (Erickson et al. 2017) is an emerging branch of machine learning, and proposes a wide variety of neural network architectures to accomplish various tasks (Hinton et al. 2006; Krizhevsky et al. 2012). Deep learning technologies have a significant impact on several application domains (Erickson et al. 2017), e.g., computer vision (Krizhevsky et al. 2012; Russakovsky et al. 2015; Ledig et al. 2017), speech recognition (Ding et al. 2014; Hannun et al. 2014), nature language processing (Cai et al. 2017; Duan et al. 2018), machine translation (Weng et al. 2017; Hoang et al. 2017) and software engineering (Lee et al. 2017; Wan et al. 2018; Liu et al. 2018). The technologies significantly improve the performance of those application domains (Bahrampour et al. 2015) in comparison with the state of art.

Since deep learning is increasingly being used in software engineering domain (Lee et al. 2017; Wan et al. 2018; Liu et al. 2018), it has become more and more important for software engineering researchers. Researchers using these technologies can significantly improve the performance of their methods and in turn, this will also provide some ideal areas for researchers to perform further studies about deep learning. Deep learning frameworks are proposed to help practitioners and researchers better use deep learning technologies. The findings in this paper can give guidance to software engineering researchers in the research community, and also provide suggestions to practitioners to improve the development process and improve the user experience. The deep learning frameworks of Tensorflow, PyTorch and Theano are all typical in the deep learning domain. Tensorflow represents the most widely-deployed deep learning framework (Abadi et al. 2016). PyTorch is the framework with fastest increase trend recently (Ketkar 2017). And Theano is one of the oldest deep learning frameworks (Bergstra et al. 2010).

Those frameworks attract a great number of discussions on popular platforms, e.g., GitHub and Stack Overflow. GitHub provides more developer perspectives, while Stack Overflow provides more of a user's perspective. Both platforms play a significant role in improving the capabilities of software developers/users and accelerating software development (Mo et al. 2015). Analyzing on the two platforms will point out the topics of interests of the developers. There have existed many empirical studies concerning each of the two platforms (Treude et al. 2011; Allamanis and Sutton 2013; Bajaj et al. 2014; Barua et al. 2014; Rosen and Shihab 2016; Treude and Robillard 2016; Yang et al. 2016; Azad et al. 2017; Wang et al. 2018; Wan et al. 2019). These empirical studies are mainly related to topic trends on Stack Overflow, specific domain topics analysis, encumbrance in API usage, factors associated with question answering time and so on. Although there have existed many studies via Stack Overflow or GitHub, respectively, there are almost no empirical studies on deep learning frameworks using both the two platforms.

Besides, previous studies use Latent Dirichlet Allocation (LDA) topic modeling techniques for various researches, e.g., Li et al. (2018) and Rosen and Shihab (2016). Our study spans multiple frameworks in deep learning and the maturity of these frameworks are quite different. LDA blindly captures topics without considering the diversity of datasets and the domain-specific knowledge, where the derived LDA-topics are not linked to higher-level domain-specific concepts. Therefore, it can not be used merely in our case. Hence, we introduce the higher-level domain-specific workflow to link LDA-topics and make our analysis more comprehensive.

Specifically, we derive a domain-specific workflow and incorporate it with the LDA topic modeling techniques. First of all, we derive the generalized workflow for deep learning frameworks, which is composed of seven stages: Preliminary Preparation, Data Preparation,

Model Setup, Model Training, Model Evaluation, Model Tuning and Model Prediction. Then, we classify the dataset into six corpora according to the studied deep learning frameworks (i.e., Tensorflow, PyTorch and Theano) and the platforms (i.e., Stack Overflow and GitHub) and run LDA on each corpus independently. After that, we aggregate the LDA-topics within the workflow and derive many different categories for each workflow stage, which are topic categories. Furthermore, we analyze the impact trends of topics at different levels and make a comparison across the three studied deep learning frameworks. Our results show that the topic distribution patterns on Tensorflow and PyTorch remain the same while the topic distribution pattern on Theano is quite different. Moreover, we also compare the topics at different workflow levels between the two platforms, i.e., Stack Overflow and GitHub. The results yield some interesting insights and give some suggestions to developers, users, and researchers.

Our study answers the following research questions:

RQ1. What issues do practitioners discuss about deep learning frameworks?

A great variety of topics at different workflow stages are discussed for the three studied deep learning frameworks. Especially, we derive 75 LDA-topics from the six corpora in our dataset. We also observe that Model Training and Preliminary Preparation are the most frequently discussed workflow stages for all the six corpora. Meanwhile, the Preliminary Preparation stage for the corpus of Theano on GitHub takes up the largest amount of LDA-topics. On the contrary, the Model Tuning stage has not been discussed in all the six corpora, which means it is either not an issue or users are not concerned with this stage. Also, our results show that the top 3 LDA-topics with largest increases or largest decreases for the six corpora are quite different.

RQ2. What are the differences in the discussions across deep learning frameworks?

The topic distributions at the workflow level and topic category level on Tensorflow and PyTorch always keep the same pattern while the topic distribution pattern on Theano is quite different. For instance, the posts about Tensorflow and PyTorch on Stack Overflow are mainly concentrated on Preliminary Preparation, Data Preparation, Model Setup and Model Training workflow stages, while the vast majority of the posts about Theano on Stack Overflow fall into the Model Training stage. Nevertheless, the biggest similarity is that across the three deep learning frameworks, Model Training stage still accounts for the highest proportion.

In addition, we also find that only one LDA-topic has appeared twice on Tensorflow and Theano on Stack Overflow, which is *File Operation*. Surprisingly, it showed the decreasing trends of interests on Tensorflow, while showed the increasing trends of interests on Theano. Moreover, at the workflow level, we can observe that the overall impact trends of the workflow stages on Tensorflow and Theano are relatively flat, while the impact trends of the workflow stages on PyTorch fluctuates intensely.

RQ3. What are the differences in the discussions between Stack Overflow and GitHub?

We find that the topics at the workflow level show different trends across the two platforms. For instance, the Preliminary Preparation stage shows an increasing trend on the whole on both two platforms but the trend on Stack Overflow comes to be relatively stable after 2016, while the trend on GitHub shows a stronger upward trend after 2016. One potential explanation for this finding is that the developers with problems on Preliminary Preparation are apt to seek out answers on GitHub but not Stack Overflow after 2016. In addition, we also find that whether on Stack Overflow or GitHub, the Model Training stage still showed the highest impact in comparison with the other workflow stage topics during the study time period.

Our contributions are as follows:

1. We first perform a large-scale empirical study across the deep learning frameworks of Tensorflow, Pytorh and Theano both on Stack Overflow and GitHub platforms.
2. We derive the domain-specific workflow and aggregate the domain-specific topic categories for each workflow stage of the deep learning frameworks.
3. We compare the topic distributions at the workflow level and topic category level, we also compare the topic trends at the workflow level, topic category level and LDA-topic level across the studied deep learning frameworks and make some key insights and suggestions.
4. We compare the topics at different workflow stages across the two platforms and make some significant insights.

The rest of the paper is organized as follows. In Section 2, we describe the deep learning background. In Section 3, we introduce the research questions and methodology. The results of our empirical study are elaborated in Section 4 and Section 5 discusses the implications of this paper, LDA parameter analysis and the threats to validity. Section 6 presents the related work and Section 7 draws conclusions and introduces the future work. Last but not least, Section Appendix.

2 Background

Deep learning (DL), which is an artificial intelligence computational paradigm, is part of a broader family of machine learning methods based on learning data representations (Schmidhuber 2015; LeCun et al. 2015; Zhang et al. 2018).

Due to the popularity of deep learning, a number of deep learning frameworks have emerged, which play a significant role in carrying out deep learning projects. Deep learning frameworks can be differentiated based on the supporting languages, i.e., python-based framework, C++-based framework, lua-based framework and R-based framework, etc. The characteristics of different deep learning frameworks are illustrated in Table 1.

TensorFlow (Abadi et al. 2016) is an open source software library for numerical computation using data flow graphs, it was originally developed by researchers and engineers at the Google Brain team of Google Machine Intelligence Research organization for the purposes of conducting machine learning and deep neural network research. TensorFlow is the most widely-deployed deep learning framework, which was first released in November 2015. PyTorch (Ketkar 2017) is an open source machine learning library based on Torch, which is primarily developed by Facebook's artificial-intelligence research group. In this

Table 1 Characteristics of deep learning frameworks

Framework	Supporting languages
Tensorflow	Python
PyTorch	Python
Theano	Python C++
Torch	Lua Python
MXNet	Python R Julia Scala
Caffe	Python C++
CNTK	Python C++

regard, PyTorch is a Python package that provides two high-level features, i.e., tensor computation (like NumPy) with strong GPU acceleration and deep neural networks built on a tape-based autograd system. Need to know, PyTorch was first released in October 2016. Theano (Bergstra et al. 2010) is a Python library for fast numerical computation that can be run on the CPU or GPU, which is primarily developed by a machine learning group at the University of Montreal. As for the release time, Theano was first released in 2007, which is one of the oldest and most stable libraries. Torch (Collobert et al. 2011) is an open source machine learning library, a scientific computing framework, and a script language based on the Lua programming language. Earlier than Theano, torch was first released in October 2002. MXNet is a modern open-source deep learning framework used to train, and deploy deep neural networks, which is scalable and supports a flexible programming model and multiple languages (Li et al. 2014). As for caffe, it is a fast open framework for deep learning and CNTK is also an open source deep learning toolkit built by Microsoft, but the popularity and penetration rate of CNTK is not as good as Tensorflow, Theano and so on. Apart from the deep learning frameworks mentioned above, there also exist many other deep learning frameworks, such as caffe2,[1] chainer,[2] keras[3] and so on.

3 Research Settings

In this section, we first motivate and present our three research questions. Then, we introduce our research data and the collection process of our data sources. Lastly, we describe our research methodology, e.g., metrics used to analyze our research data.

3.1 Research Questions

RQ1. What issues do practitioners discuss about deep learning frameworks? In the past few years, the interest and use of deep learning frameworks have grown a lot, which has gained tremendous traction from deep learning practitioners and software engineering researchers. Identifying the major topics of different deep learning frameworks across different platforms can help find the major challenges that deep learning practitioners face and give some feasible suggestions to them. Besides, it can also help the research community understand the real issues that they face. Software engineering researchers thus can perform their researches on specific issues to help support and improve the development process of deep learning frameworks.

RQ2. What are the differences in the discussions across deep learning frameworks? Developers dedicated to different deep learning frameworks may face different problems, which may lead to differences in discussed topics across the deep learning frameworks. Therefore, we examine the distribution and evolution of issues that practitioners face for the three typical deep learning frameworks, i.e., Tensorflow, PyTorch, and Theano. Identifying and comparing the issues that practitioners face with different deep learning frameworks will give some guidance when we need to choose a proper deep learning framework to carry out deep learning related works. Moreover, it will

[1] https://caffe2.ai/
[2] https://chainer.org/
[3] https://keras.io/

help researchers better understand the issues for different deep learning frameworks and guide future research and development efforts for these frameworks, which in turn, help the deep learning community grow and mature.

RQ3. What are the differences in the discussions between Stack Overflow and GitHub? Users on Stack Overflow and developers on GitHub may encounter different problems. To figure out the similarities and differences between different platforms, we make a comparison between Stack Overflow and GitHub platforms. Results show that discussion topics of a specific deep learning framework on different platforms usually show different development trends. By analyzing the growth and decline trends of discussion topics on different platforms, developers dedicated to deep learning domain are capable of mastering the direction of deep learning problems, which can help a big deal with their development process. Moreover, the topics at the same workflow stages may have a different focus on different platforms, e.g., developers may tend to find answers on GitHub by submitting the pull requests or issues of bugs but not asking questions on Stack Overflow. Identifying the tendency of developers to find answers on the two different platforms will give some guidance when we need to choose a platform to solve the problems.

3.2 Research Data

We collected our dataset from two platforms: Stack Overflow and GitHub. Stack Overflow (Vasilescu et al. 2013) is a popular online programming question-answering community, which provides its developers with convenient access to knowledge and expertise of their peers. When developers face intractable problems during their development process, they may ask questions and seek answers from Stack Overflow (Mo et al. 2015). In the process of seeking questions and answers, if users consider a question or answer useful, he/she can "vote" it up, otherwise, "vote" it down. Based on the values received by the posts, users earn reputation score, which is the incentive system designed to encourage users to perform desirable activities (Wang et al. 2018). Many developers also share and learn open-source projects in social coding sites such as GitHub. GitHub (Yu et al. 2014) is a large and popular open source project platform, which has changed the way of development in a distributed collaborative manner and is widely adopted by software developers.

The Stack Overflow dataset is publicly available in XML format (Barua et al. 2014), which stores five XML documents: posts.xml, users.xml, comments.xml, votes.xml and badges.xml. Posts.xml stores all the information of posted questions and answers. e.g., the title and body of questions, the creation date, the tags that are associated with questions, etc. In this regard, we use Posts.xml to make our research. The downloaded Stack Overflow dataset was last updated in April 2018.

The GitHub dataset can be obtained via the GitHub API.[4] Through the GitHub API, we can crawl all the information that we need. e.g., the commit messages, pull request messages, issues, etc. For our purpose, we crawled all the pull requests and issues of projects tensorflow/tensorflow, pytorch/pytorch and theano/theano, which are the three framworks that we attempt to analyze. The GitHub dataset was downloaded in July 2018. Notably, both GitHub dataset and Stack Overflow dataset are all easy to obtain and analyze.

[4] https://api.github.com

Table 2 The statistics of the posts with different deep learning frameworks

Frameworks	Statistics of posts
Tensorflow	23,908
PyTorch	615
Theano	2,364

3.2.1 Stack Overflow Data Collection

A tag is a keyword or label that categorizes your question with other similar questions. Using the right tags makes it easier for others to find and answer your question. Also, tags can be used to search and browse related issues. On Stack Overflow, each question post can have up to five tags and must have at least one tag.

During the process of data collection on Stack Overflow, we leverage the tags of "tensorflow", "pytorch" and "theano" to obtain the question posts of Tensorflow, PyTorch and Theano frameworks, respectively. Consequently, we extracted 23,908 question posts related to Tensorflow, 615 question posts related to PyTorch and 2,364 question posts related to Theano. The statistics of the posts are shown in Table 2.

After that, we transform the posts into records for each deep learning framework and store it in a MongoDB database with three collections. For that every record includes the textual content (e.g., the title, the body and the tags) and the information of metadata (e.g., the identifier, the timestamp and the viewcount). The records of the three deep learning frameworks make up three Stack Overflow corpora which are used in the rest of the experiments.

3.2.2 GitHub Data Collection

Different deep learning frameworks have the open source projects on GitHub, for the frameworks of Tensorflow, PyTorch and Theano, their open source projects on GitHub are tensorflow/tensorflow,[5] pytorch/pytorch[6] and theano/theano,[7] respectively. To collect dataset from GitHub, we crawled all the pull requests and the issues of the three projects aforementioned. As a result, we obtained 7,987 pull requests and 12,400 issues from the project of tensorflow/tensorflow, 4,888 pull requests and 4,400 issues from the project of pytorch/pytorch, 4,055 pull requests and 2,600 issues from the project of theano/theano, respectively. All the data were collected between November 2015 and July 2018.

For every project, we then merged the pull requests and the issues to filter out the duplicate records. The duplicate records are defined as the issues and pull requests are with the same content (the same number, the same title, and the same body, etc.). Since the pull requests and issues all have an unique number, we thus filter the duplicate records like this: if a pull request has the same number with an issue, we remove the pull request. After that, 15,333 records remained for the project of tensorflow/tensorflow, 6,736 records remained for the project of pytorch/pytorch and 5,354 records remained for the project of theano/theano. We would like to emphasize that the remained records may have different states, e.g., "open" state or "closed" state. Since the closed records have completed and contain more useful information, we then do a filter to remove the open records and leave only

[5] https://github.com/tensorflow/tensorflow
[6] https://github.com/pytorch/pytorch
[7] https://github.com/theano/theano

Table 3 The statistics of the dataset with different deep learning frameworks on GitHub

Frameworks	Statistics of pull requests	Statistics of issues	Statistics of pull requests and issues	Statistics of deduplicate records	Statistics of closed records
Tensorflow	7,987	12,400	20,387	15,333	13,666
PyTorch	4,888	4,400	9,288	6,736	5,753
Theano	4,055	2,600	6,655	5,354	4,977

the closed records. In the end, the Tensorflow dataset on GitHub is composed of 13,666 records, the PyTorch dataset on GitHub is composed of 5,753 records and the Theano dataset on GitHub is composed of 4,977 records, which make up our GitHub dataset that will be used in the rest of the experiments. The statistics of the dataset are shown in Table 3.

Note that all the dataset of different deep learning frameworks on GitHub are obtained via the GitHub API. Through the GitHub API, we extract the information of pull requests and issues for every deep learning frameworks, i.e., the number, the title, the body, the state and the creation time. After extracting the information that we needed, we then transform the pull requests or issues into records and store it in the MongoDB database with three collections. As a result, we obtain three GitHub corpora which are used in the rest of the experiments.

3.3 Research Methodology

Our research methodology is composed of five steps, that are shown in Fig. 1. In the five steps, we first introduce the process of data preprocessing, then, we present the generalized workflow of deep learning frameworks. After that, we use topic model to find LDA-topics and aggregate the generated LDA-topics into different topic categories in different workflow stages. Lastly, we elaborate the metrics and analysis. Each step is detailed in the following subsections.

3.3.1 Data Preprocessing

To filter out the noisy information in the textual content of the extracted Stack Overflow dataset and GitHub dataset, we perform data preprocessing in five steps.

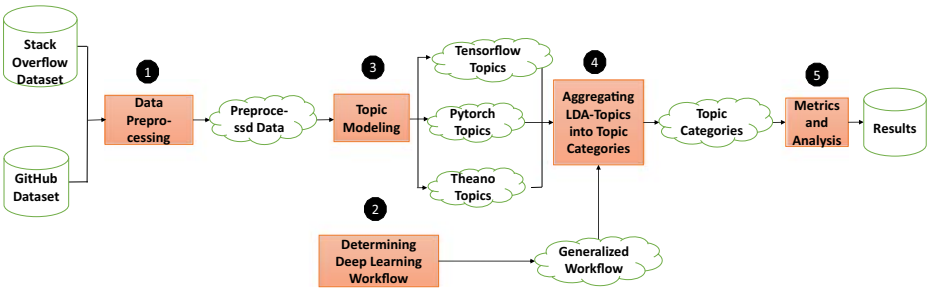


Fig. 1 An overview of research methodology

Step 1: Remove code snippets. Since posts on Stack Overflow are often accompanied by code snippets and the code snippets usually contain programming language syntax and keywords, which may lead to bad topic modeling results and introduce noises into the future analysis (Barua et al. 2014). Furthermore, since most of the source code on Stack Overflow only consists of small segments, there is not sufficient context to allow meaningful content to be extracted from the code snippets (Barua et al. 2014). Therefore, we remove the code snippets that are enclosed in `<code>` HTML tags of Stack Overflow dataset, where the adopted preprocessing approach is the same as previous studies (Barua et al. 2014; Yang et al. 2016). As for the GitHub dataset, we do not carry out this approach for the reason that it does not contain HTML tags of `<code>`. Specifically, we search for the backquote " ` " punctuation in the records of GitHub dataset, and remove the code wrapped between the backquote. The rationale is that the writing of issues and pull requests in GitHub basically follows the markdown syntax, and in the markdown syntax, the code is wrapped with a backquote. Since the code snippets include the source code and the code comments, after removing the code snippets, the source code and the code comments are all excluded.

Step 2: Remove HTML tags and URLs. Owing to that the HTML tags (e.g., `<p>`, `<pre>` and `<b>`) and URLs are useless in the process of topic modeling, we remove all the HTML tags and URLs of both Stack Overflow dataset and GitHub dataset.

Step 3: Remove numbers, punctuation marks and other non-alphabetic characters. For better topic modeling results, we continue to remove the number, the punctuation marks and other non-alphabetic characters of both Stack Overflow dataset and GitHub dataset.

Step 4: Lemmatize. After step 3, we apply the NLTK WordNet Lemmatizer (Miller 1995) to transform words to their base forms (e.g., "makes" transformed to "make").

Step 5: Filter out stop words. Finally, for that common English-language stop words do not work for generating meaningful topics (Schütze et al. 2008), we filter out it by using the NLTK stop words corpus (Loper and Bird 2002).

3.3.2 Determining Deep Learning Workflow

The workflow of a specific domain defines the basic stages of the domain and the relationships between these stages. For deep learning domain, there also exists a workflow. Starting from the architecture documentation of the various deep learning frameworks, e.g., Tensorflow, PyTorch, and Theano, we develop the generalized workflow for the deep learning domain with different deep learning frameworks. The generated workflow is composed of seven stages, which can be seen in Fig. 2. Next, we elaborate the stages in detail.

1) **Preliminary preparation.** Before using deep learning frameworks, we first need to solve the installation problem, platform compatibility problem, version problem and so on. All these problems pave the way for the subsequent use of deep learning frameworks.
2) **Data preparation.** This category is a prerequisite for all deep learning frameworks for that it is necessary to prepare data for different models. Only by converting the raw data into the suitable input data format required by the deep learning frameworks can the models be at the best performance. Most deep learning frameworks provide this functionality.

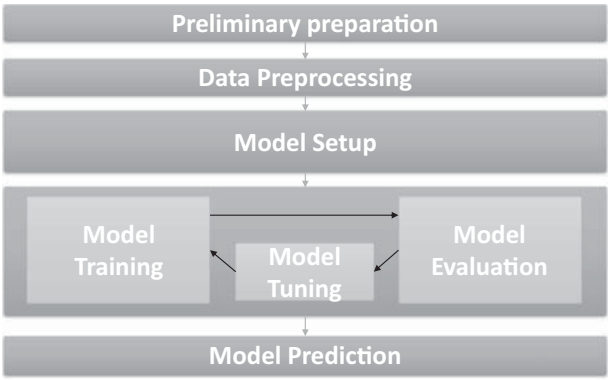


Fig. 2 The Workflow of Deep Learning Frameworks

3) **Model setup.** Model Setup category is one of the most crucial categories in all deep learning frameworks. This category is dedicated to the problems with the model itself, e.g., model selection, model setup, model loading and model saving.
4) **Model training.** After selecting a suitable model, it is time to train the model to generate better model accuracy and performance. It is worth mentioning that there are a lot of techniques on model training, e.g., the model parameter selection, the loss funtion selection, optimization strategy, gpu acceleration.
5) **Model evaluation.** The category of model evaluation is responsible for evaluating the trained model generated in the previous step. Sometimes, it needs to visualize the trained model, so that the developers can have a better understanding of the model training process and easily assess the changes in loss function, model accuracy and so on.
6) **Model tuning.** The model's hyper-parameter tuning is responsible for improving the model's performance and accuracy. An inappropriate loss function may lead to a poor model and bring about a low model accuracy, which explains the importance of hyper-parameter tuning.
7) **Model prediction.** After training and evaluating the model, it is time to use the trained model to predict the new input data. During the process of model prediction, we also need to pay attention to the prediction accuracy and other problems appeared in stage.

3.3.3 Topic Modeling

Our goal involves finding the discussion topics in different deep learning frameworks and our dataset is derived from both Stack Overflow platform and GitHub platform. Due to that the tags on Stack Overflow are coarse-grained and fail to provide fine-grained categorization information and the GitHub dataset includes no tags to provide categorization information, we use LDA-based topic models to discover and capture the fine-grained discussion topics in different deep learning frameworks.

LDA is a probabilistic topic model, which represents topics as probability distributions over the words in the corpus of text. Moreover, it also represents the documents as the probability distributions of the discovered topics. LDA uses the word frequencies and word co-occurrences in the corpus of the documents to build a model of related words (Thomas 2012). The words in a referenced topic are usually semantically related, overall, which gives

a special implication to the referenced topic. However, due to the fact that the LDA model has no semantic knowledge, the meanings of the referenced topics must be determined by manually examining the set of words. For example, the words "model", "tensorflow", "trained", "file" and "save" presented in a topic indicate that this topic may related to model saving on Tensorflow.

Due to that the topics generated by LDA are less likely to be overfitting and are easier to interpret, LDA has been recognized as the widely used technique for generating discussion topics of unstructured documents (Blei et al. 2012). Thus, a tremendous amount of researches have emerged in Software Engineering domain, including extracting topics from bug reports (Lukins et al. 2008), analyzing topics on Stack Overflow (Barua et al. 2014; Rosen and Shihab 2016; Yang et al. 2016) and studying software loggings (Li et al. 2018).

LDA implementation. We use the implementation of the LDA model provided by the gensim Python library,[8] which is an implementation of the online LDA by Hoffman et al. (2010) - an online variational Bayes (VB) algorithm for LDA. We use 5 passes (the number of passes through the corpus during training) with K topics to run the gensim LDA and all other parameters have default settings.

Number of Topics. The number of topics, denoted as K, is usually a parameter specified by user, which can control the granularity of the discovered topics (Barua et al. 2014). The value of K can neither be too large nor too small, too large of the value of K may result in topic crossover and redundancy, while too small of the value of K may lead to coarser-grained topics (Rosen and Shihab 2016). We experimented with different K values to obtain the topics with the highest coherence measures. Here, the coherence measures are used to quantify the coherence of fact sets (Bovens and Hartmann 2010), which can also be used to assess the comprehensibility of the discovered topics (Newman et al. 2010).

To better analyze the discussion topics in different deep learning frameworks, we run one LDA per corpus to capture the discussion topics in our dataset. Our dataset are composed of six corpora: 23,908 posts on Stack Overflow in Tensorflow corpus, 615 posts on Stack Overflow in PyTorch corpus, 2,364 posts on Stack Overflow in Theano corpus, 13,666 records on GitHub in Tensorflow corpus, 5,753 records on GitHub in PyTorch corpus and 4,977 records on GitHub in Theano corpus. We generate different LDA models for each corpus to obtain the discussion topics for each corpus in our dataset. Thus, we can use different number of K values for each LDA models along with each corpus according to its richness of the content.

For each corpus, to choose the proper K value that derives topics with high coherence score, we set K range from 5 to 50 with a step of 5 (i.e., 5, 10, 15, 20, ..., 50). We calculate the topic coherence for each LDA model by applying the gensim module **CoherenceModel**, which has implemented the Roder et al.'s four stage topic coherence pipeline (Both and Hinneburg 2015).

3.3.4 Aggregating the Generated LDA-topics into Topic Categories

Different deep learning frameworks implement the aforementioned workflow in different ways. To study the discussion issues of different deep learning frameworks on different platforms (i.e., Stack Overflow and GitHub) in more detail, the first author and a graduate

[8] https://radimrehurek.com/gensim/

student aggregate the generated LDA-topics to each workflow stage and derive the aggregated topic categories in each workflow stage. We manually look into the LDA-topics and categorize them to each workflow stage. Then we apply the card sorting approach (Spencer 2009; Wan et al. 2017) to derive the aggregated topic categories in each workflow stage. The aggregated topic categories in each workflow stage for different deep learning frameworks on different platforms are shown in Fig. 3 and Table 4. From Fig. 3 and Table 4, we have the following observations:

Preliminary preparation. Tensorflow and PyTorch on GitHub both discuss the *System Installation*, while Tensorflow, PyTorch and Theano on GitHub all raise the *Version Problem*. Tensorflow on Stack Overflow, PyTorch on GitHub and Theano on GitHub refer to *Other API*, and all the studied deep learning frameworks on the two platforms involve *Error*.

Data preparation. Tensorflow and PyTorch on Stack Overflow are all related to the *Data Reading* problems, Tensorflow and Theano on GitHub match the *Data Type* problem. Moreover, Tensorflow on Stack Overflow, Tensorflow, PyTorch and Theano on GitHub all refer to the *Data Shape* problem. Different from the former, Tensorflow on Stack Overflow, PyTorch on Stack Overflow and GitHub, and Theano on GitHub fall into the category of *Error*, which pays attention to the errors that appear in this stage.

Model setup. Tensorflow and PyTorch on Stack Overflow all pay close attention to the *Model Application* and *Model Structure*, and the *Model Saving* problem is followed by Tensorflow on Stack Overflow and Theano on GitHub.

Model training. The process of model training is composed of many categories. In these categories, many techniques are discussed and studied, e.g., the loss funtion, optimization

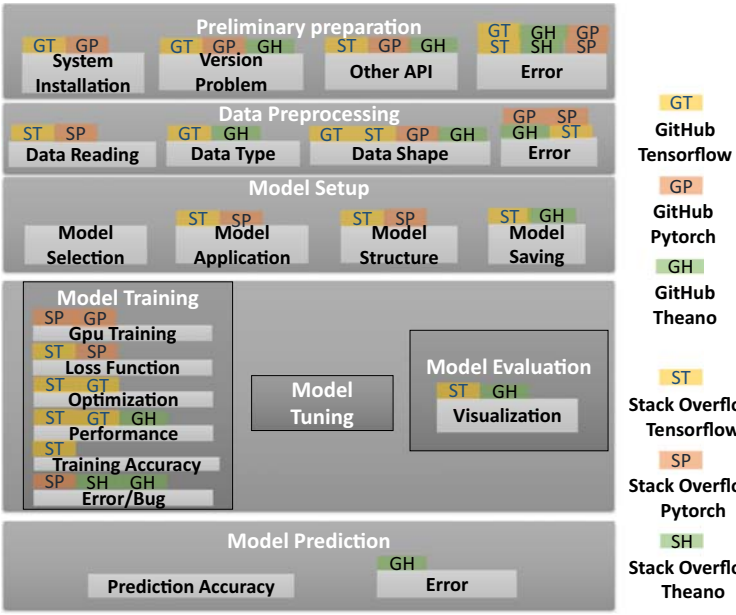


Fig. 3 The Aggregated Topic Categories of the Workflow for Different Deep Learning Frameworks on Different Platforms

Table 4 The Aggregated Topic Categories of the Workflow for Different Deep Learning Frameworks on Different Platforms

Topics categories	SO Tensorflow	SO PyTorch	SO Theano	GH Tensorflow	GH PyTorch	GH Theano
Preliminary Preparation						
System Installation				✓	✓	
Version Problem				✓	✓	✓
Other API	✓				✓	✓
Error	✓	✓	✓	✓	✓	✓
Data Preparation						
Data Reading	✓	✓				
Data Type				✓		✓
Data Shape	✓			✓	✓	✓
Error	✓	✓			✓	✓
Model Setup						
Model Selection						
Model Application	✓	✓				
Model Structure	✓	✓				
Model Saving	✓					✓
Model Training						
Gpu Training		✓			✓	
Loss Function	✓	✓				
Optimization	✓			✓		
Performance	✓			✓		✓
Training Accuracy	✓					
Error/Bug		✓	✓			✓
Model Evaluation						
Visualization	✓					✓
Model Tuning						
Model Prediction						
Prediction Accuracy						
Error						✓

"✓" indicates the distribution of topic categories at different deep learning frameworks on different platforms. SO is the abbreviation of Stack Overflow and GH is the abbreviation of GitHub

strategy and gpu training. Moreover, the model performance also falls into the process of model training. Also, there may exist a lot of errors in the training process.

PyTorch on Stack Overflow and GitHub are concerned about the issues of *Gpu Training*, *Loss Funtion* and *Error/Bug*. Tensorflow on Stack Overflow and GitHub pay attention to the issues of *Loss Funtion*, *Optimization*, *Performance* and *Training Accuracy*. Theano on Stack Overflow and GitHub refer to the issues of *Performance* and *Error/Bug*. Interestingly, only PyTorch on Stack Overflow and GitHub are concerned about the issue of *Gpu Training* and only Tensorflow on Stack Overflow and GitHub pay attention to the issues of *Optimization* and *Training Accuracy*.

Model evaluation. In the process of model evaluation, there is only one implementation, which is the issue of *Visualization* concerned by Tensorflow on Stack Overflow and Theano on GitHub.

Model tuning. Model tuning, which is also the hyper-parameter tuning, is significant in improving the model's performance and accuracy. However, there are no studied frameworks related to model tuning, currently.

Model prediction. Only Theano on GitHub is associated with the issue of *Error*.

3.3.5 Metrics and Analysis

We apply LDA to each post/record in our dataset. Thus, each post/record derives K topics, $z_1, z_2, ..., z_k$. We define the topic membership value of topic z_i in post/record d_j as $\theta(d_j, z_i)$. It is worth emphasizing that $\forall i, k : 0 \leq \theta(d_i, z_k) \leq 1$ and $\forall i : \sum_1^k \theta(d_i, z_k) = 1$. Under such conditions, we carry out computations on the following metrics to answer our research questions.

Dominant Topic The topic with the highest probability value is the dominant topic for each post/record. The dominant topic of a post/record d_j is denoted as

$$dominant(d_j) = z_i : \theta(d_j, z_i) = max(\theta(d_j, z_k)), 1 \leq j \leq K \tag{1}$$

Topic Trends over Time To make an analysis on the temporal trends of topics, we introduce the *impact* metric of a topic z_i in month m, which is defined in Barua et al. (2014):

$$impact(z_i, m) = \frac{1}{|D(m)|} \sum_{d_j \in D(m)} \theta(d_j, z_i) \tag{2}$$

where $D(m)$ is the set of posts/records in month m. The *impact* metric is used to measure the relative proportion of posts/records associated with one specific topic z_i compared to the other topics in that particular month m.

In addition, we also define the *impact* metric of topics at a specific workflow stage S in month m, which is:

$$impact(S, m) = \frac{1}{|D(m)|} \sum_{d_j \in D(m), z_i \in S} \theta(d_j, z_i) \tag{3}$$

where $D(m)$ is the set of posts/records in month m.

4 Results

In this section, we present the results of applying the research methodology to our research dataset and answer of each research question. It is worth noting that the topics discussed in our paper consist of three levels, namely workflow stage (seven stages for the generalized workflow), LDA-topic (the topics derived from LDA models directly), and topic category (the aggregated topic categories in each workflow stage). We will then use the three terms to analyze the distribution and evolution of topics at different levels.

4.1 RQ1. What issues do practitioners discuss about deep learning frameworks?

As described above, we generated different LDA models for each corpus and chose the optimal K with the highest coherence score for each corpus respectively. As a result, we derive 75 LDA-topics from the six corpora in our dataset, 20 LDA-topics for the corpus of Tensorflow on Stack Overflow, 10 LDA-topics for the corpus of PyTorch on Stack Overflow, 5 LDA-topics for the corpus of Theano on Stack Overflow, 10 LDA-topics for the corpus of Tensorflow on GitHub, 10 LDA-topics for the corpus of PyTorch on GitHub, and 20 LDA-topics for the corpus of Theano on GitHub. Tables 7, 8, 9, 10, 11 and 12 illustrate the details of the discovered LDA-topics and their top keywords, which can be seen in Appendix A. Figures 17, 18, 19, 20, 21 and 22 shows the trendline of the top 3 LDA-topics with the largest increases and decreases, which can be seen in Appendix B. These trendlines give explanations of the rise and fall of interest on specific LDA-topics.

The topic categories aggregated by our methodology are shown in Table 5. From Table 5, we can observe the workflow stages, the aggregated topic categories, the number of LDA-topics matched to a specific topic category and the proportion of posts/records in a specific topic category within a corpus. Then, we present the relationships between the workflow and the LDA-topics and give a subset of representative example posts/records for each LDA-topic, which can be seen in Appendix C.

> Out of the 75 LDA-topics, preliminary preparation stage, data preparation stage, model setup stage, model training stage, model evaluation stage, model tuning stage, and model prediction stage account for 29, 13, 8, 22, 2, 0, 1 LDA-topics, respectively.

Furthermore, we can gain some key insights from Table 5. From Table 5, we can observe that the most popular workflow stages are *Model Training* and *Preliminary Preparation*. Out of all the six corpora, they all make frequent discussions on the two stages and the two stages all account for a large proportion of posts/records. Meanwhile, we can find that the *Preliminary Preparation* stage for the corpus of Theano on GitHub takes up the largest amount of LDA-topics, where the largest number is 9. We can also see that the *Model Tuning* stage has not been discussed in all the six corpora, which is basically in line with our normal knowledge. As we all know, deep learning frameworks only have a few parameters to be debugged (e.g., learning rate and drop-out). The debugging of parameters is mostly based on practitioners' experience and are determined by constantly trying, thus there exist no issues about the *Model Tuning* stage. Moreover, at the level of topic categories, results show that only the topic category of *Error* makes discussions in all the 6 corpora. Besides, the largest amount of LDA-topics also belong to the topic category of *Error* for the corpus of Theano on GitHub, which is consistent with the results in workflow topic level.

> Our findings show that the most popular workflow stages are *Model Training* and *preliminary preparation* for the six corpora, while the *Model Tuning* stage has not been discussed in all the six corpora, which is basically in line with our normal knowledge. Due to the fact that the debugging of parameters are mostly based on practitioners' experience and are determined by constantly trying, thus there exist no issues about it. Besides, at the topic category level, only the topic category of *Error* makes discussions in all the six corpora. Moreover, when it comes to the trends of LDA-topics, results show that the LDA-topics with largest increases or largest decreases at different corpus are quite different on the whole.

4.2 RQ2. What are the Differences in the Discussions Across Deep Learning Frameworks?

In this section, we make an analysis on the topics across different deep learning frameworks within a specific platform, aiming to find out the similarities or differences across different deep learning frameworks. We first elaborate on the topic distributions of different deep learning frameworks, which can be seen in Table 5. Then, we can gain some interesting insights from the topic trends at the LDA-topic level and topic category level across different deep learning frameworks. In the end, we make some comparisons about the topic

Table 5 The percentage of posts/records in the aggregated topic categories for the six corpora

Topics	SO Tensorflow	SO PyTorch	SO Theano	GH Tensorflow	GH PyTorch	GH Theano
Preliminary Preparation	22.1% ×5	17.1% ×2	16.1% ×2	30.8% ×4	34.4% ×5	20.2% ×9
System Installation				7.3% ×1	3.2% ×1	
Version Problem				3.7% ×1	7.7% ×1	5.0% ×1
Other API	4.0% ×1				1.2% ×1	3.4% ×1
Error	16.0% ×3	17.1% ×2	11.8% ×1	16.5% ×2	20.4% ×2	9.3% ×6
Data Preparation	23.3% ×4	17.5% ×2		6.9% ×2	31.3% ×2	10.5% ×3
Data Reading	8.1% ×2	5.0% ×1				
Data Type				2.7% ×1		4.2% ×1
Data Shape	7.4% ×1			4.2% ×1	14.1% ×1	4.8% ×1
Error	7.7% ×1	12.5% ×1			17.2% ×1	1.5% ×1
Model Setup	24.8% ×3	22.0% ×2				0.8% ×1
Model Selection						
Model Application	13.9% ×3	10.6% ×1				
Model Structure	5.4% ×1	11.4% ×1				
Model Saving	5.5% ×1					0.8% ×1
Model Training	25.5% ×5	43.4% ×4	83.9% ×3	62.3% ×3	34.3% ×2	46.4% ×5
Gpu Training		10.9% ×1			3.2% ×1	
Loss Function	5.3% ×1	13.2% ×1				
Optimization	3.4% ×1			47.8% ×1		
Performance	5.5% ×1			9.0% ×1		7.5% ×1
Training Accuracy	6.5% ×1					
Error/Bug		9.4% ×1	27.6% ×1			36.5% ×3
Model Evaluation	4.3% ×1					9.2% ×1
Visualization	4.3% ×1					9.2% ×1
Model Tuning						
Model Prediction						12.9% ×1
Prediction Accuracy						
Error						12.9% ×1

"× number" indicates the number of LDA-topics matched with a particular topic category. SO is the abbreviation of Stack Overflow and GH is the abbreviation of GitHub

trends at the workflow level. Notably, since there exist two platforms (i.e., Stack Overflow and GitHub), we thus compare the studied deep learning frameworks on each platform separately.

4.2.1 Stack Overflow Platform

We first compare the deep learning frameworks on the Stack Overflow platform. The comparison process includes the following aspects: topic distribution pattern at the workflow level, topic distribution pattern at the topic category level, LDA-topic trend, and topic category trend.

Topic Distribution Pattern at the Workflow Level Firstly, we analyze the topic distributions at the workflow level.

- **Tensorflow.** As shown in Table 5, the posts about Tensorflow are mainly concentrated on *Preliminary Preparation* (22.1%), *Data Preparation* (23.3%), *Model Setup* (24.8%) and *Model Training* (25.5%). Especially, the *Model Training* stage accompanied by the highest proportion as opposed to the *Model Evaluation* (4.3%) stage with the lowest proportion. Furthermore, the *Model Tuning* and *Model Prediction* stages have no proportions.
- **PyTorch.** Similarly, most of the posts about PyTorch fall into *Preliminary Preparation* (17.1%), *Data Preparation* (17.5%), *Model Setup* (22.0%) and *Model Training* (43.4%). Particularly, the *Model Training* stage accounts for 43.4%, which is nearly half of the whole and is consistent with the conclusion in Tensorflow. Also, there exist no proportions on the *Model Evaluation*, *Model Tuning* and *Model Prediction* stages.
- **Theano.** As for Theano, the vast majority of the posts fall into the *Model Training* stage, which accounts for 83.9%. The rest of the posts account for 16.1% on *Preliminary Preparation*. Consistent with the previous conclusion, the *Model Training* stage also takes up the highest proportion on the whole.

> The topic distributions at the workflow level on Tensorflow and PyTorch always keep the same pattern while the topic distribution pattern on Theano is quite different. Tensorflow and PyTorch are mainly concentrated on *Preliminary Preparation*, *Data Preparation*, *Model Setup* and *Model Training*, while Theano mainly fall into the *Model Training* stage.
> Nevertheless, the biggest similarity is that across the three deep learning frameworks, *Model Training* stage still accounts for the highest proportion. In this regard, we can speculate that the most practitioners suffer from confusions and doubts or have errors to be solved on this stage, which indicates that more attention should be paied for this stage.

Topic Distribution Pattern at the Topic Category Level Then, we further analyze the topic distributions in terms of different topic categories.

- **Tensorflow.** For Tensorflow, although the *Model Training* stage accounts for the highest proportion, the topic category of *Error* (16.0%) in *Preliminary Preparation* stage accounts for the highest proportion compared to the other topic categories. Besides, the *Error* topic is composed of 3 LDA-topics, which is also the largest amount in all the topic categories. The proportions of the rest topic categories vary from 3.4% to 13.9%.

- **PyTorch.** For PyTorch, which is analogous to Tensorflow, the highest proportion and the largest amount of 2 LDA-topics also belong to the topic category of *Error* (17.1%) in *Preliminary Preparation*. While the *Data Reading* (5.0%) topic category subordinates to the smallest proportion compared to the other topic categories.
- **Theano.** As for Theano, the majority of the posts divide into the *Model Training* stage, and the majority of the proportion fall into *Error/Bug* (27.6%) in *Model Training* stage, which is different from the former frameworks.

The topic distribution at the topic category level on Tensorflow and PyTorch keep the same pattern to some extent, they focus more on the topic category of *Error* in *Preliminary Preparation* stage. While the topic distribution pattern on Theano is quite different, they focus more on the topic category of *Error/Bug* in *Model Training* stage. Although their concerns are quite different, it indicates that there still need attention and improvements to the errors in the two stages.

LDA-topic Trend Except the difference of topic distributions in terms of workflow level or topic category level, we can also observe the discrepancy between different deep learning frameworks from the top 3 LDA-topics with the largest increases or decreases in Figs. 17, 19 and 21. From Figs. 17, 19 and 21, we can draw a conclusion that whether on Tensorflow, PyTorch or Theano, the top 3 LDA-topics with largest increases or decreases are always different, which means that the users using different deep learning frameworks on Stack Overflow show different concerns and interests. It is worth mentioning that in all the top 3 LDA-topics with the largest increases or decreases of Tensorflow, PyTorch and Theano, only one LDA-topic has appeared twice, which is *File Operation*. Surprisingly, it showed the decreasing trends of interests on Tensorflow, while showed the increasing trends of interests on Theano. A proper explanation for this trend may be attributed to the fact that Tensorflow has drawn more and more attention from users and the tutorials or answers towards *File Operation* are becoming more and more complete, which results in the decreasing trends of interests. However, although Theano is a veteran deep learning framework, there are still many places to be completed.

The top 3 LDA-topics with largest increases or decreases are always different on the three studied deep learning frameworks, which implies that users show different concerns and interests on different deep learning frameworks. Surprisingly, only one LDA-topic has appeared twice and shown different trend on different deep learning frameworks.

Topic Category Trend Moreover, due to the fact that the three frameworks all include the topic category of *Error* in *Preliminary Preparation* stage, we take it as an example to explore the similarities and differences of the topic category trend across different frameworks, which is illustrated in Fig. 4. We find that all the three deep learning frameworks showed a very high impact at the very beginning during the study time period. A reasonable explanation for this trend can be attributed to the lack of tutorials and the insufficiency of documentation system, which results in a sharp increase on *Error* at the beginning. On this condition, we can improve the documentation system and provide more exhaustive tutorials to ameliorate this problem. Moreover, after the abnormal peak at the very beginning, the trend of *Error* at all the three deep learning frameworks come to be steady and smooth with

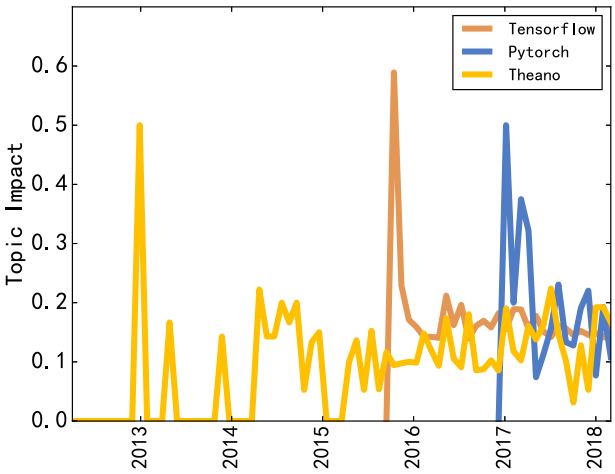


Fig. 4 Comparative trend analysis of the *Error* topic category in *Preliminary Preparation* stage across deep learning frameworks

impact scores vary from 0 to 0.2. In general, there exists no obvious monotonic trend for *Error* at all the three deep learning frameworks.

The topic category of *Error* showed a very high impact at the very beginning during the study time period, which can be attributed to the lack of tutorials and the insufficiency of documentation system at the beginning. On this condition, we can improve the documentation system and provide more exhaustive tutorials to ameliorate this problem.

4.2.2 GitHub Platform

We then compare the deep learning frameworks on the GitHub platform. The comparison process includes the following aspects: topic distribution pattern at the workflow level, topic distribution pattern at the topic category level, LDA-topic trend, and topic category trend.

Topic Distribution Pattern at the Workflow Level The topic distributions of Tensorflow, PyTorch and Theano on GitHub can be seen in Table 5.

- **Tensorflow.** From Table 5, we can speculate that the posts about Tensorflow are mainly concentrated on *Preliminary Preparation* (30.8%), *Data Preparation* (6.9%) and *Model Training* (62.3%), where the *Model Training* stage accounts for the highest proportion as opposed to the *Data Preparation* stage with the lowest proportion. Besides, no proportions are assigned for the other 4 stages.
- **PyTorch.** When it comes to PyTorch, the most of the posts fall into *Preliminary Preparation* (34.4%), *Data Preparation* (31.3%) and *Model Training* (34.3%), which presents the same distribution pattern as Tensorflow. Furthermore, the other 4 stages also have no proportions. It is worth mentioning that the proportion distribution on the three workflow stages in PyTorch is relatively equal, which expresses discrepancy with the proportion distribution in Tensorflow.

– **Theano.** In addition, we can speculate that Theano wrap up six of the seven workflow stages, which are: *Preliminary Preparation* (20.2%), *Data Preparation* (10.5%), *Model Setup* (0.8%), *Model Training* (46.4%), *Model Evaluation* (9.2%) and *Model Prediction* (12.9%). Among them, the *Model Training* stage still accounted for the highest proportion, which is consistent with the conclusion in Tensorflow. On this condition, we can draw a conclusion that more useful tools and thorough tutorials should be provided to help developers solving the problems emerged in the *Model Training* stage.

The topic distribution at the workflow level on Tensorflow and PyTorch presents the same pattern, they all fall into *Preliminary Preparation*, *Data Preparation* and *Model Training*. While Theano shows a quite different distribution pattern, they account for six of the seven workflow stages. However, the distribution ratios of topics at the workflow level for different deep learning frameworks are slightly different.

Topic Distribution Pattern at the Topic Category Level Furthermore, we compare the topic distributions of topic categories across the three different deep learning frameworks.

– **Tensorflow.** In Tensorflow, the topic category of *Optimization* (47.8%) in *Model Training* stage occupies nearly half of the whole, which is aligned with the highest proportion of *Model Training* stage. Although *Optimization* topic category takes up the highest proportion compared to the other topic categories, it only consists of 1 LDA-topic. On the contrary, the topic categories of *Data Type*, *Version Problem* and *Data Shape* take up 2.7%, 3.7% and 4.2%, respectively.
– **PyTorch.** Different from Tensorflow, the highest proportion of the topic category in PyTorch is *Error* (20.4%) in *Preliminary Preparation*. Besides, the second highest proportion of the topic category is *Error* (17.2%) in *Data Preparation* stage. The *Error* in *Preliminary Preparation* is composed of 2 LDA-topics while the *Error* in *Data Preparation* only includes 1 LDA-topic.
– **Theano.** Lastly, the proportions of the topic categories in Theano are relatively equal. The *Error* (36.5%) in *Model Training* stage accounts for the highest proportion with 3 LDA-topics while the *Error* (9.3%) in *Preliminary Preparation* stage takes up the largest amount of 6 LDA-topics.

Tensorflow focuses more on the topic category of *Optimization*, while the highest proportion of the topic category in PyTorch is *Error* in *Preliminary Preparation* and *Error* in *Data Preparation*. Simultaneously, the topic category of *Error* in *Model Training* stage accounts for the highest proportion in Theano. Results show that different deep learning frameworks on GitHub show different discussions and concerns on the topics at the topic category level.

LDA-topic Trend After the comparison of the topic distributions at different levels, we then compare the top 3 LDA-topics with the largest increases or decreases on different deep learning frameworks, which can be seen in Figs. 18, 20 and 22. We can observe that out of all the top 3 LDA-topics with the largest increases or decreases of Tensorflow, PyTorch and Theano on GitHub, only two LDA-topics have appeared twice, which are *Version Problem* and *Performance*. The LDA-topic of *Version Problem* presents the increasing trend both on

PyTorch and Theano, which implies that the two deep learning frameworks all need to better solve the version problems and improve the compatibility between different versions.

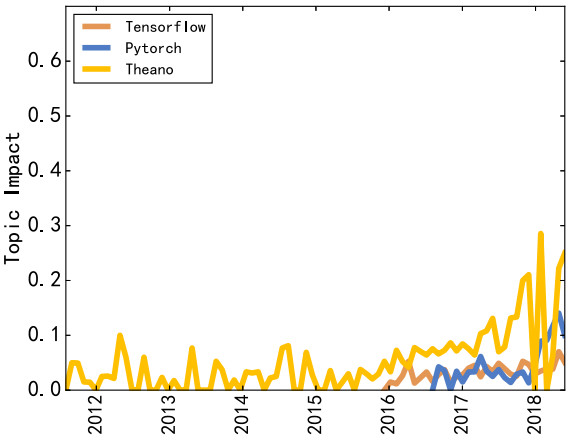
Interestingly, the LDA-topic of *Performance* shows distinct trends on Tensorflow and Theano. It shows an increasing trend on Tensorflow, while shows a decreasing trend on Theano. To properly account for the phenomenon, we conjecture that on owing to that Tensorflow has many defects when using, it can often lead to performance problems, e.g., memory explosion. Besides, many new developers have not mastered the skills well, which may also result in performance problems. However, Theano shows less and less interest in performance problems.

The LDA-topic of *Version Problem* presents the increasing trend both on PyTorch and Theano, which implies that the two deep learning frameworks all need to better solve the version problems and improve the compatibility between different versions.
The LDA-topic of *Performance* shows an increasing trend on Tensorflow and a decreasing trend on Theano. A reasonable explanation for this trend can be attributed to that many new developers have not mastered skills well, which may result in performance problems.

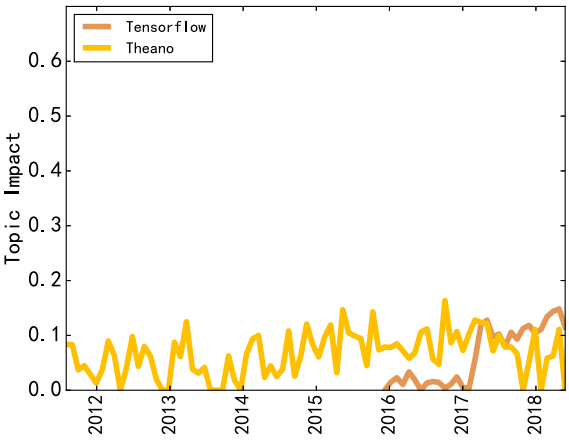
Topic Category Trend It should be noted that the topic category of *Version Problem* in *Preliminary Preparation* stage has distributions on all the three frameworks. We now probe the similarities and differences of the topic trend across the three frameworks, which is described in Fig. 5a. The impact score of *Version Problem* in the three frameworks all presents an increasing trend on the whole from August 2011 to July 2018. Besides, we can find that the time when *Version Problem* emerged is aligned with the time when the frameworks began to appear, and the time when *Version Problem* spiked is aligned with the time when the frameworks released a new version. It gives a hint that more endeavors should be paid to ameliorate the version problems and try the best to improve the compatibility between different versions.

After the trend analysis of *Version Problem*, we examine the topic category of *Performance* in *Model Training* stage, which is illustrated in Fig. 5b. From Fig. 5b, we find that there only two deep learning frameworks on GitHub have produced this topic category, that are Tensorflow and Theano. The *Performance* topic category in Tensorflow has both increases and decreases during the study time period. It was relatively smooth during 2016 to 2017, while presented a sharp increase after January 2017. A reasonable explanation for this trend may be attributed to the release of Tensorflow 1.0, which results in many new problems on *Performance* in a short-term. As for the *Performance* topic category in Theano, we can observe that the topic category has both increases and decreases during the study time period, while presents an relatively steady trend on the whole, which is quite different from the topic trend in Tensorflow.

The impact score of *Version Problem* in the three frameworks all presents an increasing trend on the whole. Besides, we can find that the time when *Version Problem* emerged is aligned with the time when the frameworks began to appear, which gives a hint that more endeavors should be paid to ameliorate the version problems.
The impact of *Performance* was relatively smooth during 2016 to 2017, while presented a sharp increase after January 2017. A reasonable explanation for this trend may be attributed to the release of Tensorflow 1.0, which results in many new problems on *Performance* in a short-term.



(a) Comparative trend analysis of the Version Problem topic category



(b) Comparative trend analysis of the Performance topic category

Fig. 5 Comparative trend analysis of the *Version Problem* topic category in *Preliminary Preparation* stage and the *Performance* topic category in *Model Training* stage across different deep learning frameworks

Workflow Stage Trend. After the trend analysis in topic category level, we further analyze the topic trends in workflow level at different deep learning frameworks. We apply the *impact* metric (3) to investigate the trends and the results are presented in Figs. 6, 7 and 8. We can first observe that Tensorflow includes 5 workflow stages, PyTorch includes 4 workflow stages, while Theano includes 6 workflow stages, which implies that different deep learning frameworks show different concerns on the workflow stages.

Moreover, we can observe that the overall impact trends of the workflow stages on Tensorflow and Theano are relatively flat, while the impact trends of the workflow stages on PyTorch fluctuates intensely. From this phenomenon, we can draw a conclusion that the discussions about workflow stages are generally stable on Tensorflow and Theano, while the discussions about workflow stages vary greatly over time on PyTorch. In detail, we can further observe that the *Model Training* stage on Tensorflow and Theano always accounts for the highest impact score, while on PyTorch, it shows the relatively analogous highest

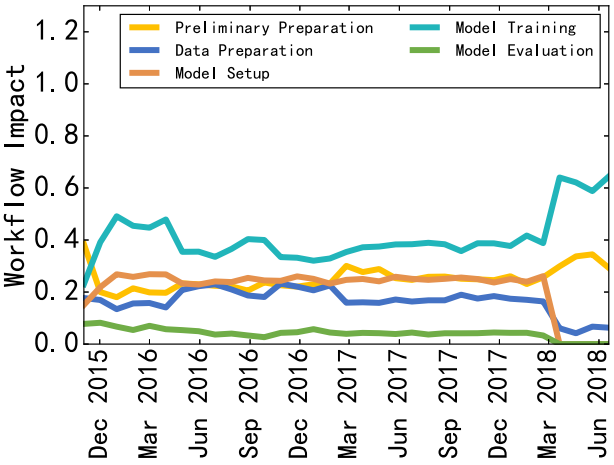


Fig. 6 The impact scores of topics at different workflow stages on Tensorflow between December 2015 and June 2018

impact score with the *Data Preparation* stage. Although the *Model Training* stage takes up the highest impact score on the whole, it is worth noting that it presents a decreasing trend on the whole on PyTorch and Theano. Only on Tensorflow, the impact of it shows an increasing trend.

Results show that the overall impact trends of the workflow stages on Tensorflow and Theano are relatively flat, while the impact trends of the workflow stages on PyTorch fluctuates intensely. From this phenomenon, we can draw a conclusion that the discussions about workflow stages are generally stable on Tensorflow and Theano, while the discussions about workflow stages vary greatly over time on PyTorch.

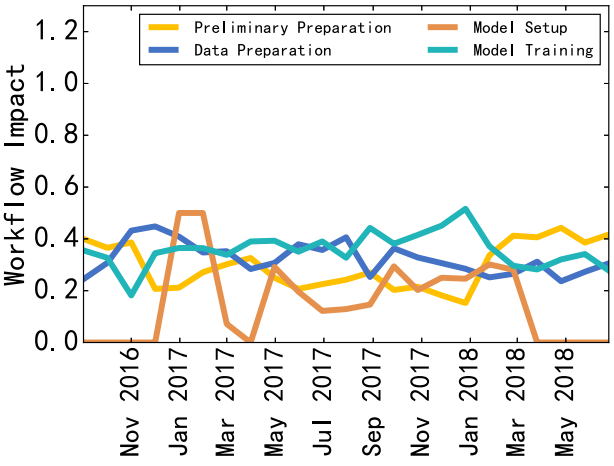


Fig. 7 The impact scores of topics at different workflow stages on PyTorch between September 2016 and July 2018

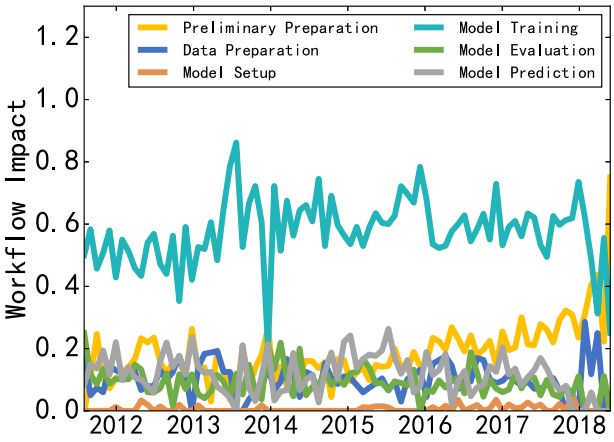


Fig. 8 The impact scores of topics at different workflow stages on Theano between August 2011 and June 2018

4.3 RQ3. What are the differences in the discussions between Stack Overflow and GitHub?

In this section, we analyze the topics of a specific deep learning framework across different platforms to determine the similarities and differences across different platforms. For that we have elaborated the topic distributions across different deep learning frameworks in RQ2, here, we briefly compare the topic distribution on different platforms. We then emphasize the topic trends at the LDA-topic level across different platforms. Lastly, we elaborate on the workflow trends on different platforms. Notably, since there exist three deep learning frameworks, we thus compare each framework on different platforms, separately.

4.3.1 TensorFlow

We first compare the TensorFlow framework on the two studied platforms. The comparison process includes the topic distribution pattern at the workflow level and LDA-topic trend.

Topic Distribution Pattern at the Workflow Level For Tensorflow, we can see that whether on Stack Overflow or GitHub, *Preliminary Preparation* and *Model Training* all account for the highest proportion, which implies that both developers and users all pay close attention to the two stages and derive many problems needed to be resolved. Furthermore, researchers should pay more effort on the two stages, to provide more tools, system documentation, and tutorials. In addition, we need to keep in mind that the posts of it have distributions in *Model Setup* and *Model Evaluation* on Stack Overflow, while it has no distributions on GitHub in terms of the two-stage topics. The results give a hint that users tend to discuss the two-stage topics on Stack Overflow, while developers do not discuss the two-stage topics on GitHub.

> Whether on Stack Overflow or GitHub, *Preliminary Preparation* and *Model Training* all account for the highest proportion, which implies that both users and developers all pay close attention to the two stages and derive many problems needed to be resolved.
> Stack Overflow has topic distributions on *Model Setup* and *Model Evaluation*, while GitHub has no distributions on the two workflow stages. The results give a hint that users tend to discuss the two stage topics on Stack Overflow, while developers do not discuss the two stage topics on GitHub.

LDA-topic Trend We can find that the top 3 LDA-topics with largest increases and decreases on Stack Overflow and GitHub are quite different, which shows that users and developers have different concerns. Making analyses on the differences can help clarify the individual needs of users and developers. On Stack Overflow, the interests of users on *Model Training*, *Input Error* and *Model Saving* show the highest increasing trends, which indicates that more and more users are paying attention to the three LDA-topics. A reasonable explanation for this trend may be attributed to the lack of tutorials and the complexity of using for framework. Thus, it is desirable to complete the tutorials and make the framework more easy-using in different stages. While on GitHub, the interests of developers on *Variable Shape*, *Installation in Linux* and *Performance* present the highest increasing trends. It is very likely that more and more developers face with such problems in their development process, which implies that establishing a sound documentation system that provides richer knowledge and experience is urgent. In the future, researchers can take into account augmentation of documentation system and improvement of framework accessibility.

On the other hand, the top 3 LDA-topics with largest decreases on Stack Overflow and GitHub show different results. The users on Stack Overflow show less and less interests in *Keras*, *Runtime Error* and *File Operation*, while the developers on GitHub pay less and less attention on *Gradient* and *Fixing Error*. The less and less interests in error means that whether users or developers are becoming more and more mature when using the framework, which is a good phenomenon. Besides, we need to keep in mind that although the developers on GitHub show less and less interests in *Gradient*, the impact score of *Gradient* is still more higher than many other LDA-topics, which may explain that there still exist many problems towards this issue. Last but not least, we can speculate that for the top 3 LDA-topics with largest increases and decreases, the discussions of developers on GitHub is always earlier than the discussions of users on Stack Overflow.

> On Stack Overflow, the interests of users on *Model Training*, *Input Error* and *Model Saving* show the highest increasing trends. A reasonable explanation for this trend may be attributed to the lack of tutorials and the complexity of using for framework. While on GitHub, the interests of developers on *Variable Shape*, *Installation in Linux* and *Performance* present the highest increasing trends. It is very likely that more and more developers face with such problems in their development process, which implies that establishing a sound documentation system that provides richer knowledge and experience is urgent.
> The users on Stack Overflow show less and less interests in *Keras*, *Runtime Error* and *File Operation*, while the developers on GitHub pay less and less attention on *Gradient* and *Fixing Error*. The less and less interests in error means that whether users or developers are becoming more and more mature when using the framework, which is a good phenomenon.

4.3.2 PyTorch

We then compare the PyTorch framework on the two studied platforms. The comparison process includes the topic distribution pattern at the workflow level and LDA-topic trend.

Topic Distribution Pattern at the Workflow Level For PyTorch, it yields an insight that whether on Stack Overflow or GitHub, developers and users all focus on the *Preliminary Preparation*, *Data Preparation* and *Model Training* stages. The only difference is that users

on Stack Overflow are also absorbed in the *Model Setup* stage, while the developers on GitHub pay no attention to this stage. Our analysis reveals that more mature tutorials and system documentation are needed for *Preliminary Preparation* stage and more tools are needed for *Data Preparation* and *Model Training* stages. In addition, the users on Stack Overflow are concentrated on the *Model Setup* stage, which gives a hint that many users are still confused with the *Model Setup* stage, researchers should derive more feasible tools to help to select models, creating models and verifying models.

On Stack Overflow, the interests of users on *Model Training*, *Input Error* and *Model Saving* show the highest increasing trends. A reasonable explanation for this trend may be attributed to the lack of tutorials and the complexity of using for framework. While on GitHub, the interests of developers on *Variable Shape*, *Installation in Linux* and *Performance* present the highest increasing trends. It is very likely that more and more developers face with such problems in their development process, which implies that establishing a sound documentation system that provides richer knowledge and experience is urgent.
The users on Stack Overflow show less and less interests in *Keras*, *Runtime Error* and *File Operation*, while the developers on GitHub pay less and less attention on *Gradient* and *Fixing Error*. The less and less interests in error means that whether users or developers are becoming more and more mature when using the framework, which is a good phenomenon.

LDA-topic Trend To this extent, we can find that the top 3 LDA-topics with largest increases and decreases on Stack Overflow and GitHub are quite different. The interests of users on Stack Overflow show the highest increasing trends on *Loss Function*, *Gpu Training* and *Tensor Error*, while the interests of developers on GitHub show the highest increasing trends on *Tensor Operation*, *Code Error* and *Version Problem*. From our observation, we find that the users on Stack Overflow pay close attention to the *Model Training* stage (i.e., Loss Function and Gpu Training) and the interests are becoming higher and higher, which suggests that more tools should be generated to help selecting proper loss functions and assisting gpu training process. Consequently, more and more researches can take into consideration providing tools for assisting development processes. While the attention of developers on GitHub are quite different. Their attention on *Version Problem* is becoming higher and higher, which implies that there still exists much work to do with version problems, e.g., improving the compatibility between different versions.

Moreover, the top 3 LDA-topics with largest decreases on Stack Overflow and GitHub present different results. The interests of users on Stack Overflow show the highest decreasing trends on *Network Layer*, *Installation Error* and *Image Training*, while the interests of developers on GitHub show the highest decreasing trends on *Function Operation*, *Build Error* and *System Installation*. The less and less interests of users on Stack Overflow in *Network Layer* and *Installation Error* may be due to the fact that the installation tutorials and the architecture design is getting better. As for the decreasing interests of users in *Image Training*, a reasonable explanation for this trend may be attributed to that deep learning is more and more widely used in many other fields. Furthermore, the decreasing interests of developers in *Build Error* and *System Installation* also reveal the phenomenon that the installation tutorials are becoming more and more complete, which is consistent with the results of users on Stack Overflow. Last but not least, we can observe that for the top 3

LDA-topics with largest increases and decreases, the discussions of developers on GitHub is always earlier than the discussions of users on Stack Overflow.

Whether on Stack Overflow or GitHub, users and developers all focus on the *Preliminary Preparation*, *Data Preparation* and *Model Training* stages. Our analysis reveals that more mature tutorials and system documentations are needed for *Preliminary Preparation* stage and more tools are needed for *Data Preparation* and *Model Training* stages. In addition, the users on Stack Overflow are concentrated on the *Model Setup* stage, which gives a hint that many users are still confused with the *Model Setup* stage, researchers should derive more feasible tools to help selecting models, creating models and verifing models.

4.3.3 Theano

We lastly compare the Theano framework on the two studied platforms. The comparison process includes the topic distribution pattern at the workflow level and LDA-topic trend.

Topic Distribution Pattern at the Workflow Level For Theano, the vast majority of the posts on Stack Overflow fall into the *Model Training* stage and the rest of it falls into *Preliminary Preparation*, while the posts of it on GitHub take up six out of the seven workflow stages. In this regard, we can draw a conclusion that the developers on GitHub encounter a wide range of problems and more solutions should be provided for the developers.

The developers on GitHub encounter a wide range of problems and more solutions should be provided for the developers.

LDA-topic Trend We find that the users on Stack Overflow show increasing interest in *File Operation*, *Code Error* and *Model Training*, while the developers on GitHub show the highest increasing trends of interests on *File Operation*, *Version Problem* and *Using Numpy*. We can observe that the users and developers all show an increasing trends on *File Operation*, which implies that there still exist many problems needed to be resolved in this point. The increasing attention of users on *Code Error* and *Model Training* gives a hint that more tutorials should be completed and more tools should be generated to assist the development processes. The increasing attention of developers on *Version Problem* and *Using Numpy* also imply that there still exists much work to do with version problems and numpy using problems.

Furthermore, the users on Stack Overflow show the decreasing trends of interests on *Gpu Error*, while the developers on GitHub show the highest decreasing trends of interests on *Data Shape*, *Code Graph* and *Performance*. The decreasing interests of users in *Gpu Error* may due to the mature of users in gpu training process. The decreasing interest of developers in *Data Shape*, *Code Graph* and *Performance* may be due to the proficiency in the use of framework and the richness of hardware resources. Last but not least, the discussions of developers on GitHub is still earlier than the discussions of users on Stack Overflow.

We can observe that the users and developers all show an increasing trends on *File Operation*, which implies that there still exist many problems needed to be resolved in this point.
The decreasing interests of users in *Gpu Error* may due to the mature of users in gpu training process. The decreasing interest of developers in *Data Shape*, *Code Graph* and *Performance* may be due to the proficiency in the use of framework and the richness of hardware resources.

Workflow Stage Trend We further use the *impact* metric (3) to investigate the trend of topics at different workflow stages on Stack Overflow and GitHub, respectively. The results are presented in Figs. 9 and 10. We can draw a conclusion that whether on Stack Overflow or GitHub, *Model Training* stage still maintains the highest impact score compared to the other workflow stages. The impact score of the *Model Training* stage on Stack Overflow ranges from 1.0 to 0.3, while on GitHub, it ranges from 0.5 to 0.4. The trendlines of the *Model Training* stage on Stack Overflow and GitHub all present a decreasing trend on the whole. As for the *Preliminary Preparation* stage, we can see that the trendlines of it on both two platforms show an increasing trend on the same. However, the trend of the *Preliminary Preparation* stage on Stack Overflow comes to be relatively stable after 2016, while the trend of it on GitHub shows a stronger upward trend after 2016. One potential explanation for this finding is that the developers with problems on *Preliminary Preparation* are apt to seek out answers on GitHub but not Stack Overflow after 2016.

Then, we make an observation on the *Data Preparation* stage and find that the overall development trend of it is flat. Moreover, the *Data Preparation* stage on Stack Overflow did not come into spring until the end of 2015, while the *Data Preparation* stage on GitHub still stays impact during 2011 to 2018. It is reasonable to expect that before the end of 2015, the developers tend to solve the *Data Preparation* problems on GitHub and start looking for answers on Stack Overflow until the end of 2015. After the analysis on the *Data Preparation* stage, we analyze the *Model Setup* stage trend. The results show that the *Model Setup* stage on Stack Overflow presents a sharp rise in the end of 2015, while the impact of it on GitHub

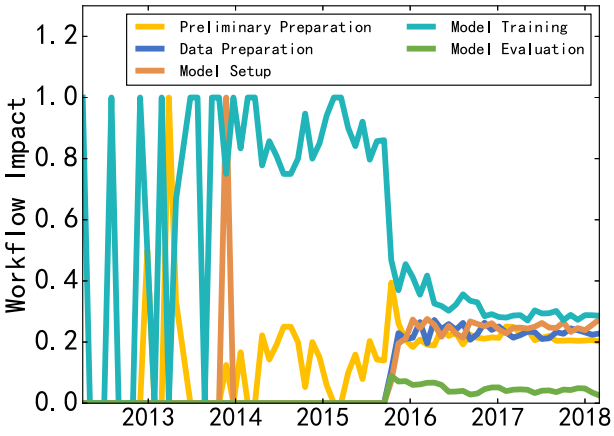


Fig. 9 The impact scores of topics at different workflow stages on *Stack Overflow* between April 2012 and March 2018

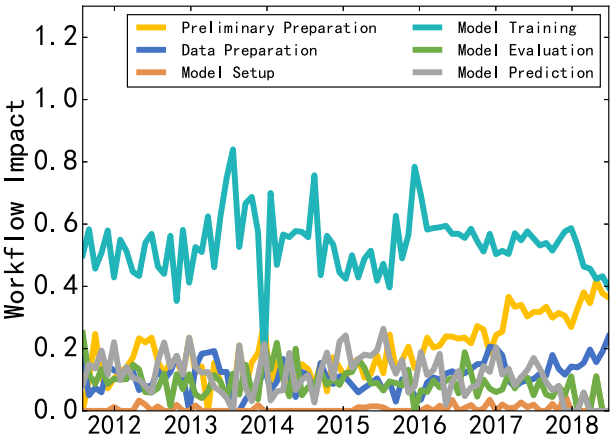


Fig. 10 The impact scores of topics at different workflow stages on *GitHub* between August 2011 and July 2018

is still close to 0. From the results, we can speculate that the users are prone to discuss *Model Setup* problems on Stack Overflow after the end of 2015 but not on GitHub.

When it comes to the *Model Evaluation* stage, the impact score of it is still close to 0.1 on both two platforms, but the trendlines of it on the two platforms are quite different. On Stack Overflow, it comes to be active in the end of 2015, while on GitHub, it is still active during the study time period. We can conjecture that the developers' interests in this stage are not high on both two platforms. Last but not least, we can only observe the trendline of the *Model Prediction* stage on GitHub and the impact score of it is still close to 0.1. The result implies that the developers only discuss the *Model Prediction* problems on GitHub and the interests are not high on the whole.

We derive the trendlines of the topics at different workflow stages on different platforms and observe that *Model Training* on the two platforms still showed the highest impact compared to the other workflow stages during August 2011 to July 2018, *Preliminary Preparation* and *Data Preparation* all showed an increasing trend on the whole on the two platforms during the study time period, *Model Setup* experienced a sharp increase since 2016 on Stack Overflow and the impact of it on GitHub is still slight. As for *Model Evaluation*, the impact score of it is still small on both two platforms and *Model Prediction* only existed on GitHub.

5 Discussions

5.1 Implications

5.1.1 Implications for Researchers

We find that the most popular workflow stages are Model Training and Preliminary Preparation for all the six corpora, where the practitioners all make frequent discussions on the two stages and the two stages all account for a large proportion of posts/records. For researchers,

this will be an ideal area for further research, which can solve the common concerns of developers and users. Moreover, researchers can study a specific issue in depth, e.g., they can make researches on the specified topics, such as system installation, version problems, errors in the model training process, etc.

Researchers addicted to specific issues in depth can provide strong support and guidance for the development of deep learning frameworks, which will further promote and improve the deep learning frameworks. For example, actual posts on Stack Overflow titled "tensorflow not found in pip" (id: 38896424), "Installing tensorflow with anaconda in windows" (id: 37130489), "Trouble with TensorFlow in Jupyter Notebook" (id: 37061089), "How can I install torchtext" (id: 42711144), and "PyTorch doesn't import after installing Anaconda" (id: 41818618) all belong to the LDA-topic of installation error. Therefore, researchers can summarize and categorize the errors practitioners encountered in their installation process. Simultaneously, researchers can also provide tools and documents for newcomers to accelerate the installation process.

5.1.2 Implications for Practitioners

The impact trends of LDA-topics give a hint to practitioners and help them to better understand the development trend of different problems in deep learning frameworks. Software developers and users dedicated to different deep learning frameworks may encounter different problems. Therefore, identifying the issues that practitioners encounter for different deep learning frameworks will give some guidance when they need to choose a deep learning framework to carry out deep learning works. In addition, the LDA-topics with the largest increases or decreases on Tensorflow, PyTorch, and Theano are always different, which means that the practitioners using different deep learning frameworks show different concerns and interests. To choose a proper framework, they should pay more attention to the problems that they may encounter on different deep learning frameworks.

Moreover, since the same topic category on different deep learning frameworks shows different development trend, it can also provide some inspiration to the developers and users. It can persuade them to consider completing the documentation system, augmenting the tutorials, improving the framework accessibility, ameliorating the version problems, generating more tools to provide assistance in the development processes and helping to resolve many other corresponding problems.

Furthermore, the impact trends of the topics at the workflow level also reveal a phenomenon that the overall impact trends of the workflow stages on Tensorflow and Theano are relatively flat, while the impact trends of the workflow stages on PyTorch fluctuate widely. The phenomenon implies that if practitioners are fond of the frameworks with relatively stable discussions, they can choose Tensorflow or Theano. Otherwise, they can choose PyTorch, which is the latest deep learning framework with discussions that intensely fluctuate, which may indicate many new features or novel ideas being suggested.

Discussion topics of a specific deep learning framework on different platforms may be different and always show different development trends. For example, the LDA-topics with the largest increases or decreases of the same deep learning framework between Stack Overflow and GitHub are quite different on the whole, which gives a hint that although using the same deep learning framework, users on Stack Overflow and developers on GitHub always show different concerns and interests. Moreover, the same workflow stages also have different trends on different platforms, e.g., more developers focus on the Model Prediction stage, they may tend to find answers on GitHub by submitting the pull requests or issues of bugs but not asking questions on Stack Overflow. Identifying the tendency of developers

and users to find answers on the two different platforms will give some guidance when we need to choose a proper platform to solve the problems.

5.2 LDA Parameter Analysis

Previous studies show that default LDA parameters always lead to sub-optimal solutions (De Lucia et al. 2014). Therefore, in the process of applying LDA, how to choose the optimal LDA parameters are usually difficult. To choose an optimal value of K and alleviate the threat, we use Roder et al.'s four stage topic coherence pipeline (Both and Hinneburg 2015) to calculate the topic coherence for each LDA model with different values of K. After that, we choose the topic models with optimal K that accompany with the highest coherence score. Besides, since LDA itself is a probabilistic method, when it is run several times on the same corpus, it may produce different results. Therefore, our topic results may vary to a certain degree. To alleviate the variation in the results, we run LDA models at least three times and compare the optimal number of topics in each run. The results of varied K values with its coherence scores for each corpus are shown in Figs. 11, 12, 13, 14, 15 and 16. From Figs. 11 to 16, we can find that there are no significant differences in the optimal K that the LDA runs. Moreover, we can also observe that after the K value reaches the highest coherence score, the coherence score then decreases as the K value increases overall.

Moreover, the default settings for parameters of α and β accompanied with K in gensim LDA is shown in Table 6. Different K values are accompanied with different parameters of α and β, e.g., the optimal K value for the corpus of Tensorflow on Stack Overflow is 20, then the parameters of α and β are 0.05, respectively, which indicates that the LDA model with $K = 20$, $\alpha = 0.05$ and $\beta = 0.05$ reaches the highest coherence score.

5.3 Threats to Validity

Internal Validity. When applying topic models to find LDA-topics, we use information in the title and body of posts/records. Using the body text can improve this process, however, it may also introduce some noise. Since the publisher may add details about their attempts, code errors thrown by the compiler, or other details. These details may

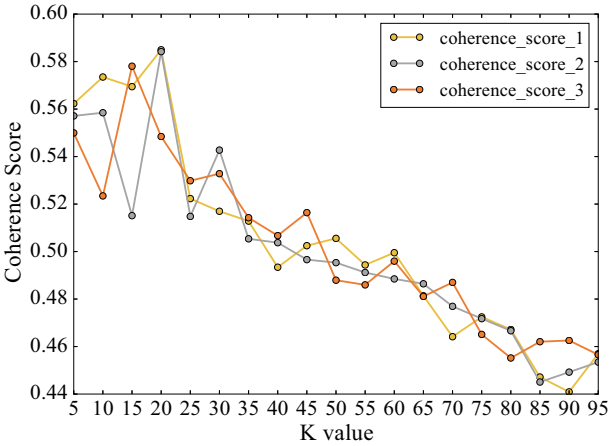


Fig. 11 The vaired K values with its coherence scores for Tensorflow on Stack Overflow

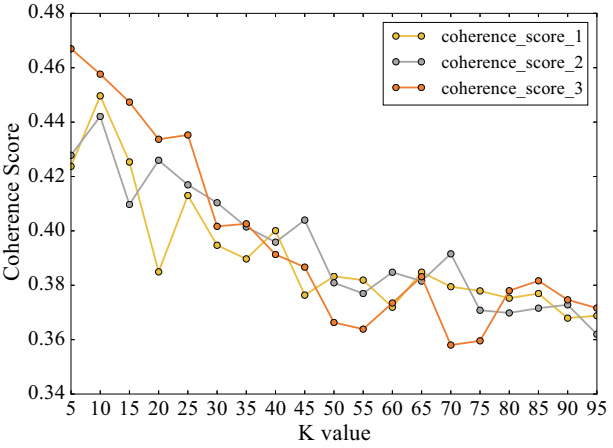


Fig. 12 The vaired K values with its coherence scores for PyTorch on Stack Overflow

take away from the topic that the publisher really asks. To alleviate this threat, when determining the LDA-topics, we not only use the keywords but also randomly select 10 sample posts/records belong to this topic to help determine the final LDA-topics.

The mapping process from LDA-topics to workflow is manually done by two authors, thus, some biases might exist in this mapping process. Nevertheless, in the mapping process, we have referred to many online documents and consulted experienced users in this domain to ensure the accuracy of our classification results to the utmost extent.

External Validity. Although we have incorporated Stack Overflow and GitHub in our study, it may still not enough since there may exist some activities on other platforms. In this regard, our study can be ameliorated by incorporating more sources in the future.

Moreover, in the process of data collection on Stack Overflow, we only leverage the tags of "tensorflow", "pytorch" and "theano" to obtain the question posts of Tensorflow, PyTorch

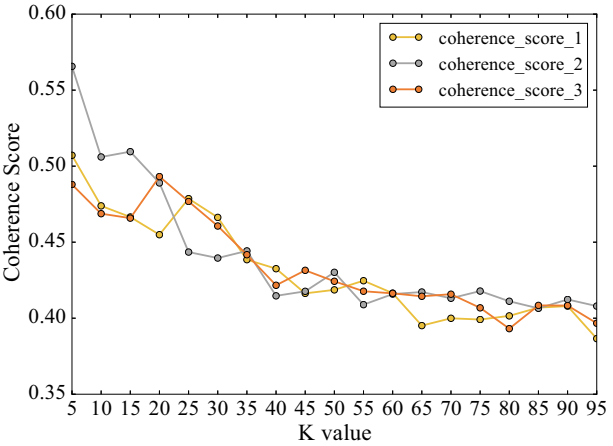


Fig. 13 The vaired K values with its coherence scores for Theano on Stack Overflow

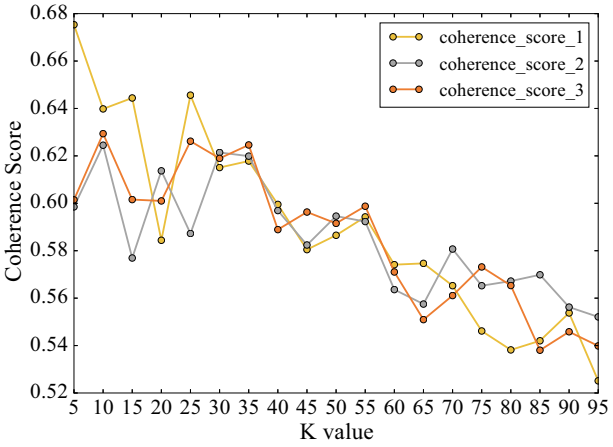


Fig. 14 The vaired K values with its coherence scores for Tensorflow on GitHub

and Theano frameworks, respectively. This may result in the fact that some question posts are related to Tensorflow, PyTorch and Theano frameworks, but was not obtained in our dataset on owing to the absence of "tensorflow", "pytorch" and "theano" tags. Therefore, our study should identify all of the tags related to the Tensorflow, PyTorch and Theano frameworks and use the tags to complete our dataset in the future.

Different deep learning frameworks pay attention to different issues. The discussions on the topics at different levels in this paper may not be generalized to all the deep learning frameworks. To ensure the generalization, more deep learning frameworks should be incorporated. However, the studied deep learning frameworks in this paper are typical in the deep learning domain. Tensorflow represents the most widely-deployed deep learning framework (Abadi et al. 2016). PyTorch stands for the framework with fastest increasing trend recently (Ketkar 2017). And Theano is one of the oldest deep learning frameworks (Bergstra et al. 2010). Through the studies on the three typical deep learning frameworks,

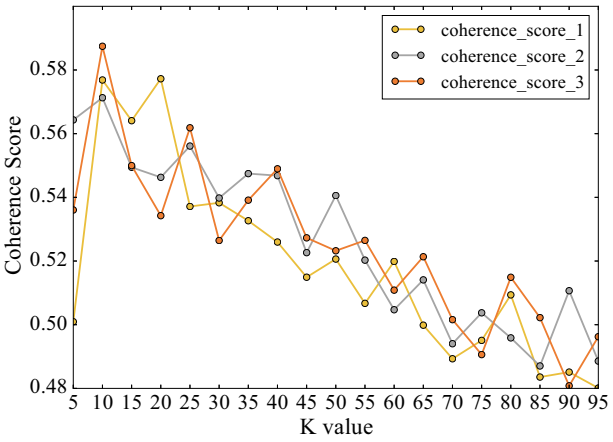


Fig. 15 The vaired K values with its coherence scores for PyTorch on GitHub

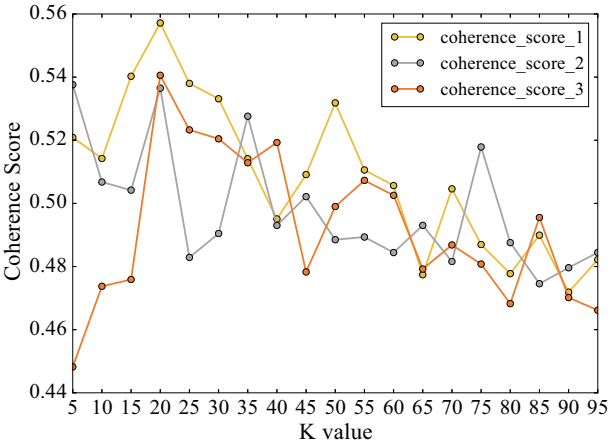


Fig. 16 The vaired K values with its coherence scores for Theano on GitHub

we believe that our findings have some representativeness. We also hope that the findings can bring some enlightenments to developers, users and researchers who use other deep learning frameworks.

Construct Validity After determining the LDA-topics, the first and third authors work together to perform a manual inspection of partial data to guarantee the correctness of our

Table 6 The parameters of α and β with K in LDA models

K value	α	β
5	0.200	0.200
10	0.100	0.100
15	0.067	0.067
20	0.050	0.050
25	0.040	0.040
30	0.033	0.033
35	0.029	0.029
40	0.025	0.025
45	0.022	0.022
50	0.020	0.020
55	0.018	0.018
60	0.017	0.017
65	0.015	0.015
70	0.014	0.014
75	0.013	0.013
80	0.013	0.013
85	0.012	0.012
90	0.011	0.011
95	0.011	0.011

results. To mitigate the risk, we randomly select three sample posts/records for each LDA-topic in each corpus (225 sample posts/records in total) and manually inspect it. We then manually label the 225 sample posts/records and compare the manual-topics with the existing LDA-topics by applying the Cohen's Kappa. As a result, an almost perfect agreement (kappa=0.90) is achieved.

6 Related Work

6.1 Studies on Deep Learning

Zhang et al. (2018) aimed to find the characteristics and root causes of deep learning defects. To achieve this goal, they studied the deep learning applications built on the basis of TensorFlow and collected program bugs related to TensorFlow from both Stack Overflow and Github. They also studied the strategies deployed by TensorFlow users for bug detection and localization, which can give some suggestions and implications to the researchers and TensorFlow users. Another study related to deep learning came from Bahrampour et al. (2015). In this paper, they introduced a comparative study of five deep learning frameworks (Caffe, Neon, TensorFlow, Theano, and Torch) on three aspects, which are: extensibility, hardware utilization, and speed. Experimental results showed that the most easily extensible frameworks are Theano and Torch, and Torch is the most suitable framework for all the deep architectures on CPU. However, the best performance on GPU is achieved by Theano, which can be used to train and deploy the LSTM (Hochreiter and Schmidhuber 1997) networks. In the end, TensorFlow is very flexible, which is analogous to Theano.

6.2 Studies on Stack Overflow

The data dump of Stack Overflow has been widely used in many studies and the researchers used Stack Overflow data dump to solve a diversity of problems. Out of all the studies, some researchers tended to study the development on Stack Overflow. Barua et al. (2014) made an analysis on the textual contents of Stack Overflow discussions and analyzed the relationships and trends of the topics on Stack Overflow. Rosen and Shihab (2016) summarized the mobile-related questions and determined the difficulty of it, they also explored platform-specific issues and investigated the types of questions. Yang et al. (2016) explored the security-related questions and determined the popularity and difficulty of it. Moreover, Baltes et al. (2018) built SOTorrent, which is an open dataset based on the official Stack Overflow data dump to analyze how content on Stack Overflow evolves.

In addition to the researches dedicated to the development of Stack Overflow, there also exist many other studies absorbed in resolving a variety of other problems. Li et al. (2013) conducted an empirical research with 24 developers to identify the needs and challenges developers face in performing development tasks. Bajaj et al. (2014) made a research on the web developers to find the common challenges and misunderstandings of them. Chen et al. (2016) presented an approach to recommend analogical libraries, which can be used to solve analogical-libraries questions. Ye et al. (2016) exploited name synonyms and rich semantic context of API mentions to extract API mentions in informal social text. Yao et al. (2018) mined question-code pairs from Stack Overflow and predicted whether or not a code snippet is a standalone solution to a question. Wang et al. (2018) used a logistic regression

model to analyze 46 factors along four dimensions so as to understand the relationship between the studied factors and the needed time to get an accepted answer. Zagalsky et al. (2018) reported a study of how the R software development community creates and curates knowledge on Stack Overflow.

6.3 Applying LDA on Software Engineering Domains

Nguyen et al. (2011) proposed BugScout, which is an automated approach based on LDA to locate buggy code. Panichella et al. (2013) introduced a solution called LDA-GA, which uses Genetic Algorithms (GA) to determine the approximate optimal configuration of LDA for different SE tasks. Chen et al. (2017) used topic models to generate topics of resources and found that which topics are well tested and which are defect prone. Li et al. (2018) used LDA topic modeling to study the relationship between the topics of a code snippet and the likelihood of a code snippet being logged.

7 Conclusion and Future Work

In this paper, we discover the discussion topics across three different deep learning frameworks (Tensorflow, PyTorch and Theano) on two platforms (Stack Overflow and GitHub). However, due to the fact that LDA blindly captures topics without considering the diversity of datasets and the domain-specific knowledge, we introduce the higher-level domain-specific workflow. After that, we aggregate the generated LDA-topics to the workflow and derive the aggregated topic categories in each workflow stage. Furthermore, we make a comparison across the studied deep learning frameworks and find that the topic distributions at the workflow level and topic category level on Tensorflow and PyTorch still keep the same pattern. Moreover, we analyze the impact trends of topics at different topic levels for a specific deep learning framework across different platforms and gain some key insights. Besides, we also compare the discrepancy of the topics at different workflow levels between the two platforms and yield many interesting conclusions and give many useful suggestions to software developers, users and researchers.

In the future, we tend to analyze how the impact trends of topics at different topic levels vary with respect to the number of newcomers and the number of unique users and gain some key insights. Moreover, we can also incorporate more deep learning frameworks to make the analysis more common and generalized.

Acknowledgment This research was partially supported by the National Key Research and Development Program of China (No. 2017YFB1400601), Key Research and Development Project of Zhejiang Province (No. 2017C01015), National Science Foundation of China (No. 61772461), Natural Science Foundation of Zhejiang Province (No. LR18F020003 and No.LY17F020014).

Appendix

A LDA Topics and Keywords

Tables 7, 8, 9, 10, 11 and 12 illustrate the details of the discovered LDA-topics and their top keywords.

Table 7 Topics Generated by LDA for tensorflow on Stack Overflow

Topic name	Top keywords
Stack Overflow Tensorflow ($K = 20$)	
Model Training	object model tensorflow detection api estimator tf train google using cloud contrib learn ml step
Image Classification	image tensorflow using inception model class cnn size label want classification one trained dataset input
Tensor Operation	tensor tensorflow matrix shape x want like array dimension numpy tf values get vector sparse
Training Accuracy	training model accuracy data loss code learning set network test problem epoch using results validation
Input Error	error code shape tensor tensorflow following get trying input valueerror using placeholder feed help array
Keras	tensorflow keras using code way find know question function could something one need implementation learning
Installation Error	tensorflow python error install import found pip core windows cuda library version anaconda environment module
Performance	gpu memory tensorflow cpu time training using device multiple one run model gb process performance
Runtime Error	tensorflow error run version gpu using code python following get tried problem cuda trying installed
Build Error	tensorflow build bazel run distributed server docker worker file serving using command project error example
Word Embedding	data one input features embedding like tensorflow word feature model would example label use column
File Operation	py tensorflow file line python lib packages site local self users name op run usr
Tensorboard	code tensorboard graph run tensorflow session get following print error see like summary using however
Variable	variable tensorflow weights tf use way like get value new scope how function create need
Loss Function	loss function tensorflow gradient softmax output cross optimizer using tf cost entropy values mean compute
Network Layer	layer network neural output input tensorflow convolutional weights fully connected cnn hidden size keras first
RNN LSTM	rnn lstm sequence input time state output size length cell tensorflow model decoder batch dynamic
Batch	batch tf input graph size tensorflow queue function feed data operation tensor run batches loop

Table 7 (continued)

Topic name	Top keywords
Data Reading	data file dataset tensorflow read training code images mnist test csv set example tutorial like
Model Saving	model tensorflow graph trained file save checkpoint load android restore java inference want code weights

Table 8 Topics Generated by LDA for PyTorch on Stack Overflow

Topic name	Top keywords
Stack Overflow PyTorch ($K = 10$)	
Network Layer	network layer pytorch output use neural input like lstm implement way learning batch get weights
Input Size	pytorch input size torch lstm layers module output error call lib line nn state work
Code Error	code error pytorch get rnn following new tensor word input problem target anyone wrong data
Loss Function	function loss pytorch variable code network gradients backward understand parameters find update class custom something
Installation Error	pytorch install file error py python using code version command get pip packages run conda
Model Training	data model pytorch training example one loss sequence batch simple time two test encoder question
Gpu Training	code gpu using get pytorch training different run work batch following tried something cpu results
Cuda Error	pytorch error cuda following torch model code method trying help gpu message function run loading
Tensor Error	tensor torch size error floattensor pytorch numpy x variable dimension array b w matrix way
Image Training	model images dataset pytorch image cnn make load trained sure one used convolution different models

Table 9 Topics Generated by LDA for Theano on Stack Overflow

Topic name	Top keywords
Stack Overflow Theano ($K = 5$)	
Gpu Error	theano gpu error python using run get following code installed keras version cuda windows trying
File Operation	theano file line py lib packages site cuda python local error import users usr anaconda
Function Operation	theano function matrix tensor code variable using way numpy vector gradient get example value shared

Table 9 (continued)

Topic name	Top keywords
Code Error	error code input theano shape output layer x keras size get following model data batch
Model Training	model network keras using neural data training layer code learning output weights input loss image

Table 10 Topics Generated by LDA for Tensorflow on GitHub

Topic name	Top keywords
GitHub Tensorflow ($K = 10$)	
Value Type	type device name value cpu framework float tensor kernel node run tensorflow shape key list
Build Error	tensorflow build error bazel core external contrib kernels lib cuda cmake usr opt tools bin
Gradient	gradients resnet float unit bottleneck dt mean dynamicpartition batchnorm const truediv moving weights branch grads
Variable Shape	tf x shape self variable size input batch ops name rnn layers dtype output scope
Installation in Linux	ocal bazel tensorflow test root bin linux usr build pip eigen python org framework libtensorflow
File Operation	tensorflow file python line lib packages site import contrib local lite call self home toco
Fixing Error	tensorflow fix java android github org add error issue support api code trees master like
Performance	tensorflow gpu core runtime device common allocator size bfc cuda job task chunk cpu localhost
Model Training	tf model train graph data image input run dataset batch fn step estimator py def
Version Problem	tensorflow version source code tf command problem python gpu cuda ubuntu cudnn binary installed linux

Table 11 Topics Generated by LDA for PyTorch on GitHub

Topic name	Top keywords
GitHub PyTorch ($K = 10$)	
Function Operation	code functions aten tests one make new add changes instead change fixes need cuda fix
Tensor Fixing	tensor grad fix backward add cudnn const long docs native rnn function conv autograd input double
Tensor Operation	tensor test torch add onnx type sparse variable fix scalar return python x script size
Code Error	torch x nn size import input model cuda data print loss return output variable def

Table 11 (continued)

Topic name	Top keywords
Distributed Process	torch size type count jit temp build time autograd distributed python process group win release
File Operation	lib file py line packages torch home site usr linux local rw data gnu call
Build Error	test build pytorch error users fix desktop edward yang signed tolia gpu windows api context
System Installation	lib include install torch pytorch tmp cmake build home found test cuda gaoxiang mkl performing
Third Party	psimd pytorch party third src home cmakefiles usr aten error include nnpack nvidia build mkl
Version Problem	version cuda pytorch gpu conda python source cudnn pip build error gcc os torch installed

Table 12 Topics Generated by LDA for Theano on GitHub

Topic name	Top keywords
GitHub Theano ($K = 20$)	
Data Shape	shape id byte totalsize inplace elemsize elemwise x input tensorconstant w dimshuffle shared b add
Object Error	self n py node null var pyobject idx int ssize pylist call output owner clazylinker
Code Error	fix crash doc add txt news error code update bug gpu reported test remove change
Cuda Error	cuda usr dylib lib ndarray include versions users library system framework nvcc home python error
Gpuarray Bug	gpuarray pygpu cudnn mode test algo fwd dnn call convolution gpu pyx hash milliseconds max
File Operation	theano file line py lib packages gof site local opt error usr compile home node
Clang Error	error clang target feature argument unknown file directory us miniconda theano ms users env int
Import Error	theano import file mno lib packages site line compile py error files module gof include
Installation Problems	core inumpy src anaconda use run py numpy version ga ic configtest file setup build
Warnings	theano tensor py nnet tests conv use deprecated usr instead src shape userwarning warn opt
Data Type	theano x inputs self function outputs value dtype none tensor type scan import np size
Theano Composite	theano elemwise composite inplace time true apply scan gpuelemwise switch ops scalar x add module gpu

Table 12 (continued)

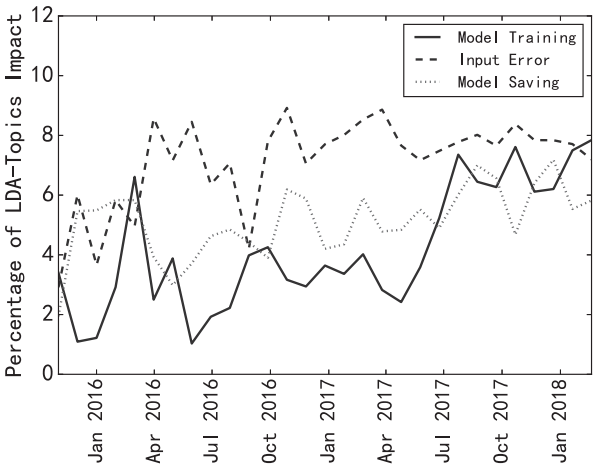
Topic name	Top keywords
Gpu Error	theano cuda gpu device cudnn version error gpuarray import use sandbox python flags init run
Code Graph	code time op make function one use instead graph get way change implementation also first
Errors on Windows	users local appdata theano windows compiledir model family stepping genuineintel mod numpy lib lazylinker undefined
Using Numpy	theano numpy tensor function build builddir test import dot sparse array opti rel batch updated
Performance	gpu new code cpu backend problem running python memory cudnn old work support results one
Storage Error	py storage error pyobject input struct pyerr null pyarray int compiled label item include void
Test Error	test fix make add scan grad also check work new issue bug code travis optimization
Compile Error	theano compiledir linux mod cpp generic ubuntu cache invalid root home error vivid lazylinker access

B LDA-topic Trends

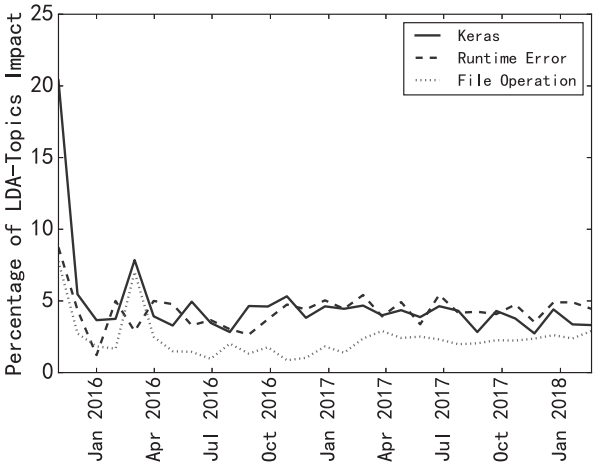
We further make analyses on the development trends of LDA-topics, we first detemine the dominant topics of the posts/records by applying the *dominant topic* metric (1). Then, we calculate the topic trends of the LDA-topics using the *impact* metric (2). Figures 17, 18, 19, 20, 21 and 22 illustrate the top 3 LDA-topics with the largest increases or decreases over time for the six different corpora.

C Examples of LDA Topics on Different Workflow Stages

Preliminary Preparation. Out of the derived 75 LDA-topics, 29 LDA-topics belong to the preliminary preparation stage, which are: 1). *File Operation* for Tensorflow on Stack Overflow, 2). *Keras* for Tensorflow on Stack Overflow, 3). *Installation Error* for Tensorflow on Stack Overflow, 4). *Runtime Error* for Tensorflow on Stack Overflow, 5). *Build Error* for Tensorflow on Stack Overflow, 6). *Code Error* for PyTorch on Stack Overflow, 7). *Installation Error* for PyTorch on Stack Overflow, 8). *File Operation* for Theano on Stack Overflow, 9). *Code Error* for Theano on Stack Overflow, 10). *File Operation* for Tensorflow on GitHub, 11). *Installation in Linux* for Tensorflow on GitHub, 12). *Version Problem* for Tensorflow on GitHub, 13). *Build Error* for Tensorflow on GitHub, 14). *Fixing Error* for Tensorflow on GitHub, 15). *File Operation* for PyTorch on GitHub, 16). *System Installation* for PyTorch on GitHub, 17). *Version Problem* for PyTorch on GitHub, 18). *Third Party* for PyTorch on GitHub, 19). *Build Error* for PyTorch on GitHub, 20). *Code Error* for PyTorch on GitHub, 21). *File Operation* for Theano on GitHub, 22). *Version Problem* for Theano on GitHub, 23). *Using Numpy* for Theano on GitHub, 24). *Import Error* for Theano on GitHub, 25). *Installation Error* for Theano on GitHub, 26). *Warnings* for Theano on GitHub, 27). *Errors on Windows* for Theano on GitHub, 28).



(a) Top 3 increasing trends for Tensor flow on Stack Overflow

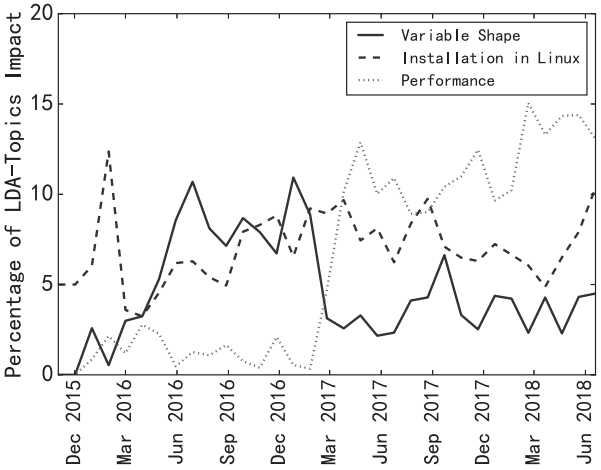


(b) Top 3 decreasing trends for Tensor flow on Stack Overflow

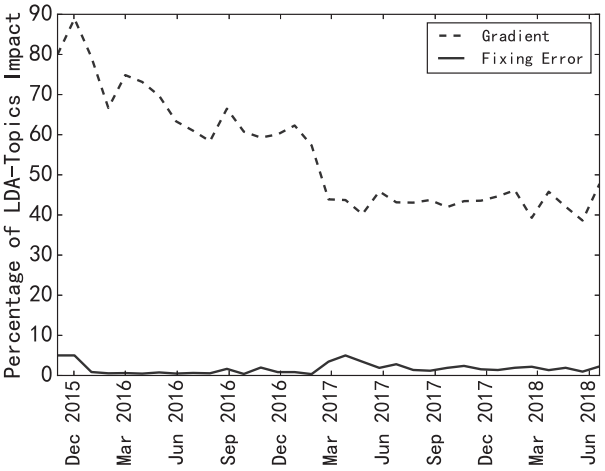
Fig. 17 The top 3 LDA-topics with largest increasing and decreasing trends for the corpus of Tensorflow on Stack Overflow, as measured by the percentage change in topic *impact* scores during November 2015 to March 2018

Compile Error for Theano on GitHub, 29). *Clang Error* for Theano on GitHub. Some example posts/records of these LDA-topics are shown as following:

– *Detect object from video stream using Keras .h5 file*
I am using keras and tensorflow to train a custom model using transfer learning. I was wondering, is there any tutorial which covers custom object detection from live video stream using keras .h5 file? Here is my sample code for training based on *https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/5.3-using-a-pretrained-convnet.ipynb*
Dominant Topic: *Tensorflow on Stack Overflow/Keras*



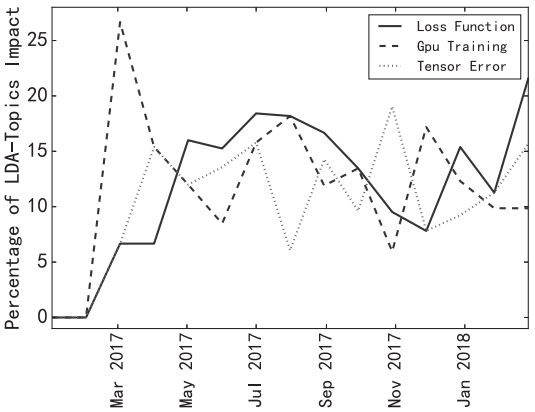
(a) Top 3 increasing trends for Tensorflow on GitHub



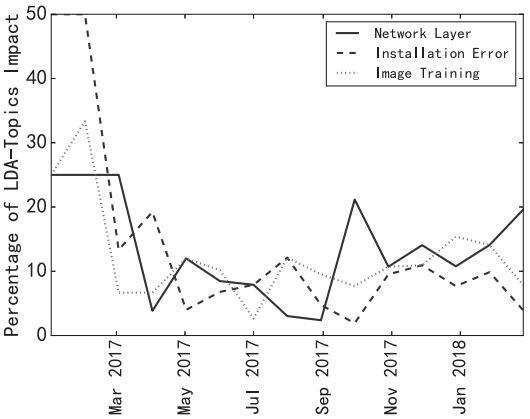
(b) Top 3 decreasing trends for Tensorflow on GitHub

Fig. 18 The top 3 LDA-topics with largest increasing and decreasing trends for the corpus of Tensorflow on GitHub, as measured by the percentage change in topic *impact* scores during November 2015 to July 2018

– *Install PyTorch on Windows*
I am trying to install PyTorch on Windows8.1. I am using Python 3.6.4 and no GPU. I've tried already the Anaconda package provided by peterjc123 by running conda install -c peterjc123 pytorch_legacy cuda80 using a virtual environment. While the installation goes smooth (without errors), after import torch I get the following error. Can somebody help me to install it? Dominant Topic: *PyTorch on Stack Overflow/Installation Error*
– *strides argument, the layer received both the legacy keyword argument subsample and the Keras 2 keyword argument strides*
when I try to run this code with keras 2.1.3 I get this error



(a) Top 3 increasing trends for PyTorch on Stack Overflow



(b) Top 3 decreasing trends for PyTorch on Stack Overflow

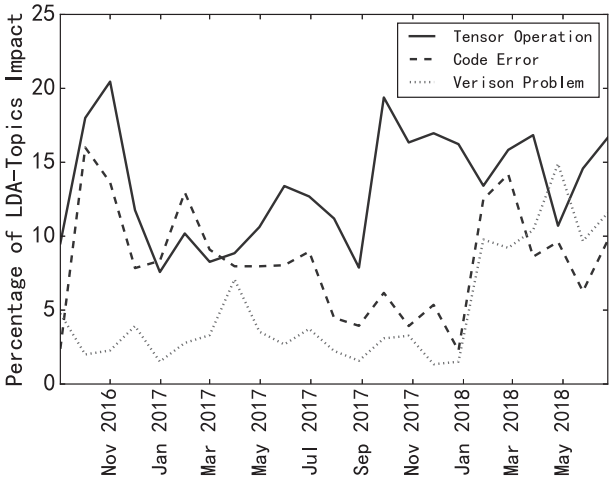
Fig. 19 The top 3 LDA-topics with largest increasing and decreasing trends for the corpus of PyTorch on Stack Overflow, as measured by the percentage change in topic *impact* scores during January 2017 to March 2018

https://github.com/marcellacornia/sam
Dominant Topic: *Theano on Stack Overflow/Code Error*

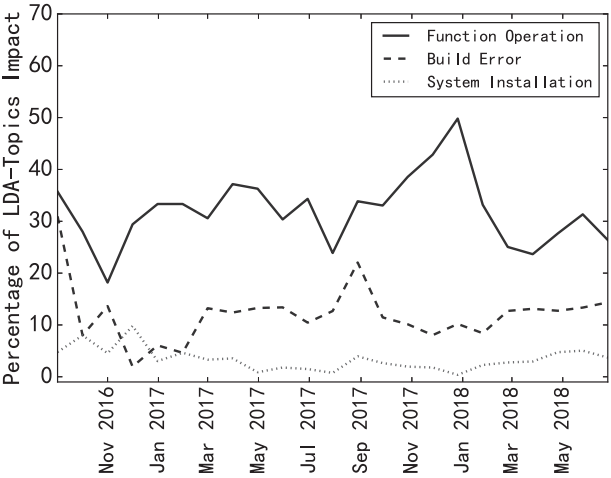
– *cannot import name bayesflow Error*
Hi, I get the error I mentioned in the title. I did a search on Google and I usually found a solution to update dask. I updated Dask to version 0.17.2 but I still get the same error. I can not import BayesFlow. The Tensorflow version is 0.12.1. Thanks for the answers.

Code : from tensorflow.examples.tutorials.mnist import input_data mnist = input_data.read_data_sets("/tmp/data/")

OS : Ubuntu 16.04 LTS
Tensorflow version is 0.12.1



(a) Top 3 increasing trends for PyTorch on GitHub

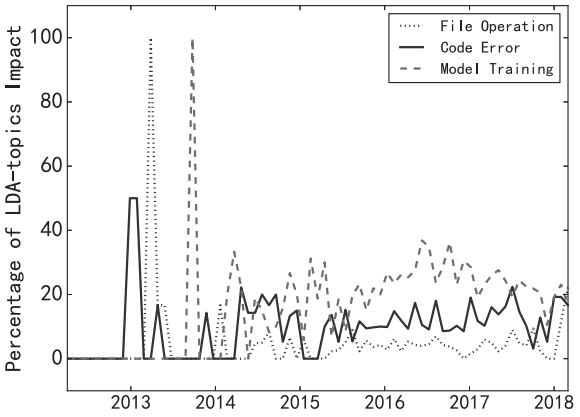


(b) Top 3 decreasing trends for PyTorch on GitHub

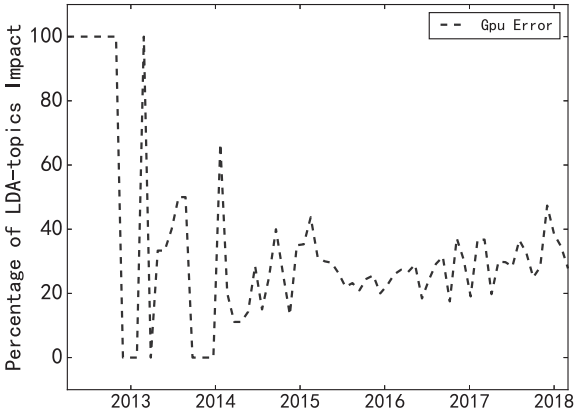
Fig. 20 The top 3 LDA-topics with largest increasing and decreasing trends for the corpus of PyTorch on GitHub, as measured by the percentage change in topic *impact* scores during September 2016 to July 2018

Cuda : 8.0
CuDNN : 5.1
GPU : 4 GB GTX 1050Ti
Dask : 0.17.2
Dominant Topic: *Tensorflow on GitHub/Version Problem*

– *Install hpp headers for CPP Extensions*
With the Cppzation of a few files in 'TH'/'THC', the CPP extensions got broken whenever the user uses feature from 'THC' in their files, when pytorch is installed



(a) Top 3 increasing trends for Theano on Stack Overflow



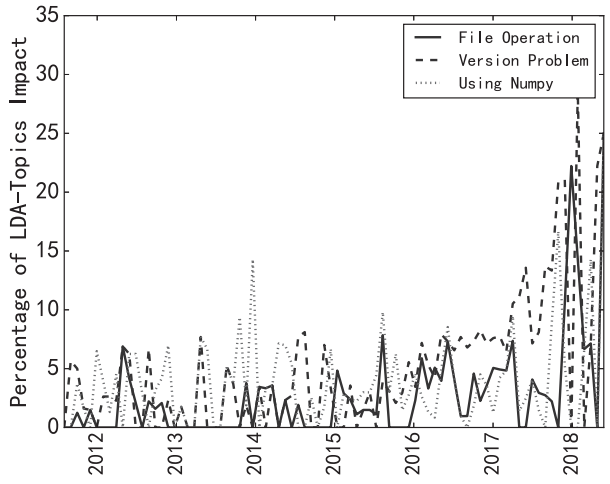
(b) Top 3 decreasing trends for Theano on Stack Overflow

Fig. 21 The top 3 LDA-topics with largest increasing and decreasing trends for the corpus of Theano on Stack Overflow, as measured by the percentage change in topic *impact* scores during April 2012 to March 2018

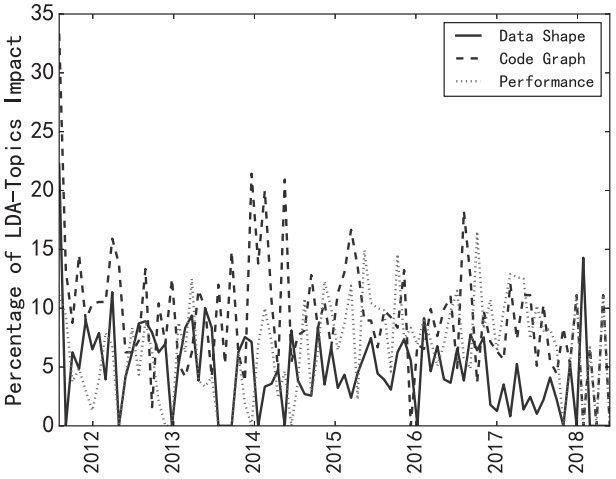
via 'python setup.py install'.

This addresses issues such as "/home/me/.conda/envs/pytorch/lib/python3.6/site-packages/torch/lib/include/THC/THCDeviceTensorUtils.cuh:5:25: fatal error: THCTensor.hpp: No such file or directory"
Dominant Topic: *PyTorch on GitHub/File Operation*

– *Theano cannot detect clang++ in Mac OS X*
I am using the dev version of theano under Mac OS X 10.11.3 with command line tools for Xcode 7. Running theano gives me the following warning:



(a) Top 3 increasing trends for Theano on GitHub



(b) Top 3 decreasing trends for Theano on GitHub

Fig. 22 The top 3 LDA-topics with largest increasing and decreasing trends for the corpus of Theano on GitHub, as measured by the percentage change in topic *impact* scores during August 2011 to July 2018

'WARNING (theano.configdefaults): Only clang++ is supported. With g++, we end up with strange g++/OSX bugs.'

I've also got g++ installed. It seems theano cannot detect the 'clang++'.

Besides, I had strang NaN problems resulting from a simple calculation 'T.dot(W, X)' where 'W' and 'X' do not have nan value (checked with 'np.any(np.isnan)'). I doubt this is because of I am not using 'clang++'
Dominant Topic: *Theano on GitHub/Clang Error*

Data Preparation. We obtain 13 LDA-topics in the data preparation stage, that is: 1). *Variable* for Tensorflow on Stack Overflow, 2). *Data Reading* for Tensorflow on Stack Overflow, 3). *Tensor Operation* for Tensorflow on Stack Overflow, 4). *Input Error* for Tensorflow on Stack Overflow, 5). *Input Size* for PyTorch on Stack Overflow, 6). *Tensor Error* for PyTorch on Stack Overflow, 7). *Value Type* for Tensorflow on GitHub, 8). *Variable Shape* for Tensorflow on GitHub, 9). *Tensor Operation* for PyTorch on GitHub, 10). *Tensor Fixing* for PyTorch on GitHub, 11). *Data Type* for Theano on GitHub, 12). *Data Shape* for Theano on GitHub, 13). *Object Error* for Theano on GitHub. The following is the examples of posts/records of the LDA-topics.

- *Dataset API 'flat_map' method producing error for same code which works with 'map' method*
  I am trying to create a pipeline to read multiple CSV files using TensorFlow Dataset API and Pandas. However, using the flat_map method is producing errors. However, if I am using map method I am able to build the code and run it in session. This is the code I am using. I already opened #17415 issue in TensorFlow Github repository.
  Dominant Topic: *Tensorflow on Stack Overflow/Data Reading*

- *Image Captioning Example input size of Decoder LSTM PyTorch*
  I'm new to PyTorch, there is a doubt that am having in the Image Captioning example code. We first embed the captions and then concat the embeddings with the context feature from the EncoderCNN, but the concat increases the size from embed size how we can forward that to the lstm? As the input size of lstm is already defined as embed_size.
  Dominant Topic: *PyTorch on Stack Overflow/Input Size*

- *Feature request: tf.as_dtype(float) should work just as tf.as_dtype('float')*
  In NumPy, 'np.dtype(float)' works just the same as 'np.dtype("float")'.
  In TensorFlow 'tf.as_dtype("float")' works but 'tf.as_dtype(float)' crashes with 'TypeError: Cannot convert value ¡class 'float'¿ to a TensorFlow DType.'.
  Is there a particular reason for this behaviour or was it just overlooked?
  (same error for other builtins such as 'int' and 'complex')
  Dominant Topic: *Tensorflow on GitHub/Value Type*

- *Explicitly specify the output ndim in reshape*
  The whole code expects the shape to be of length 4, and the output to be 4D already. Fixes #5613.
  Dominant Topic: *Theano on GitHub/Data Shape*

Model Setup. We find 8 model setup LDA-topics in our dataset, namely: 1). *Image Classification* for Tensorflow on Stack Overflow, 2). *Word Embedding* for Tensorflow on Stack Overflow, 3). *RNN LSTM* for Tensorflow on Stack Overflow, 4). *Network Layer* for Tensorflow on Stack Overflow, 5). *Model Saving* for Tensorflow on Stack Overflow, 6). *Image Training* for PyTorch on Stack Overflow, 7). *Network Layer* for PyTorch on Stack Overflow, 8). *Storage Error* for Theano on GitHub. Here are the 3 examples of these LDA-topics.

- *single neuron layer after softmax (keras)*
  I need to create a neural network (with keras) that has as last layer a single neuron that contains the index of the neuron with the maximum value prediction in the

precedent softmax layer. For example my softmax layer gives as result this: [0.1, 0.1, 0.7, 0.0, 0.05, 0.05] And I want that the single neuron layer (after the softmax layer) gives as result 2 (considering a 0 based valutation). How can I do that?
Dominant Topic: *Tensorflow on Stack Overflow/Network Layer*

- *In tensorflow deep and wide tutorial, what's the embedding principle*
  When I played tensorflow tutorial, one embedding trick is used in Wide and Deep tutorial like this.
  The tutorial shows how transfer sparse feature (usually one hot encoding) to embedding vector. I knew there are some approaches to create this embedding, such as word embedding, PCA or t-SNE or matrix factorization. But in this tutorial, they did not show how to create an embedding for the sparse vector. Or did the tutorial just use neural network to finish the embedding?
  Dominant Topic: *Tensorflow on Stack Overflow/Word Embedding*

- *Reading multiple images as custom dataset for PyTorch?*
  I want to read in multiple images for the main_image set and blur_image set. For example, 5 main images and 5 blurred images. The goal is determine what values for the kernel in the convolutional layer convert the main images to the blurred images. The assumption is that the same kernel is used to blur each of the 5 original images to produce the 5 blurred images.
  My code is available at: https://pastebin.com/PWf7rjd4 and https://pastebin.com/VxryDb7g
  However, it seems to only be processing the first image, that is "1.png" for the main and blurred images. It is not processing images 2.png, 3.png, 4.png, and 5.png How can I fix this?
  Dominant Topic: *PyTorch on Stack Overflow/Image Training*

Model Training. We discover 22 model training LDA-topics in our dataset, including: 1). *Model Training* for Tensorflow on Stack Overflow, 2). *Loss Function* for Tensorflow on Stack Overflow, 3). *Batch* for Tensorflow on Stack Overflow, 4). *Performance* for Tensorflow on Stack Overflow, 5). *Training Accuracy* for Tensorflow on Stack Overflow, 6). *Model Training* for PyTorch on Stack Overflow, 7). *Gpu Training* for PyTorch on Stack Overflow, 8). *Loss Function* for PyTorch on Stack Overflow, 9). *Cuda Error* for PyTorch on Stack Overflow, 10). *Function Operation* for Theano on Stack Overflow, 11). *Model Training* for Theano on Stack Overflow, 12). *Gpu Error* for Theano on Stack Overflow, 13). *Model Training* for Tensorflow on GitHub, 14). *Gradient* for Tensorflow on GitHub, 15). *Performance* for Tensorflow on GitHub, 16). *Function Operation* for PyTorch on GitHub, 17). *Distributed Process* for PyTorch on GitHub, 18). *Theano Composite* for Theano on GitHub, 19). *Performance* for Theano on GitHub, 20). *Gpu Error* for Theano on GitHub, 21). *Cuda Error* for Theano on GitHub, 22). *Gpuarray Bug* for Theano on GitHub. The following are the examples of posts/records in our dataset.

- *Try to define pearson correlation as loss function but got error*
  I would like to use pearson correlation as the loss function in Keras with backend of tensorflow. The dimensions of the tensor is (Batch, Coils, Time). The correlation coefficients are to be calculated along time and across coils. For example, if the number of coils is 3, the averaged correlation coefficients will be calculated between

coil #1 and #2, #1 and #3, and #2 and #3.
Dominant Topic: *Tensorflow on Stack Overflow/Loss Function*

- *RNN is not training (PyTorch)*
  I can't get what I am doing wrong when training RNN. I am trying to train RNN for AND operation on sequences (to learn how it works on simple task). But my network is not learning, loss stays the same and it can't event overfit the model. Can you please help me to find the problem?
  Dominant Topic: *PyTorch on Stack Overflow/Model Training*
- *Enabling GPU with theano generates Exception*
  I have followed the steps from here to enable gpu with theano on an Ubuntu 16.04 machine. I installed cuda toolkit, cudnn, drivers but I am still not able to get it to work.
  Dominant Topic: *Theano on Stack Overflow/Gpu Error*
- *[Java] Support addition of gradient operations in a graph*
  This calls the C-api 'TF_AddGradients' method through a new JNI binding for adding gradient nodes to a graph. It also includes an 'AddGradients' wrapper for invoking this operation smoothly while building a graph using the new Java Ops API. Dominant Topic: *Tensorflow on GitHub/Gradient*
- *Use customized python interpreter for distributed launch util*
  For spawning sub-processes, I think it should be quite intuitive to use the interpreter of 'launch.py' rather than the default 'python'.
  Dominant Topic: *PyTorch on GitHub/Distributed Process*

Model Evaluation. We find 2 model evaluation LDA-topics in our dataset, 1). *Tensorboard* for Tensorflow on Stack Overflow , 2). *Code Graph* for Theano on GitHub. The following example post is from these LDA-topics:

- *Tensorboard not creating network graph (Python)*
  I really can't understand why tensorboard is not showing the graph of my network. I have followed the tutorials on Tensorboard Website and other stuff in the web, none of these allowed to display the graph.
  I am embedding the part of my code related to the network. I've tried to remove all the other parts but I did not want to reduce to much otherwise it can create confusion. The only thing it displays on the graph sections is the global_step.
  Dominant Topic: *Tensorflow on Stack Overflow/Tensorboard*

Model Tuning. In our derived 75 LDA-topics, no LDA-topic is related to the model tuning stage.

Model Prediction. Only 1 LDA-topic is associated with the model prediction stage, which is 1). *Test Error* for Theano on GitHub.

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) Tensorflow: A system for large-scale machine learning. In: OSDI, vol 16, pp 265–283

Allamanis M, Sutton C (2013) Why, when, and what: Analyzing stack overflow questions by topic, type, and code. In: Proceedings of the 10th working conference on mining software repositories, IEEE Press, pp 53–56

Azad S, Rigby PC, Guerrouj L (2017) Generating api call rules from version history and stack overflow posts. ACM Trans Softw Eng Methodol (TOSEM) 25(4):29

Bahrampour S, Ramakrishnan N, Schott L, Shah M (2015) Comparative study of deep learning software frameworks. arXiv:151106435
Bajaj K, Pattabiraman K, Mesbah A (2014) Mining questions asked by web developers. In: Proceedings of the 11th working conference on mining software repositories, ACM, pp 112–121
Baltes S, Dumani L, Treude C, Diehl S (2018) Sotorrent: Reconstructing and analyzing the evolution of stack overflow posts. arXiv:180307311
Barua A, Thomas SW, Hassan AE (2014) What are developers talking about? an analysis of topics and trends in stack overflow. Empir Softw Eng 19(3):619–654
Bergstra J, Breuleux O, Bastien F, Lamblin P, Pascanu R, Desjardins G, Turian J, Warde-Farley D, Bengio Y (2010) Theano: A cpu and gpu math compiler in python. In: Proc. 9th python in science conf, vol 1
Blei DM, Ng AY, Jordan MI (2012) Latent dirichlet allocation. J Mach Learn Res 3:993–1022
Both A, Hinneburg A (2015) Exploring the space of topic coherence measures. In: 8th ACM international conference on web search and data mining, pp 399–408
Bovens L, Hartmann S (2010) Bayesian Epistemology. Clarendon
Cai R, Xu B, Yang X, Zhang Z, Li Z (2017) An encoder-decoder framework translating natural language to database queries. arXiv:171106061
Chen C, Gao S, Xing Z (2016) Mining analogical libraries in q&a discussions–incorporating relational and categorical knowledge into word embedding. In: 2016 IEEE 23rd international conference on software analysis, evolution, and Reengineering (SANER), IEEE, vol 1, pp 338–348
Chen TH, Thomas SW, Hemmati H, Nagappan M, Hassan AE (2017) An empirical study on the effect of testing on code quality using topic models: A case study on software development systems. IEEE Trans Reliab R 66(3):806–824
Collobert R, Kavukcuoglu K, Farabet C (2011) Torch7: A matlab-like environment for machine learning. In: BigLearn, NIPS workshop, EPFL-CONF-192376
De Lucia A, Di Penta M, Oliveto R, Panichella A, Panichella S (2014) Labeling source code with information retrieval methods: An empirical study. Empir Softw Eng 19(5):1383–1420
Ding W, Wang R, Mao F, Taylor G (2014) Theano-based large-scale visual recognition with multiple gpus. arXiv:14122302
Duan C, Cui L, Chen X, Wei F, Zhu C, Zhao T (2018) Attention-fused deep matching network for natural language inference. In: IJCAI, pp 4033–4040
Erickson BJ, Korfiatis P, Akkus Z, Kline T, Philbrick K (2017) Toolkits and libraries for deep learning. J Digit Imaging 30(4):400–405
Hannun A, Case C, Casper J, Catanzaro B, Diamos G, Elsen E, Prenger R, Satheesh S, Sengupta S, Coates A et al (2014) Deep speech: Scaling up end-to-end speech recognition. arXiv:14125567
Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
Hoang CDV, Haffari G, Cohn T (2017) Towards decoding as continuous optimisation in neural machine translation. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp 146–156
Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
Hoffman MD, Blei DM, Bach F (2010) Online learning for latent dirichlet allocation. In: International conference on neural information processing systems, pp 856–864
Ketkar N (2017) Introduction to pytorch. In: Deep learning with python, Springer, pp 195–208
Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken AP, Tejani A, Totz J, Wang Z et al (2017) Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR, vol 2, p 4
Lee SR, Heo MJ, Lee CG, Kim M, Jeong G (2017) Applying deep learning based automatic bug triager to industrial projects. In: Proceedings of the 2017 11th joint meeting on foundations of software engineering, ACM, pp 926–931
Li H, Xing Z, Peng X, Zhao W (2013) What help do developers seek, when and how? In: 2013 20th Working Conference on Reverse Engineering (WCRE), IEEE, pp 142–151
Li H, Chen THP, Shang W, Hassan AE (2018) Studying software logging using topic models. Empir Softw Eng 23(5):2655–2694
Li M, Andersen DG, Park JW, Smola AJ, Ahmed A, Josifovski V, Long J, Shekita EJ, Su BY (2014) Scaling distributed machine learning with the parameter server. In: OSDI, vol 14, pp 583–598
Liu H, Xu Z, Zou Y (2018) Deep learning based feature envy detection. In: Proceedings of the 33rd ACM/IEEE international conference on automated software engineering, ACM, pp 385–396

Loper E, Bird S (2002) Nltk: The natural language toolkit. In: Proceedings of the ACL-02 workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1, Association for Computational Linguistics, pp 63–70

Lukins SK, Kraft NA, Etzkorn LH (2008) Source code retrieval for bug localization using latent dirichlet allocation. In: Working conference on reverse engineering, 2008. Wcre '08, pp 155–164

Miller GA (1995) Wordnet: A lexical database for english. Commun ACM 38(11):39–41

Mo W, Shen B, Chen Y, Zhu J (2015) Tbil: A tagging-based approach to identity linkage across software communities. In: Software Engineering Conference (APSEC) 2015 Asia-Pacific, IEEE, pp 56–63

Newman D, Lau JH, Grieser K, Baldwin T (2010) Automatic evaluation of topic coherence. In: Human language technologies: Conference of the North American chapter of the association of computational linguistics, Proceedings, June 2-4, 2010 Los Angeles, California, USA, pp 100–108

Nguyen AT, Nguyen TT, Al-Kofahi J, Nguyen HV, Nguyen TN (2011) A topic-based approach for narrowing the search space of buggy files from a bug report. In: Proceedings of the 2011 26th IEEE/ACM international conference on automated software engineering, IEEE Computer Society, pp 263–272

Panichella A, Dit B, Oliveto R, Di Penta M, Poshyvanyk D, De Lucia A (2013) How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In: Proceedings of the 2013 international conference on software engineering, IEEE Press, pp 522–531

Rosen C, Shihab E (2016) What are mobile developers asking about? a large scale study using stack overflow. Empir Softw Eng 21(3):1192–1223

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252

Schmidhuber J (2015) Deep learning in neural networks: An overview. Neural Netw 61:85–117

Schütze H, Manning CD, Raghavan P (2008) Introduction to information retrieval, vol 39. Cambridge University Press, Cambridge

Spencer D (2009) Card sorting: Designing usable categories. Rosenfeld Media

Thomas S (2012) Mining unstructured software repositories using ir models

Treude C, Robillard MP (2016) Augmenting api documentation with insights from stack overflow. In: 2016 IEEE/ACM 38th international conference on software engineering (ICSE), IEEE, pp 392–403

Treude C, Barzilay O, Storey MA (2011) How do programmers ask and answer questions on the web?: Nier track. In: 2011 33rd international conference on software engineering (ICSE), IEEE, pp 804–807

Vasilescu B, Filkov V, Serebrenik A (2013) Stackoverflow and github: Associations between software development and crowdsourced knowledge. In: 2013 international conference on social computing (SocialCom), IEEE, pp 188–195

Wan Y, Zhao Z, Yang M, Xu G, Ying H, Wu J, Yu PS (2018) Improving automatic source code summarization via deep reinforcement learning. In: Proceedings of the 33rd ACM/IEEE international conference on automated software engineering, ACM, pp 397–407

Wan Z, Lo D, Xia X, Cai L (2017) Bug characteristics in blockchain systems: A large-scale empirical study

Wan Z, Xia X, Hassan AE (2019) What do programmers discuss 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 stack exchange communities. IEEE Trans Softw Eng 2019:1–1

Wang S, Chen TH, Hassan AE (2018) Understanding the factors for fast answers in technical q&a websites. Empir Softw Eng 23(3):1552–1593

Weng R, Huang S, Zheng Z, Dai X, Chen J (2017) Neural machine translation with word predictions. arXiv:170801771

Yang XL, Lo D, Xia X, Wan ZY, Sun JL (2016) What security questions do developers ask? a large-scale study of stack overflow posts. J Comput Sci Technol 31(5):910–924

Yao Z, Weld DS, Chen WP, Sun H (2018) Staqc: A systematically mined question-code dataset from stack overflow. arXiv:180309371

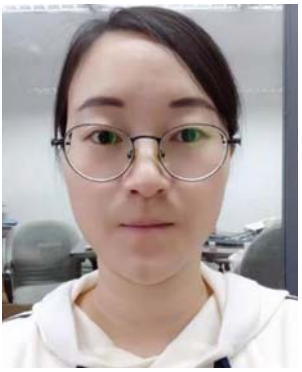
Ye D, Xing Z, Foo CY, Li J, Kapre N (2016) Learning to extract api mentions from informal natural language discussions. In: 2016 IEEE international conference on software maintenance and evolution (ICSME), IEEE, pp 389–399

Yu L, Mishra A, Mishra D (2014) An empirical study of the dynamics of github repository and its impact on distributed software development. In: OTM confederated international conferences" on the move to meaningful internet systems", Springer, pp 457–466

Zagalsky A, German DM, Storey MA, Teshima CG, Poo-Caamaño G (2018) How the r community creates and curates knowledge: an extended study of stack overflow and mailing lists. Empir Softw Eng 23(2):953–986

Zhang Y, Chen Y, Cheung SC, Xiong Y, Zhang L (2018) An empirical study on tensorflow program bugs

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Junxiao Han received the B.S degree in the College of Software Engineering, Shandong University, China, in 2015. She is currently working toward the PhD degree in the College of Computer Science, Zhejiang University. Her research interests include software engineering and repositories mining.

Emad Shihab is an Associate Professor and Concordia Research Chair in the Department of Computer Science and Software Engineering at Concordia University. Dr. Shihab's research interests are in Software Quality Assurance, Mining Software Repositories, Technical Debt, and Software Predictive Analytics. He worked at BlackBerry in Waterloo, Ontario and Microsoft Research in Redmond, Washington. Dr. Shihab is a senior member of the IEEE. More information can be found at http://das.encs.concordia.ca.

Zhiyuan Wan received the PhD degree in computer science from the College of Computer Science and Technology, Zhejiang University, China, in 2014. She is currently a postdoctoral research fellow in the College of Computer Science and Technology, Zhejiang University, China, and a research scientist in the School of Information Systems, Singapore Management University, Singapore. Her research interests include software engineering, software security, and programming languages.

Shuiguang Deng is currently a full professor at the College of Computer Science and Technology in Zhejiang University, China, where he received a BS and PhD degree both in Computer Science in 2002 and 2007, respectively. He previously worked at the Massachusetts Institute of Technology in 2014 and Stanford University in 2015 as a visiting scholar. His research interests include Edge Computing, Service Computing, Mobile Computing, and Business Process Management. He serves as the associate editor for the journal IEEE Access and IET Cyber-Physical Systems: Theory & Applications. Up to now, he has published more than 100 papers in journals and refereed conferences. In 2018, he was granted the Rising Star Award by IEEE TCSVC. He is a fellow of IET and a senior member of IEEE.

Xin Xia is a lecturer at the Faculty of Information Technology, Monash University, Australia. Prior to joining Monash University, he was a post-doctoral research fellow in the software practices lab at the University of British Columbia in Canada, and a research assistant professor at Zhejiang University in China. Xin received both of his Ph.D and bachelor degrees in computer science and software engineering from Zhejiang University in 2014 and 2009, respectively. To help developers and testers improve their productivity, his current research focuses on mining and analyzing rich data in software repositories to uncover interesting and actionable information. More information at: https://xin-xia.github.io/.

Affiliations

Junxiao Han[1] · Emad Shihab[2] · Zhiyuan Wan[1] · Shuiguang Deng[1] · Xin Xia[3]

Junxiao Han
junxiaohan@zju.edu.cn

Emad Shihab
eshihab@encs.concordia.ca

Zhiyuan Wan
wanzhiyuan@zju.edu.cn

Xin Xia
xin.xia@monash.edu

1 College of Computer Science and Technology, Zhejiang University, Hangzhou, China

2 Department of Computer Science and Software Engineering, Concordia University, 1455 Boulevard de Maisonneuve O, Montréal, QC, H3G 1M8, Canada

3 Faculty of Information Technology, Monash University, Wellington Road, Victoria, 3800, Australia