Mining What Developers Are Talking About Deep Learning

JIN Fenglei
Supervisor: Michael R. Lyu

Abstract

Deep learning are gaining more and more attentions these days, which attracts lots of engineers and researchers jumping into this area. Understanding the questions raised by developers in public discussion forum is helpful for industry practitioners and research scientists to capture the trends of deep learning and comprehend the common concerns about designing deep learning techniques. Also, detecting major bugs of different platforms and concerns of users helps increase the reliability of deep learning tools.

In this paper, we crawl over 10,000 questions about deep learning in StackOverflow, and simulate the Attention-based Aspect Extraction[1] model and IDentify Emerging App issues (IDEA) model[2] to extract common aspects embedded in these questions. Since questions have its own distinguishing features, a specific topic interpretation method is proposed. To figure out the outcomes of each approach, we propose to combine the Automatic Topic Coherence Evaluation model[3]. Finally, we evaluate the model performance quantitively and visualize the results to facilitate readers' observation.

Keywords: deep learning, unsupervised learning, aspect extraction

Contents

1. INTRODUCTION

Deep learning is a class of machine learning algorithms that[1]:

- Use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- Learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- Learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.

Deep learning techniques are gaining more and more attentions these days, which attracts lots of engineers and researchers jumping into this area. Hundreds of thousands of questions about deep learning have been asked in online Q&A forums like StackOverflow and StackExchange. For instance, under the tag "deep-learning", now you can find over 11,000 questions from developers all over the world in StackOverflow. Under this situation, it is normal to see questions such as "I'm new to this (deep learning) field, ...", which to some extend shows that many new developers tend to enter this field and ask some basic questions.

We think it is significant and necessary for these "newbies" to understand the history of development, which to some extend help them learn what this

[1] https://en.wikipedia.org/wiki/Deep_learning

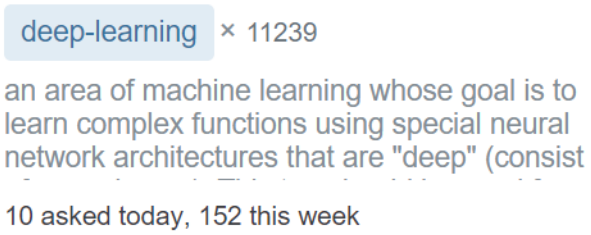

Figure 1: The "deep-learning" tag in StackOverflow.

field is really talking about. Also, for experienced developers, it is good to capture the trend of deep learning and discover hot topics among discussions of other developers. What's more, timely and precisely identifying the emerging issues is of great help for developers to get inspiration of it. We find that famous Q&A platforms only give naive methods like giving tags to analyze deep learning related questions, let alone describing key words or analyzing trend or hot topics. Therefore, we think it is reasonable to implement a tool to discover hot topics among the questions post by developers.

Moreover, a new release of deep learning tools or a breakthrough, which is called an emerging issue, may provoke a discussion and last for several months (e.g., reinforcement learning after AlphaGo Zero won Ke Jie, which leads to the rising of a new field), And for platform maintainers, new versions are always accompanied by bugs and queries of users. Following the discussion of users is a key method to make an improvement.

Questions posted by developers directly reflect the focus and common concerns of the deep leaning field. For example, the high "views" questions

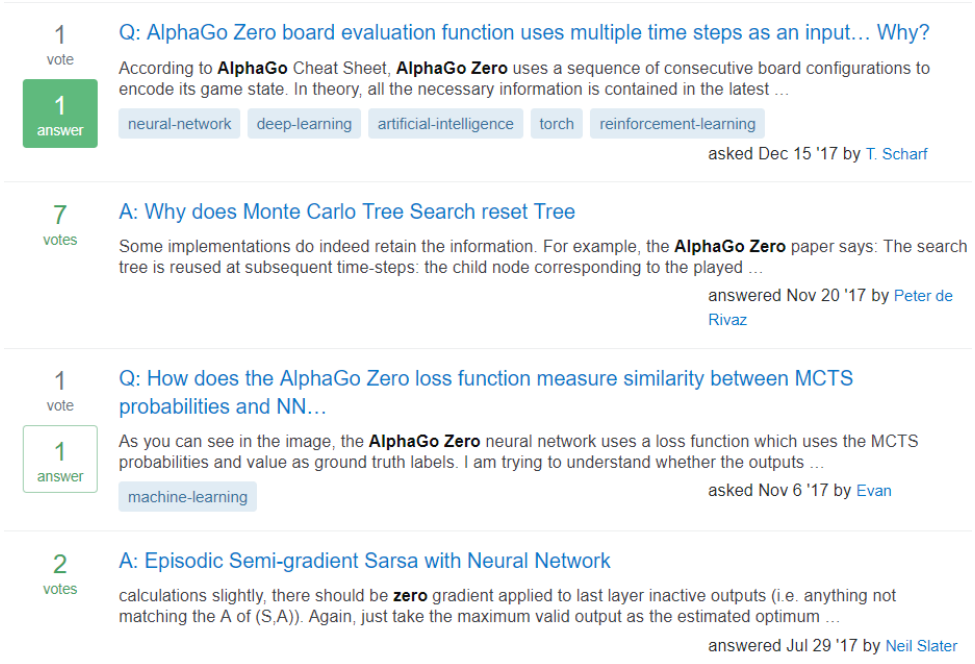

Figure 2: Discussion of AlphaGo Zero.

are more likely to be classical topics which are concerned by almost all deep learning developers. The "high-vote" (or "high-score") questions are more likely to be good questions or questions with correct and detailed answers. And recent questions may refer to new emerging issues. As an example, in October 2017, we find a lot of posts contain the word "Sophia", which is an AI robot and the first robot to receive citizenship at that time, leading to a hot discussion about artificial intelligence. These recent issues may help developers to follow closely the tide of deep learning. To automatically extract topics from thousands of questions posts by developers, we propose to base our strategy on deep learning aspect extraction method.

"Previous works for aspect extraction can be categorized into three approaches: rule-based, supervised,and unsupervised"[1]. Rule-based methods are usually based on manually made dictionaries or rules which can not group aspect terms into categories. Supervised learning is much more popular these days, and gains success with a large number of different models. However, it requires data annotation which may be not always available, and suffers from domain adaptation problems. Unsupervised methods avoid the limitation on labeled data, but always need some manually labeled test data to further confirm the result. In this project, since the data are not labeled and it is too tedious to manually assign labels to all the training questions, we chose to use unsupervised methods.

"In recent year Latent Dirichlet Allocation[4] (LDA) (Blei et al., 2003) and its variants[5] [6] [7] [8] (Titov and McDonald, 2008; Brody and Elhadad, 2010; Zhao et al., 2010; Mukherjee and Liu, 2012) have become the dominant unsupervised approach for aspect extraction"[1]. LDA models regard the whole document as a mixture of topics (aspects), and different words types contribute to topics. LDA can describe a corpus fairly well, but is struggled to extract individual aspects: aspects are of poor quality which consist unrelated or loosely-related concepts. This problem may caused by two primary reasons: "conventional LDA models do not directly encode word co-occurrence statistics which are the primary source of information to preserve topic coherence"[9] (Mimno et al., 2011), which means each word is generated from document level independently in its assumption; furthermore, LDA-based models tend to analyze the distribution of topics in each docu-

ment, facing the difficulty that when each document is too short to make a good estimation of topic distributions. This will certainly have a negative influence on the training result of the model.

To reduce the influence of LDA-bases model, He et al. proposed an Attention-based Aspect Extraction (ABAE) model. This model starts with word embedding which maps words that usually co-occur within the same context to nearby points in the embedding space (Mikolov et al., 2013), then filter the word embeddings within a sentence using an attention mechanism (Bahdanau et al., 2015) and using the filtered words to construct aspect embeddings[1]. Dimension reduction is used to extract the similar factors among embedded sentences and reconstruct each sentence through a linear combination of aspect embeddings, which is analogous to autoencoders. To reduce the the significance of the words that are not part of any aspect, and focus more on the more "meaningful" aspect words, the attension mechanism is introduced.

Since word-occurrence statistics are changed to word embeddings, the most important aspects in the documents can be extracted by dimension reduction. And with attention mechanism used to eliminate irrelevant words, the coherence of aspects are further improved.

For topic number, Gallagher et al. proposed a Correlation Explanation (CorEx)[12] approach which can help decide aspect numbers. Each aspect explains a certain portion of the total correlation (TC), and we denote the

sum of the topic TCs as the overall TC. To assess how many topics to choose, we can look at the distribution of TCs. As a rule of thumb, additional latent aspects should be added until additional aspects contribute little to the overall TC.

To analyze the training results, we propose to combine the Automatic Topic Coherence Evaluation model[3] proposed by Lau et al., which aims to automatically evaluate the whole topic models according to the topic words of the result with an open-source toolkit[2] for topic and topic model evaluation.

To extract more detailed information like phrases and sentences, a novel and automated framework IDEA[2] for detecting emerging issues/topics based on online review analysis is used and modified. IDEA takes questions of different time periods as input. To track the topic variations over versions, a novel method AOLDA (Adaptively Online Latent Dirichlet Allocation) is employed for generating time-sensitive topic distributions. The emerging topics are then identified based on the typical anomaly detection method. A new ranking schema uses both semantic relevance and user sentiment is proposed in this paper, which leads to a better result with higher topic coherence.

For data set, we find that StackExchange has already collected their questions related to deep learning and the data are available on the Internet[3].

[2]https://github.com/jhlau/topic_interpretability

[3]https://archive.org/download/stackexchange

This publicly-available data contain "posts", "tags", "votes", "views" and many other information about every question from August 2016 to December 2018, including over 8,000 questions. Though the size is not too big compared to the questions of other field like python or javascript, it is still a hard work for manually analyzing. It cost us a lot of time and effort to manually label over 500 questions in six different topics and figure out the prediction accuracy of the model. What's more, the data will be updated in a certain time period to contain more questions when time passing by, which means we can get more data in the future and always update our data set. For the "posts", we find that it contains html tags, websites, pseudo codes, codes of different programming languages, strange variables, misspelled words, repetitive words, and non-English words. It is a challenge for us to preprocess these posts and maintain the original information as much as possible. However, things go even worse for other files like "comments", so we decided to only use the posts as input to train the model. Moreover, data attributes like "votes" and "views" are important to show the quality of the the posts, as good questions always get a higher votes and views, we add them as parameters to interpret the results more accurately.

In case that we may need more data in the future, we try to crawl some questions related to deep learning in StackOverflow by ourselves. There were over ten thousand questions about deep learning when we did the crawling, and we crawled all of them in the format similar to the public StackExchange data set.

Our paper makes the following achievements:

- Crawl over 10,000 questions about deep learning in StackOverflow
- Simulate the Unsupervised Neural Attention Model for Aspect Extraction[1] and learn the aspects embedded in deep learning related quesions
- Add exponential decay method to AOLDA model to improve the topic coherence
- Modify IDEA model with new topic interpretation methods to train topic embeddings in phrase level and sentence level
- Manually label over 500 questions with six different topics and use them to test the prediction accuracy of topic embeddings
- Visualize and analyze the extracted topics and their trends

The rest of the paper is organized as following:

Section 2 presents the details of background and motivation. Section 3 analyzes the related work. Section 4 demonstrates the methodology. Section 5 analyzes and evaluates our research result. Section 6 shows the visualization figures such as word cloud and issue river. Section 7 proposes the future work.

2. BACKGROUND AND MOTIVATION

This section first demonstrates the popularity of deep learning by statistics from different websites. After that, it presents our Online Question Analysis tool and shows its advantages comparing to other tools already occurred.

2.1. Popularity of deep learning

Recently deep learning has become one of the most popular topics in the academic field. Figure 3 and Figure 4 are all generated in Google trends[4], which represent the study popularity in different time period and different locations. The popularity of deep learning growth rapidly since 2012, and it is a prevalent issue in China and Hong Kong.

In universities, research projects about deep learning and other related works become more and more popular , while farsighted companies establish AI department and hire experienced deep learning developers. As shown in Figure 5, deep learning related researches occupies a large proportion of all the publications in Tsinghua University. And in Computer Science and Engineering Department of the Chinese University of Hong Kong, one third of the Final Year Projects and Summer Researches are about deep learning. Hundreds of questions about deep learning are asked in the Internet, with a lot of hot discussions.

[4]https://trends.google.com/trends/

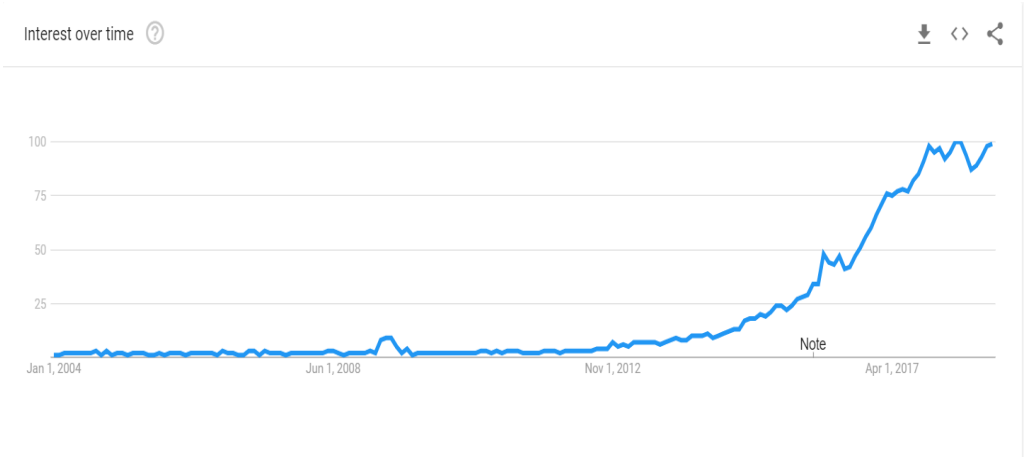

Figure 3: Numbers represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. A score of 0 means there was not enough data for this term.

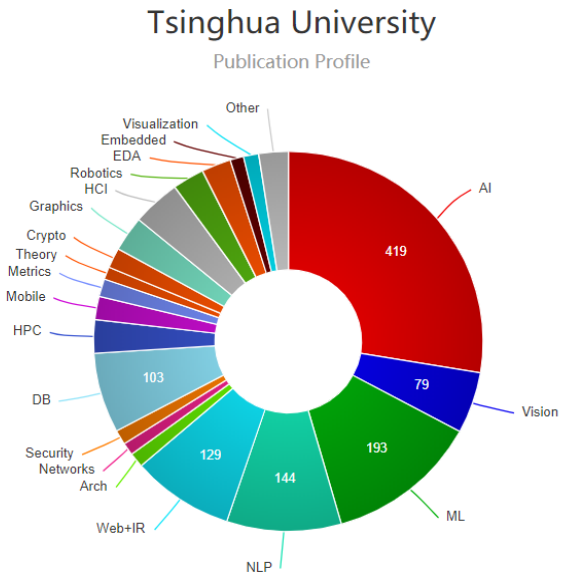

Figure 5: Publication Profile Distribution of Tsinghua University

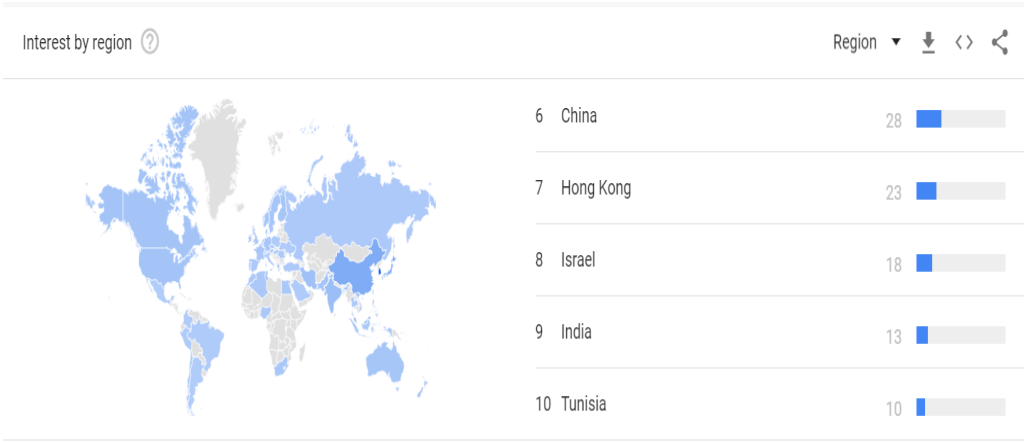

Figure 4: Values are calculated on a scale from 0 to 100, where 100 is the location with the most popularity as a fraction of total searches in that location, a value of 50 indicates a location which is half as popular. A value of 0 indicates a location where there was not enough data for this term.

In this trend, more and more deep learning beginners appear. If you go through the questions in popular Q&A forums, it is common to see someone is saying that he or she is new to this field and want to ask a naive question, like the question shown in Figure 6. And in Figure 6, you can see the "votes" in the left parts to show the quality of this question measured by other users.

For learners new to this field, it is good to get a overall understanding of deep learning, for instance, deep learning is a extremely large topic and contains many sub-topics like NLP, image classification and reinforcement learning which becomes much popular nowadays. And it is also interesting and helpful to know the developing history of deep learning.

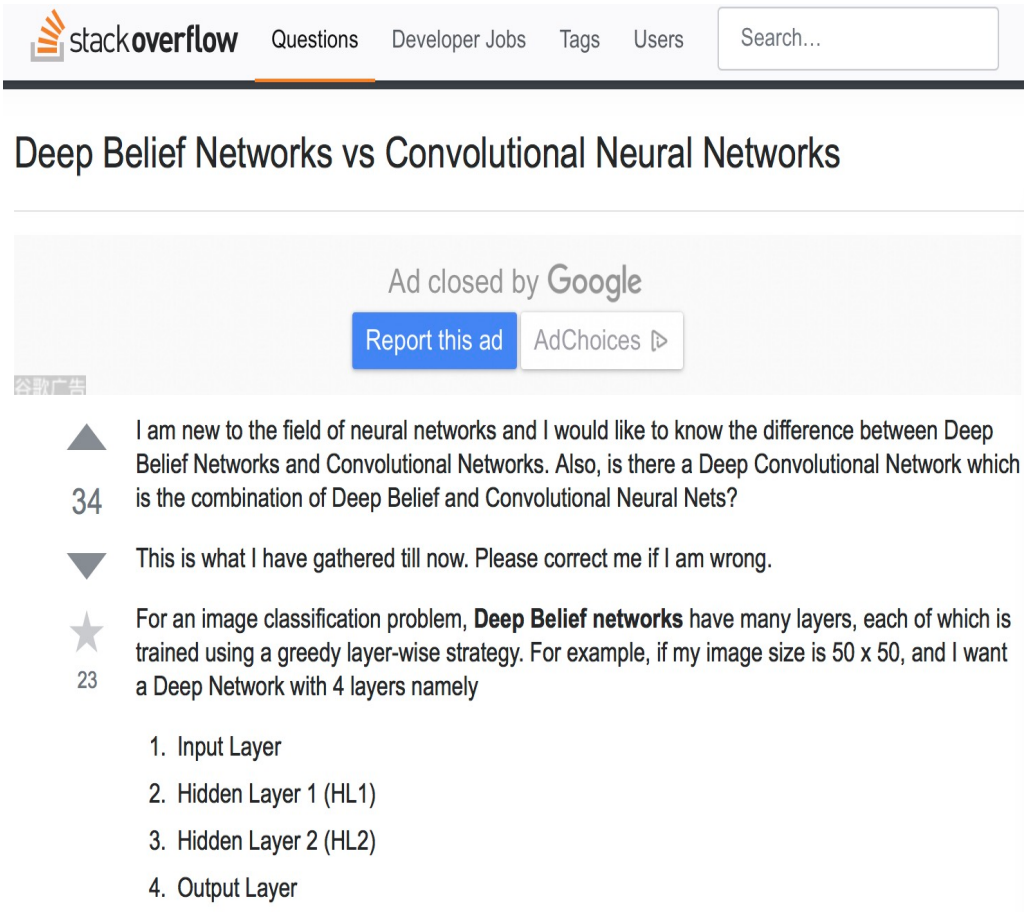

Figure 6: Question of deep learning asked by "newbie".

For deep learning experts, knowing the newest information of deep learning field gives them inspiration and helps figuring out problems more easily. For instance, some newly emerging models may be very useful to solve some certain problems, and others' mistakes may also be a warning for future study.

With the development of AI, many deep learning related tools are in-

vented and used. TensorFlow, Caffe, PyTorch, and a lot of other platforms and tools (e.g., python packages) occurred and is developing on a very high speed. Every time a new version is released, there may occur some bugs and some developers may get confused with some new features, and they will ask questions on AI related forums. It is good for maintainers of these tools to get the analysis of users' feedback automatically and react in time. This helps increase the maintenance and reliability of the tools.

Due to the age of AI is coming in, it is exceedingly reasonable to build an automatic online deep learning related questions analysis tool.

2.2. Online Question Analysis

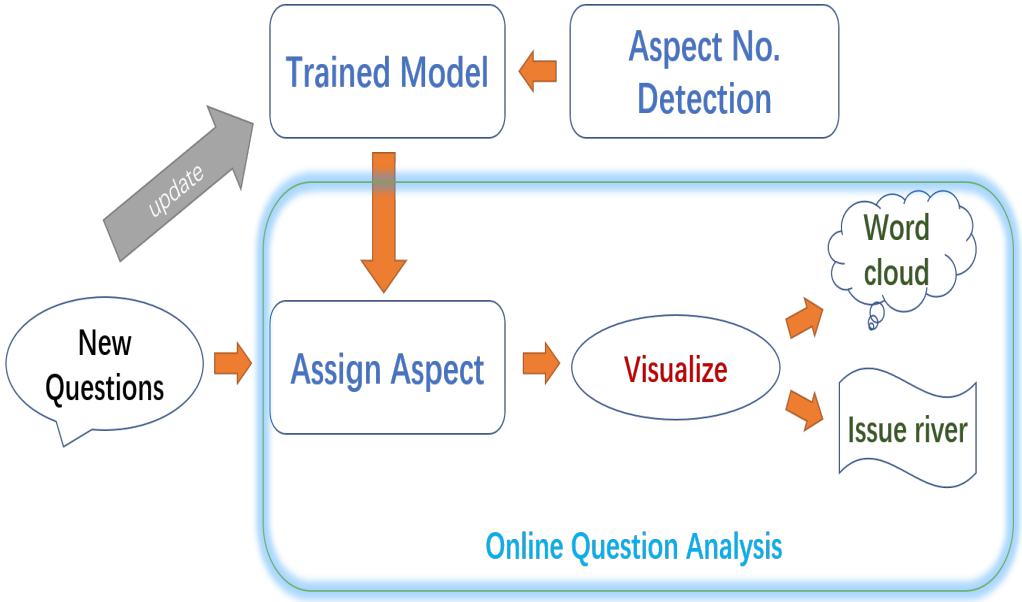

Figure 7: Proposed Online Question Analysis sketch map

To analyze new questions automatically, we want to build an Online Question Analysis tool, which is an automatic method to acquire and process developers' questions in real time as questions arrived continuously. As shown in Figure 7, Online Question Analysis takes current questions as input, and test the input through a well trained aspect extraction model to assign aspect names. Due to the functionality of IDEA model, topics in phrase level and sentence level are also generated. The result is judged and analyzed by topic coherence. Then the results are passed to the visualization part. Combined with the results of previous questions, we are able to build word clouds and issue river to show the trend of aspects in a user friendly demonstration. In this way, the viewers can directly understand what are the current issues captured by Online Question Analysis. To improve the tool, since deep learning is fast developing and may appear some totally new topics and discussions in the future, in a certain time period the model should be retrained with all questions available to make sure the model is not out-of-date. This makes the whole tool reusable and online implemented.

Compared to other API developed by StackExchange or some other website, our tool can give more details of aspects and show information in a more readable graph. What's more, it can always be improved in the future like adding emerging issue with detailed case study functionality.

3. RELATED WORK

Many researches have tried various of approaches to develope aspect extraction models. Initially, since the deep learning methods are not developed enough, like many other NLP problems, extraction methods are mainly based on manually defined rules. Hu and Liu (2004) tried to extract different features by finding the frequency of nouns and noun phrases[13]. Wordnet, which includes a lot of synonyms and antonyms, was used to extract opinions. After that, a great deal of methods have been proposed based on frequency detection. The main disadvantage of these methods is their performance highly depends on the predefined rules. If the data set is not constrained to have a small group of nouns in the rules, or have a lot of massive words, these approaches fail to analyze topics well.

Supervised learning methods takes labeled data as input, and tend to improve the accuracy between predicted labels and true labels. Famous supervised models for aspect extraction are hidden Markov models (HMM)[14] proposed by Jin and Ho (2009) and conditional random fields (CRF)[15] proposed by Li et al. (2010). To automatically learn the features, neural network models were proposed for aspect extraction like Wang et al.'s work[16] (2016). Besides the disadvantage that it is hard for supervised learning developers to get large amount of data with accurate labels, these rule-based models usually cannot categorize topics in high enough accuracy. And usually, the results of these models are not good enough.

To avoid the reliance on labeled data, unsupervised methods are develop-

ing rapidly these years. These models generally output several aspect terms with words distributions to these aspects to help understand the concept of aspects. Among all unsupervised methods, LDA (Blei et al., 2003) and its variants are the most popular ones. LDA is a generative statistical model which allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. In 2015, Wang et al. proposed a restricted Boltzmann machine (RBM)-based model[17], which aims to simultaneously extract aspects and relevant sentiments of a given review sentence. This model assumes that aspects and sentiments are separate hidden variables in RBM. However, it requires some prior knowledge such as sentiment lexicons, which are usually treated as non-informative words in question-based analysis. Based on the efforts of a lot of researches, He et al. proposed Attention-based the Aspect Extraction (ABAE) model[1], which not only solves the problem of poor quality of individual aspects in LDA, but also improves the performance of word coherence by implement word embedding before training. Also, attention mechanism are added to emphasize the words which are really meaningful to certain aspects. Each aspect will output with several words used to describe this aspect, which is called aspect terms. The attention weights can also be retrieve easily in the model to evaluate the training effect.

Attention models[18] (Mnih et al., 2014) was first proposed in image classification field to simulate human attention when looking at certain pictures. The assumption is that, when a person is looking at a picture, his or her sight will only focus more on some meaningful part of the picture, which

performs good in image deep learning researches for it decreases the influence of noisy part. This approach has also been proved useful in many other deep learning fields, and now is applied to various natural language processing tasks, especially in machine translation[11] (Bahdanau et al., 2015), sentence summarization[19] (Rush et al., 2015), sentiment classification[20] (Chen et al., 2016), and question answering[?] (Hermann et al., 2015). Instead of using all available information, attention mechanism aims to focus on the most significant information for a task. Therefore, model can automatically ignore massive words in input and focus more on meaningful words. In the model, every word is assigned a initialized attention weight, and the attention weight is one of the parameters to be learned to reduce the final loss. Higher attention weights means more significance.

To identify emerging app issues effectively based on online review analysis, Gao et al. proposed a novel and automated framework IDEA. IDEA takes reviews of different versions as input, and uses a novel method AOLDA (Adaptively Online Latent Dirichlet Allocation), which is employed for generating version-sensitive topic distributions, to track the topic variations over versions. Online Latent Dirichlet Allocation (OLDA) is a classic method for tracking the topic variations of text streams, which models the topics of texts in one time slice based on the topics of the last slice. However, since the length of the reviews may be small, the proposed AOLDA improves OLDA by adaptively combining the topic distributions in previous versions with similarity. The emerging topics are then identified based on the typical anomaly detection method. To make the topics comprehensible, each topic

are labeled with the most relevant phrases and sentences based on an effective ranking scheme considering both semantic relevance and user sentiment. However, IDEA only considers similarity between each reviews, which means it ignores the influence of the recentness of time. And IDEA is built for review analysis, which has a very different sentence interpretation method from question posts.

Compared all these works for aspect extraction, we simulated the Attention-based Aspect Extraction model for its effectiveness and suitable for unlabeled data in term 1. To achieve better results with phrase level and sentence level interpretation, we use IDEA model in term 2 and modify some part of it to overcome the shortcomings. The result are tested by our manually labeled data.

4. METHODOLOGY

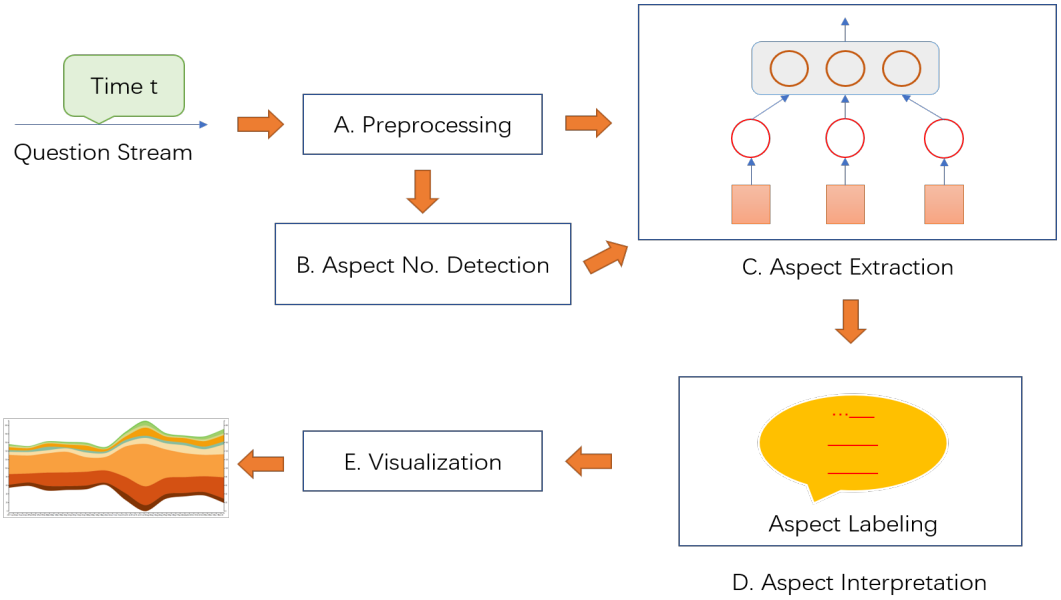

Figure 8: Framework of our model

In this section, we first outline the overall framework of out model in Figure 8. To deal with the raw data (the crawling part is not contained in the Figure 8), the first stage (Part A in Figure 8) preprocesses the raw questions from the question stream to eliminate noisy and non-informative words, and converts misspelled words and words in other forms to the original words. After data preprocessing, the processed data will be send to the Aspect Number Detection (Part B in Figure 8) first to get the most appropriate number for aspect. After that, the preprocessed data are passed to the Aspect Extraction model (used Attention-based Aspect Extraction model in term 1, and uses modified IDEA in term 2) with preferred aspect numbers as parameters, and the model will automatically extract a certain number

(the preferred aspect number parameter) of aspects of the whole data set and distribute significant words to describe each aspect in the second stage (Part C in Figure 8). Then, to interpret the topics (Part D in Figure 8), since the limitation of ABAE in term 1, we manually gave each aspect a brief distribution of a single word or phrase. Since this part is really tedious and time consuming, changing the model to IDEA helps us automatically generate readable interpretation of aspects and even reconstruct sentences. Finally, the model visualize the result by word cloud and issue river (Part E in Figure 8).

4.1. Data Crawling

Though we already have over 8,000 questions provided by StackExchange, it is always better to have more data. StackOverflow is another popular platform for computer science developers and it contains over 11,000 questions under the tag of deep-learning now. We use a python package called scrapy to crawl the data in StackOverflow. We first visit the home page of deep learning related questions in StackOverflow[5], and crawl all the significant information page by page. Comparing our crawling result to the data of StackExchange, we find that for the question part, our data only contain the title, a brief introduction and the linking url of each question. As shown in Figure 9, the "body" part actually simply contains the first few sentences of the whole question, and you can clearly see suspension points at the end of the "body". We need another method to get more information.

[5]https://stackoverflow.com/questions/tagged/deep-learning

```
{"title": "how to split input in a Keras model", "answers": ["1"],
 "body": "\r\nI'm trying to do something that's pretty simple in keras with no
           success. I have an input X with size (?, 1452, 1).
           All I want to do is split this input to a vector of 1450 and a
           vector of 2 and deal ...\r\n       ",
 "url": "/questions/51566693/how-to-split-input-in-a-keras-model"},
{"title": "tf.image.encode_png giving all black output", "answers": ["2"],
 "body": "\r\n           So basically I am using tfr1.9. I am using tf.dataset
           api to import my data.\n
           I am trying to encode a png file as output using the following but
           all my output images are coming out black.  \n\n
           output ...\r\n       ",
 "url": "/questions/51565895/tf-image-encode-png-giving-all-black-output"},
{"title": "Reduce image dimensions in python", "answers": ["1"],
 "body": "\r\n           I have in input an image with dimensions (28, 28, 3).
           I trained a keras model with
           several images with dimensions (28, 28, 1). I want \n   to check a
           single test image with this model,
           but every time I ...\r\n       ",
 "url": "/questions/51563063/reduce-image-dimensions-in-python"},
```


Figure 9: Example of crawing data from home page

The approach we used to catch the whole information of questions was that for every question we crawled, entering the "url" of this question and we could get all information since we are already in the website of this question. To fully interpret the questions, "votes", "views", "title" and other useful attributes are used in IDEA. Since "comments" and other parts of the websites as supplement inputs to our model contains little information with tedious format, so they are not used as input files. And in Figure 10, you can find the whole question "Reduce image dimensions in python", with the whole "body", compared to the last question in Figure 9.

4.2. Preprocessing

Besides the massive noisy words (misspelled words, casual words, repetitive words, and so on) and the words in other forms which are the common

```
{"title": "Reduce image dimensions in python",
 "question": "<div class=\"post-text\" itemprop=\"text\">\r\n\r\n<p>I have in input an
            image with dimensions (28, 28, 3).
            I trained a keras model with several images with dimensions (28, 28, 1). I
            want \n   to check a single test image with this model, but every time I get
            a dimension error. How can I reduce original dimensions (28, 28, 3) to (28,
            28, 1)?</p>\n\n<pre><code>test_image = image.load_img('test/number3.png' ,
            target_size = (28, 28))\ntest_image = image.img_to_array(
            test_image)\ntest_image = np.expand_dims(test_image, axis = 1)\nresult =
            classifier.predict(test_image)\n</code></pre>\n    </div>", "answer": "<div
            class=\"post-text\" itemprop=\"text\">\r\n<p>Depending on how you would like
            to reduce dimensionality you can just choose one of the colour channels like
            this</p>\n\n<pre><code>one_channel_image =
            test_image[:,:,0]\n</code></pre>\n\n<p>or you could find use the mean across
            the colour channels</p>\n\n<pre><code>one_channel_image = np.mean(
            test_image, axis=2)\n</code></pre>\n\n<p>In my experience of ML image
            problems just taking one channel works fine.</p>\n\n<p>If you need to
            increase dimensionality from (28, 28) to (28, 28, 1) you can use
            numpy.reshape</p>\n\n<pre><code>one_channel_image = test_image.reshape((28,
            28, 1))\n</code></pre>\n    </div>"},
```


Figure 10: Example of crawing data after visiting the url

difficulties in all NLP problems, since deep learning questions always contain codes and terminologies, it has some specific challenges when preprocessing. What's more, unlike sentiment analysis, irony detection or other NLP problems, questions about deep learning always contains a lot of codes and websites, and sometimes some sentiment words which simply describes user's feelings, which are regarded as non-informative words in this research. And because the raw data we downloaded and crawled contain html tags (see in Figure 9 and Figure 10), it is difficult for us to determine whether to eliminate these tags (some tags like <code>...</code>, <ol>... contains structure information which may be significant for aspect extraction) or not. In the following, we will introduce the method we found most useful in this research.

4.2.1. Word Formatting

We first convert each word in the data set into lowercase, and then stem each word into its original form. We simply used the lemmatization function in python nltk package[6] for lemmatization. What deserves to be mentioned is that we need to find out the part of speech of each word first, and then do lemmatization differently according to whether the word is noun, verb, adjective, adverb, preposition or other part of speech. For example, if we regard "does" as a noun (every word is regarded as a noun by default), the python package will change it to "doe" rather than "do". All the html tags are remained in this step.

4.2.2. Word Filtering

This filtering step aims to reduce the non-informative words, such as emotional words (e.g., "good" and "bad"), abbreviations (e.g., "asap") and useless words (e.g., "somebody"). Finding this kind of words requires a great deal of manual effort to figure out. Fortunately, these words are formally summarized in python nltk package by other developers by going through thousands of documents, which is denoted as predefined stop words. We delete all the stop words in this step. Some predefined stop words are shown below:

[6]https://www.nltk.org/

Predefined Stop Words: cool, fine, hello, alright, poor, plz, pls, thank,old, new, asap, bit,someone, love, like, annoying, beautiful, dear, app, good, excellent, awesome, please, they, very, too, like, love, nice, yeah,amazing, lovely, perfect, much, bad, best, yup, suck, super, thank,great, really, omg, gud, yes, cool, fine, hello, alright, poor, plz, pls, google, facebook, annoying, beautiful, dear, master, evernote.

4.2.3. Word Replacement

As we mentioned before, the websites in questions are always massive and meaningless, and will distinctly deduct the model performance. We find that websites in the data set always begin with "http://" or "https://", which can be simply removed by regular expression. Similarly, some questions contains images which in the format of <img src="..." ...> in the crawled bodies. We also found there are a great deal of massive variables in the code part, with misleading and intricate expressions and representations, and they are different according to different programming languages. The code part always begin with <code> and end with </code>, which may occupy large space in the whole question. We finally decided to deal with these "code"s too.

Instead of plainly removing all these parts, we think the location where they occurs may have a positive influence to word embedding and further model training, so we use the words below to replace these parts:

Sometimes the question may mention several urls together. To reduce

Non-informative parts	**Replacing words**
Websites (eg: http://..., https://...)	url
All numbers	<num>
Image html tag	img
Code, pseudocode	code
Unknown words in dictionary	<unk>

duplication, we replace duplicate urls into one single url.

4.2.4. HTML Tags Summerization

This part summarize all the html tags occur in the data set. They are all removed since we found that maintaining them would cause the model to clustering aspects according to html stuctures, which leads to a "fake" loss, which we will give a detailed analysis in the Section 5.2.1.

4.2.5. Word Embedding

Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc[7]. And Word2Vec is one of the most popular technique to learn word embeddings using shallow neural network. Figure 11 is an illustration of word2vec.

[7]https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

tags	**description**
 	new line
<hr>	thematic change in the content
	stress emphasis
	important text
<h1>, <h2>, <h3>	define HTML headings
	unordered (bulleted) list
	ordered list
<blockquote>	a section that is quoted from another source
<pre>	a preformatted text
<code>	a code or pseudocode (handled before)
<img src=...>	image (handled before)
<a href=...>	hyper link
<p>	paragraph

We begin word embedding by associating each word w in our vocabulary with a feature vector. We use word embeddings for the feature vectors as word embeddings are designed to map words that often co-occur in a context to points that are close by in the embedding space[10] (Mikolov et al., 2013). This is helpful to learn aspect embedding in the future, which will be explained in detail later in Section 4.4.

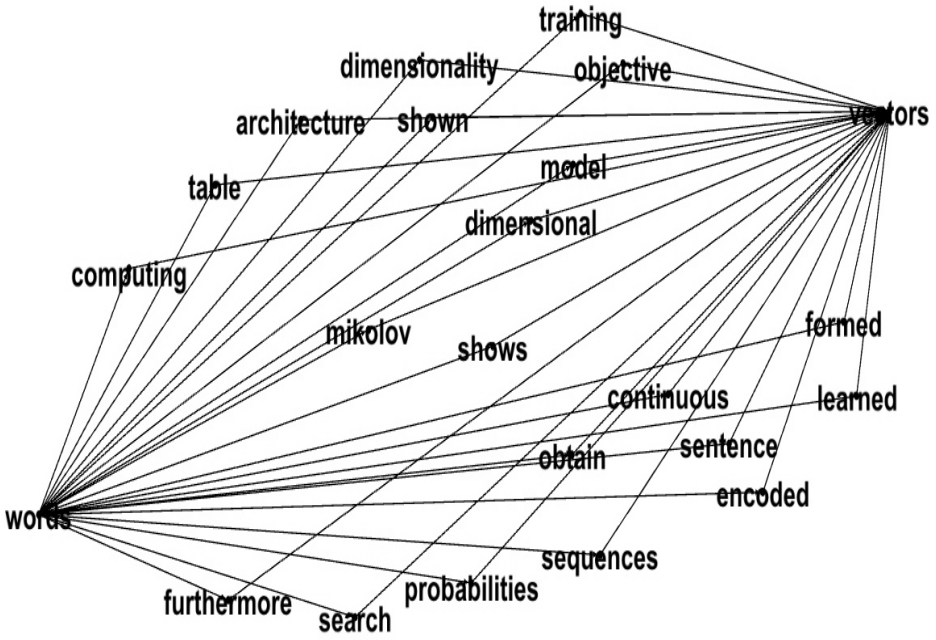

Figure 11: Illustration of word2vec

4.3. Aspect Number Detection

In the Attention-based Aspect Extraction model, we should input expecting aspect number as input parameter to distribute certain number of aspects. We used the method proposed by Gallagher et al., which is called Hierarchical Topic Modeling with Minimal Domain Knowledge[12].

In all the topic number detection part, we use preprocessed data as input.

Aspect modeling by way of Correlation Explanation (CorEx) yields rich aspects that are maximally informative about a set of data[12]. Gallagher et al.'s work mainly focuses on aspect modeling over large corpora. Though

their final purpose may not be helpful to our project, we find they came up with an effective method to compute appropriate aspect number. This method is well packaged in python and can be used directly.

The principle for choosing the number of aspects of the aspect model is to see the contribution of each new aspect to the total correlation (TC). Modern aspect extraction developers assume that each aspect explains a certain portion of the total correlation. To decide how many aspects to choose, we can look at the distribution of each aspect in the result. As a rule of thumb, additional aspects should be added until additional aspects contribute little to the overall total correlation.

The topic correlation can be accessed through the *tcs* attribute in corex-topic python package, and the overall TC (the sum of the topic *tcs*) can be accessed through *tc* attribute.

To demonstrate this approach clearly, we begin with a small number (10), and increase the number gradually to observe the distribution of TCs for each topic to see how much each additional topic contributes to the overall TC. This can be discovered simply through a bar graph. We keep adding topics until additional topics do not significantly contribute to the overall TC.

To get better aspect results, we restart the CorEx topic model several times from different initializations, and chose the topic model that has the highest TC, which is 48. The *tcs* distribution is shown in Figure 12.

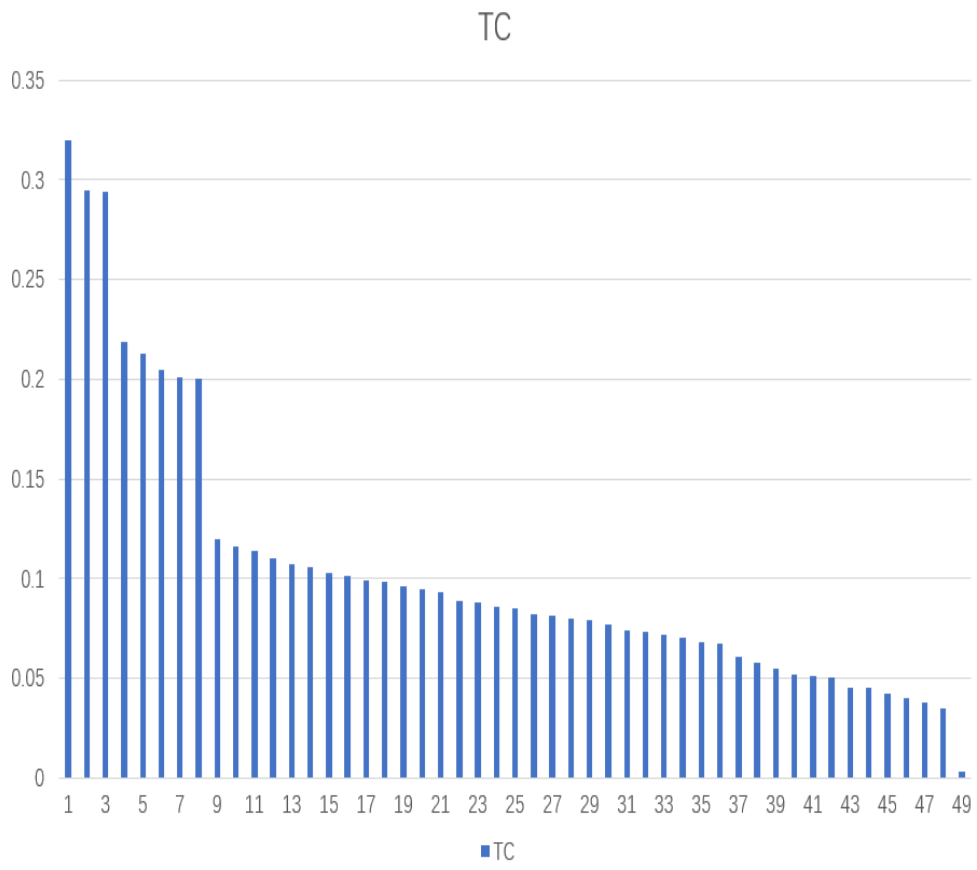

Figure 12: Topic correlation distribution

As you can see in Figure 12, it is not hard to find the *tcs* decrease greatly when aspect number reaches 4, 9, and 49. For the two numbers less than ten, it is good to choose 8 or around to briefly describe all main topics in deep learning, which is easier for us to label the data and explain a overall understanding of deep learning. We labeled our data into 6 categories: Image, NLP, Game-ai, Self-driving, Programming-languages, Reinforcement-learning. It may have a lot of labeling principles, but we decide to divide

data into these 6 categories because of three main reasons: tags from StackOverflow, less overlapping and occurring frequency. 48 is also a reasonable number since every big deep learning topics may contain a lot of sub topics. Finding some of these sub cases is good and interesting for case study, but may cost a lot of time for us to deal with all this sub topics.

Since compared to the *tcs* of aspect 1 to 8, the *tcs* of other aspects seems small, these aspects may tend to be less significant aspects or say sub topics. In term 1, we manually assign some labels to each aspect trained by ABAE model. According to Figure 12, the aspects later should theoretically have fewer frequencies and are more likely to be "Noisy" aspects. This is proved to be true in term 1.

Every time you update the training data set to input into the Attention-based Aspect Extraction model (showed in Figure 8), you should always perform Aspect Number Detection first.

4.4. ABAE Model Description

We describe the Attention-based Aspect Extraction (ABAE) model in this section. The final goal of the model is to learn a certain number of aspects embeddings, where each aspect can be interpreted by looking at the nearest words (representative words) in the embedding space[1]. The illustration of the ABAE model is shown in Figure 13 in page 50.

We begin by associating each word w in our vocabulary with a feature vector $\mathbf{e}_w \in \mathbb{R}^d$, where d is the dimension of word embedding. Word embeddings are always used to describe the correlation between different words by mapping words frequently occur nearly in context with a close points distance in the embedding space. The word embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times d}$, is used to describe the feature vectors associated with the words by row locations, where V is the number of vocabulary size. And an aspect embedding matrix $\mathbf{T} \in \mathbb{R}^{K \times d}$, where K is the number of aspects defined by the Aspect Number Detection mentioned before (K should be much smaller than V), is used to approximate the aspect words in the vocabulary. Attention mechanism is added to filter aspect words.

Each input sample to Attention-based Aspect Extraction model is a list of indexes for words in a question sentence. Shown in Figure 13, attention mechanism first helps us filter away non-aspect words by down-weighting. The weighting parameter a_i will be modified in training to achieve better results. This step makes sure that the words highly related to the aspects will be treated significantly in the model. Then sentence enbedding $\mathbf{z}_s$ is constructed from weighted words embedding. After that, we try to reconstruct the sentence embedding from aspect embedding matrix. This process of dimension reduction and reconstruction preserves most of the information of the aspects words in embedded aspects, where the Attention-based Aspect Extraction model aims to transform sentence embedding of the filtered sentences (which is $\mathbf{z}_s$ in Figure 13) into their reconstructions (which is $\mathbf{r}_s$ in Figure 13) with least possible amount of distortion[1]. Details are explained

in each subsection.

4.4.1. Sentence Embedding with Attention Mechanism

This section is to introduce the details in the attention-based encoder part in Figure 13.

Our goal is to construct a vector representation to capture the most correlated information with regards to the aspect of each question. For each input question q, we construct a vector representation $\mathbf{z}_s$, which is defined as the weight summation of word embeddings $\mathbf{e}_{w_i}$, $i = 1, ..., n$ corresponding to the word indexes in the question in the first step.

$$\mathbf{z}_s = \sum_{i=1}^{n} a_i \mathbf{e}_{w_i} \tag{1}$$

In the attention mechanism, a positive weight a_i is calculated to represent the significant of each word w_i in the question q. In other words, weight a_i can be interpreted as the probability that w_i is the correct word of the predicted main aspect of the question q.

The weight a_i is conditioned on the embedding of the word $\mathbf{e}_w$, as well as the gloabal context of the question:

$$\mathbf{a}_i = \frac{exp(d_i)}{\sum_{j=1}^{n} exp(d_j)} \tag{2}$$

$$d_i = \mathbf{e}_{w_i}^T \cdot \mathbf{M} \cdot \mathbf{y}_s, \tag{3}$$

$$\mathbf{y}_s = \frac{1}{n} \sum_{i=1}^{n} \mathbf{e}_{w_i}^T. \tag{4}$$

In the upper equations, $\mathbf{y}_s$ is simply the average of the word embeddings, which many researchers believe to capture the global context of the question. Matrix $\mathbf{M}$ is a $d \times d$ matrix mapping between the global context embedding $\mathbf{y}_s$ and word embedding $\mathbf{e}_w$. $\mathbf{M}$ is learned and updated through the training process. The idea of these equations are that given a question, we try to construct its representation by the average of its whole word representations. Then two things are concerned to assign the weight: one is filtering the word through the transformation $\mathbf{M}$ which is able to capture the relevance of the word to the K aspects; the other is capturing the relevance of the filtered word to the sentence by taking the inner product of the filtered word to the global context $\mathbf{y}_s$.

4.4.2. Sentence Reconstruction with Aspect Embeddings

In the last section, we have obtained the sentence embedding. This section is to introduce how to compute the reconstruction of the sentence embedding.

As shown in Figure 13, two steps are done in the reconstruction process, which is similar to an autoencoder. We will explain them from bottom to up.

The weight vector over K aspects embeddings, which is denoted as $\mathbf{p}_t$, represent the probability that the input question belongs to the related aspect. The higher the $\mathbf{p}_t$ is , the more likely the question is predicted to this aspect. $\mathbf{p}_t$ can simply obtained by dimensionality reduction of $\mathbf{z}_s$ from d dimensions to K dimensions, and apply a softmax non-linear activation function which yields normalize non-negative weights:

$$\mathbf{p}_t = softmax(\mathbf{W} \cdot \mathbf{z}_s + \mathbf{b}) \tag{5}$$

where $\mathbf{W}$ is the weighted matrix parameter, and $\mathbf{b}$ is the bias verctor. They are both learned as part of the training process.

We are thinking to reconstruct sentence embedding as a linear combination of aspect embeddings from $\mathbf{T}$ intuitively in the upper part:

$$\mathbf{r}_s = \mathbf{T}^T \cdot \mathbf{p}_t \tag{6}$$

where $\mathbf{r}_s$ is the reconstructed vector representation.

4.4.3. Training Objective

Attention-based Aspect Extraction model is trained to minimize the reconstruction error. It adopted the contrastive max-margin objective function used in previous work[21] [22] [23] (Weston et al., 2011; Socher et al., 2014; Iyyer et al., 2016). For every input question, we randomly seletct m questions from

the whole training data as negative samples. We compute the average word embedding of each negative sample and denote it as $\mathbf{n}_i$. To achieve better result, our training objective is to make the reconstructed embedding $\mathbf{r}_s$ as similar to the target question embedding $\mathbf{z}_s$ as possible, but different from those negative samples. To represent this idea, the objective function $J(\theta)$ is shown below:

$$J(\theta) = \sum_{s \in D} \sum_{i=1}^{m} max(0, 1 - \mathbf{r}_s \mathbf{z}_s + \mathbf{r}_s \mathbf{n}_i) \tag{7}$$

where D is the whole training data set and θ is the model parameters, i.e. $\theta = \{\mathbf{E}, \mathbf{T}, \mathbf{M}, \mathbf{W}, \mathbf{b}\}$. $J(\theta)$ is formulated as a hinge loss that minimize the inner product between $\mathbf{r}_s$ and the negative samples at the same time maximize the inner product between $\mathbf{r}_s$ and $\mathbf{z}_s$.

4.4.4. Regularization Term

Since the embedding matrix $\mathbf{T}$ may have redundancy aspects during training, a regularization term should be added to the objective function $J(\theta)$ to ensure the diversity of aspect result:

$$U(\theta) = ||\mathbf{T}_n \cdot \mathbf{T}_n^T - \mathbf{I}|| \tag{8}$$

where $\mathbf{I}$ is an identity matrix, and $\mathbf{T}_n$ is $\mathbf{T}$ with each row normalized to have length 1. To encourage uniqueness, $U(\theta)$ reaches the minimum value when the dot product between any two different aspects embeddings is 0.

Therefore, orthogonality of the aspect embedding matrix $\mathbf{T}$ is encouraged, while the redundancy between different aspect vectors is penalized. So the final objective function L is the summation of J and U:

$$L(\theta) = J(\theta) + \lambda U(\theta) \tag{9}$$

where λ is the weight parameter of the regularization term.

4.5. IDEA Model Description

In this section we introduce IDEA model. IDEA aims to IDentify Emerging App issues effectively based on online review analysis[2]. We want to detect the emerging topics of current versions by considering the topics in previous versions. To interpret the topics, IDEA employs the meaningful phrases and sentences as candidates to label each topic according to their semantic relevance and user sentiment. The topic labels are the identified app issues.

4.5.1. Phrase Extraction

Since phrases (mainly referring to two consecutive words in our research) are employed in IDEA for interpreting topics, they should be extracted in the preprocessing step and trained along with all the other words. We decide to extract common phrases and add underline between these words to make them into one single word. In this way, we can capture the semantics of each phrase just like other single words, based on which we can label the topics with the most relevant phrases. We want the phrases to be meaningful and

comprehensible, so we use a typical phrase extraction method based on PMI (Pointwise Mutual Information)[8], which is effective in identifying meaningful phrases based on co-occurrence frequencies:

$$PMI(w_i, w_j) = log\frac{p(w_i w_j)}{p(w_i)p(w_j)} \tag{10}$$

where $p(w_i w_j)$ refers to the co-occurrence probability of the phrase $w_i w_j$ and $p(w_i)$ and $p(w_j)$ indicates the probability of the word w_i and w_j in the whole review documents. The higher the PMI values are, the more likely the combination of the two words to be a meaningful phrase. We set a threshold for PMI, and phrases with higher PMIs are extracted. Some extracted phrases are shown below:

Extracted phrases: output_layer, dot_product, neural_network, deep_mind, research_paper, initial_state, sigmoid_activation_function, data_set, gradient_descent, artificial_intelligence, non_linear, image_classifier, linear_algebra, computer_vision, loss_function, machine_learn,game_engine, feature_extraction, cost_function, convolutional_neural_network, hide_layer, long_term_memory, cross_entropy,cross_validation, computer_science, tic_tac_toe, activation_function.

4.5.2. AOLDA - Adaptively Online Latent Dirichlet Allocation

OLDA[24] is a classic method to extract different topic from of several text streams, which trains one time slice's topics based on the topics of the last a

[8]https://en.wikipedia.org/wiki/Pointwise_mutual_information

few slices. Adaptively online topic modeling method, AOLDA, is proposed to reduce the influence of short texts and noisy words. The proposed AOLDA improves OLDA by adaptively combining the topic distributions in previous versions.

Denoted the preprocessed reviews as $R = \{R^1, R^2, ..., R^t, ...\}$, where R^t refers to the document in t-th version. As shown in Figure 14 in Page 51, each review is treated as one document. The prior distributions over document-topic (α) and topic-word distributions (β), which determines the topic distributions of the terms in the input document, are defined initially. We want totally K topics. For the k-th topic, ϕ_k^t is the probability distribution vector over all the input terms. For every training approach, we should define a window size w, meaning the number of previous versions to be considered for analyzing the topic distributions of the current version.

For the previous w versions, we gain the topic distribution denoted as $\{\phi^{t-1}, ..., \phi^{t-i}, ..., \phi^{t-w}\}$, for generating the prior β^t of the t-th version. The adaptive integration refers to summing up the topic distributions of different versions with different weights $\gamma^{t,i}$:

$$\beta_k^t = \sum_{i=1}^{w} \gamma_k^{t,i} \phi_k^{t-i} \tag{11}$$

where i denotes the i-th previous version ($1 \leq i \leq w$). The weight $\gamma_k^{t,i}$ is determined by the similarity of the k-th topic between the $(t-i)$-th version

and the $(t-1)$-th version, which is calculated by the softmax function:

$$\gamma_k^{t,i} = \frac{exp(\phi_k^{t-i} \cdot \beta_k^{t-1})}{\sum_{j=1}^{w} \phi_k^{t-j} \cdot \beta_k^{t-1}} \tag{12}$$

where the dot product $\phi_k^{t-i} \cdot \beta_k^{t-1}$ computes the similarity between the topic distribution ϕ_k^{t-i} and the $(t-1)$-th version β_k^{t-1}. The key idea is that the previous versions contributes differently to the topic distribution of the current version.

However, this method only take the similarity between documents into consideration. In reality, hot topics last for months, which means similar topics in consecutive months have a higher change to be related. To achieve this approach, we add exponential decay function to enhance the influence of adjacent versions, reduce the influence of distant months:

$$\mu^t = exp(-nt) \tag{13}$$

where t is the version, and n is a pre-defined exponential decay coefficient.

4.5.3. Anomaly Discovery

The anomaly topics are defined as emerging topics, which have major differences between current document from previous versions. We use similarity to describe the difference between two consecutive versions. The measurement is shown below:

$$D_{JS}(\phi_k^t||\phi_k^{t-1}) = \frac{1}{2}D_{KL}(\phi_k^t||M) + \frac{1}{2}D_{KL}(\phi_k^{t-1}||M) \tag{14}$$

where $M = \frac{1}{2}(\phi_k^t + \phi_k^{t-1})$. This is called Jensen-Shannon (JS) divergence [9], which measures the similarity between the two probability distributions. The Kullback-Leibler (KL) divergence D_{KL} is utilized to measure the discrimination from one probability distribution P to another Q, computed by:

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)} \tag{15}$$

where $P(i)$ is the i-th item in P. The higher the JS divergence is, the larger difference the two topic distributions have.

We set a threshold δ, and for each version, the topics with divergences higher than δ are regarded as emerging topics.

4.5.4. Topic Interpretation

To interpret each topic, we can utilize words, phrases, sentences, or entire questions. This part shows how we extract these items.

Phrase Candidate: We have extracted the phrases in Section 4.5.1. We employ three rules to identify more meaningful phrases: first, the length

[9]https://en.wikipedia.org/wiki/Jensen%E2%80%93Shannon_divergence

of each word in the phrase should be no less than three; second, the phrase should not contain stop words; third, phrase should include at least one noun or verb, and no adverbs (e.g., "greatly") or determiners (e.g., "the").

We consider two aspects: the semantic similarity between the candidates and the topics, and also the user sentiment of the candidates to achieve topic labeling.

Semantic Score: Good topic labels should cover the latent meaning of the topic[2]. The semantic score measures the semantic similarity between the candidate and the topic. A good topic label should be similar to the target topic and different from the other topics in semantics. For phrase, we use the method below:

$$sim(a, \phi_k^t) = -D_{KL}(a||\phi_k^{t-i}) \approx \sum_w p(w|\phi_k^t) log \frac{p(a, w|C)}{p(a|C)p(w|C)} \qquad (16)$$

where $p(w|\phi_k^t)$ is the probability of term w in the topic distribution ϕ_k^t. $p(w|C)$ and $p(a|C)$ refer to the percentages of the terms w and a in the whole post collection C, respectively. $p(a, w|C)$ indicates the co-occurrence probability of the two terms a and w in C. Since sentence have length attribute, we modify the equation to:

$$sim(s, \phi_k^t) = -D_{KL}(s||\phi_k^{t-i}) \approx \sum_w p(w|\phi_k^t) log \frac{p(w|s)}{len(s)p(w|\phi_k^t)} \qquad (17)$$

We combine the $sim(l, \phi_k^t)$ together with the similarity scores to other topics $\sum_{j \neq k} sim(l, \phi_j^t)$. This makes the label l be semantic close to the topic distribution ϕ_k^t and different from other topic distributions:

$$SCORE_{sem}(l, \phi_k^t) = sim(l, \phi_k^t) - \frac{\mu}{K-1} \sum_{j \neq k} sim(l, \phi_j^t) \tag{18}$$

where parameter μ is a pre-defined penalty of semantic similarities from other topics. l can be a phrase a or a sentence s. K is the number of topics we want to extract.

Sentiment Score: we define our own sentiment score function here. We have three attributes for each post l, the length of the post h_l, the rate of the post r_l, and the number of views v_l. The longer the length is, the more likely to provide valuable information. And higher rate and views means better post. The sentiment score is defined as below:

$$SCORE_{sen}(l) = exp(-r \frac{1}{ln(h_l + 1)} \cdot \frac{1}{ln(v_l + 1)} \cdot \frac{1}{ln(r_l + 1)}) \tag{19}$$

only when length, rate, views are all high, the sentiment score will be high.

Overall Score: combine the semantic score and the sentiment score, we have

$$SCORE(l, \phi_k^t) = SCORE_{sem}(l, \phi_k^t) + \lambda SCORE_{sen}(l) \tag{20}$$

where the weight λ is used to balance the two aspects.

4.6. Visualization

In this part, we introduce how we visualize our results. Since the IDEA model can output the attention weight score of each words in each question (which can regard as the importance of this word in a certain aspect). We can build word cloud by regarding attention score as frequency parameter, and group words from different aspects or different time period (results in the same color in the word cloud). There are a lot of tools to build word cloud, including online tools and python packages. We choose online tools called wordart[10] to create word cloud for convenience and it is more vivid and user-friendly.

After training, every new questions can be input as test to predict its label. We put the StackOverflow data in 2017 we crawled to build a issue river. The time flow from January to December is the x-axis in the river, and we define the number of questions predicted to a certain label as the width of this labeled river. Different color of rivers means different aspect. The river data are generated and stored in a csv format file, and can be read and build to issue river automatically be the html tools called "D3"[11]. In

[10]https://wordart.com/

[11]https://d3js.org/

other words, we can always update our data using different test cases and produce the issue river automatically. And thanks to "D3", the river is built in a well-interact mode and shows the trend of each aspect clearly.

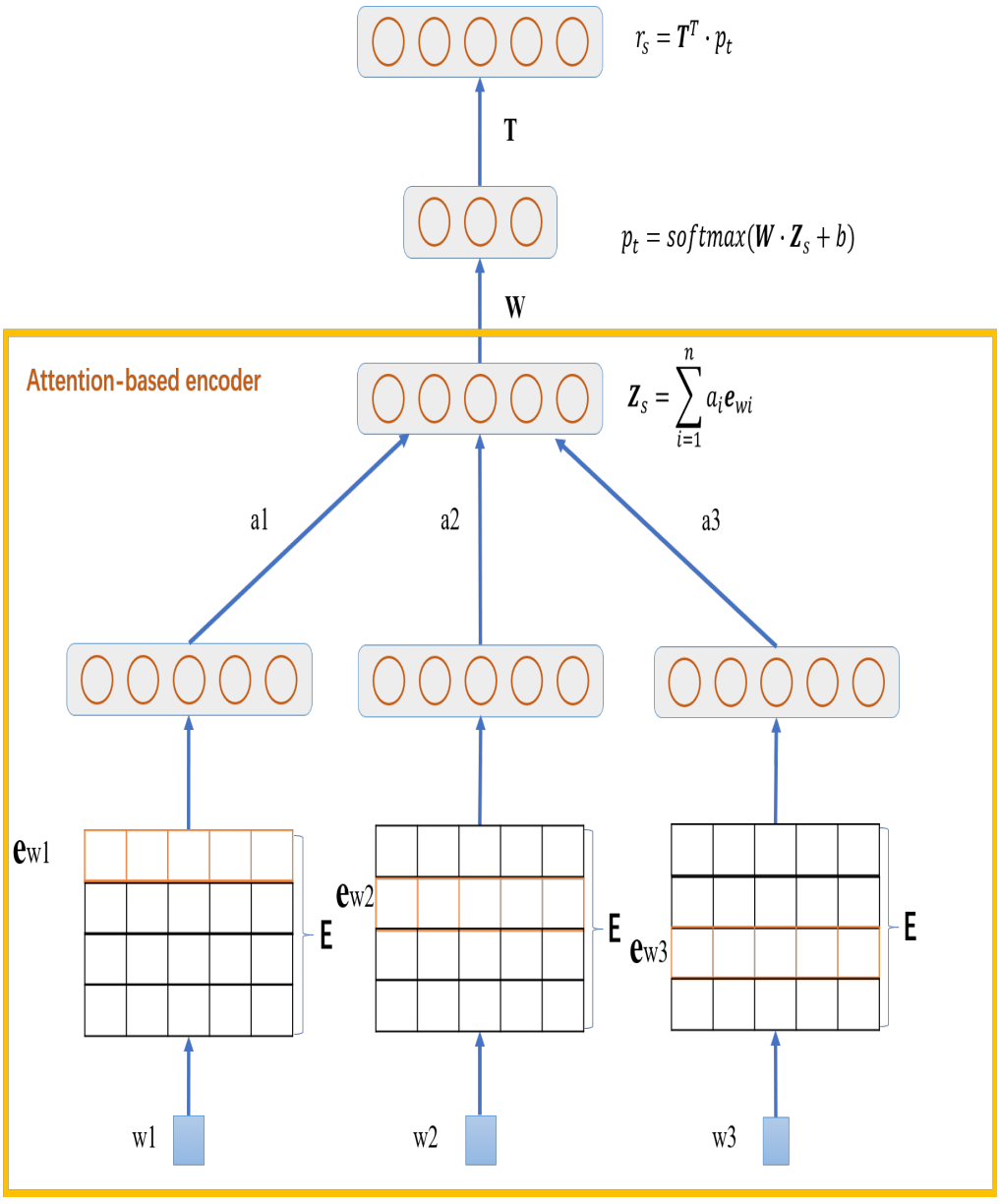

Figure 13: Illustration of ABAE

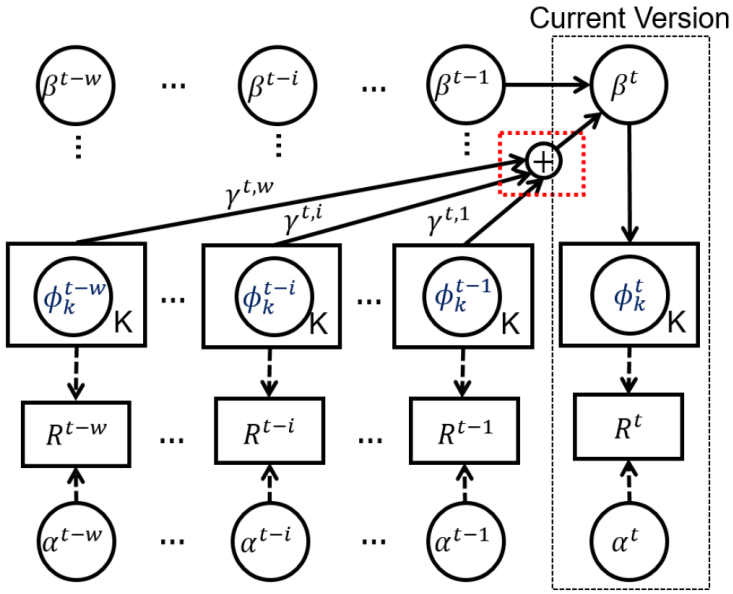

Figure 14: Overview of AOLDA. The red rectangle with dashed dots highlights the adaptive integration of the topics of the w previous versions for generating the prior β in the t-th version. R^t is the review corpus in the t-th version. The dotted lines indicate that we simplify the original LDA steps for clearness.

5. EXPERIMENTATION

This part we evaluate the performance of the Attention-based Aspect Extraction model and IDEA model. Then we visualize the results.

5.1. Dataset

As introduced before, we have two data set: one is downloaded in StackExchange, one is crawled in StackOverflow by ourselves. These two data set are of same format, with create date, votes, answer numbers and so on. We use the data of StackExchange as input, and the data of StackOverflow to do the testing. An example of training data is shown in Figure 15. You can get a glimpse of how massive the data are.

And to create issue river, we divided test data set in 2017 into 12 months according to question creating time, which the number of questions are shown in the table below:

Month	**Question No.**	**Month**	**Question No.**
2017-01	147 questions	2017-07	179 questions
2017-02	113 questions	2017-08	229 questions
2017-03	144 questions	2017-09	179 questions
2017-04	153 questions	2017-10	187 questions
2017-05	136 questions	2017-11	175 questions
2017-06	114 questions	2017-12	189 questions

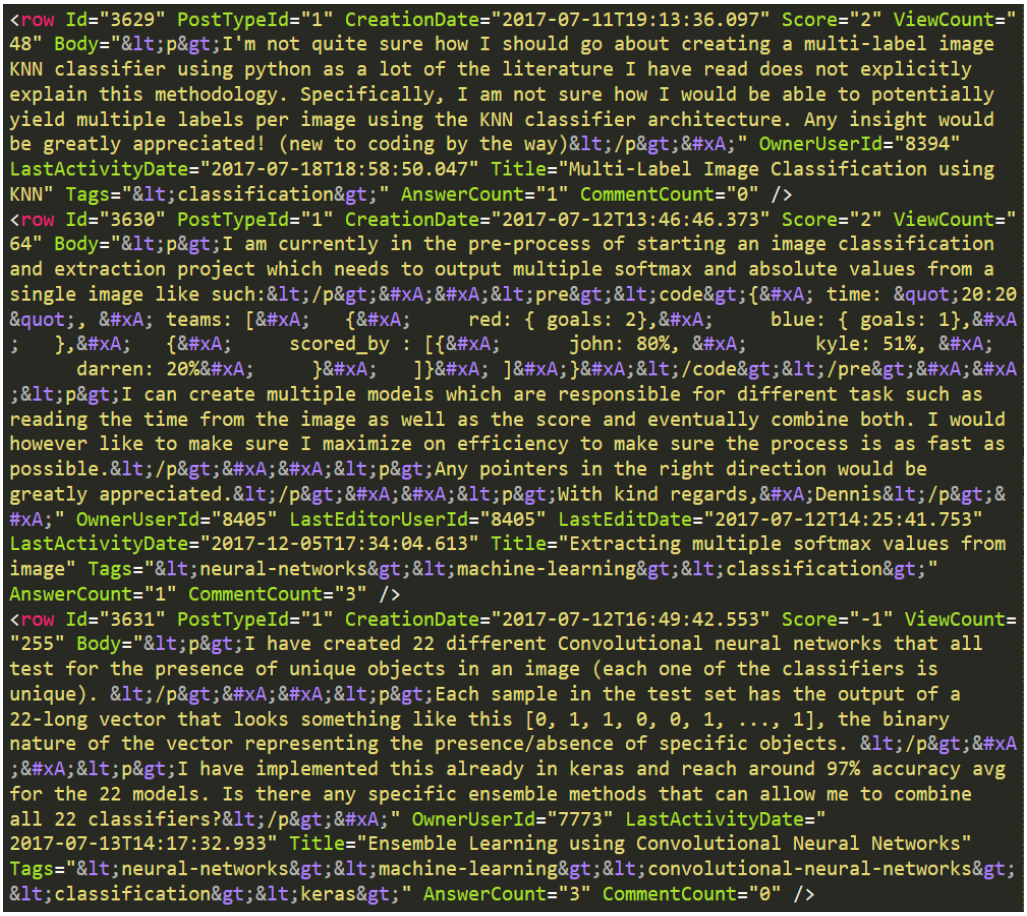

Figure 15: An example of training data

To test the performance of trained topic embeddings, we manually labeled over 500 data with golden labels Image, NLP, Game-ai, Self-driving, Programming-languages, and Reinforcement-learning. We random select 400 of them to train a SVM classifier and use the remaining to test the accuracy, which reflects the performance of topic distribution.

5.2. Training in ABAE

We tried two different preprocessing: one maintains the html tags, the other eliminates all the html tags. The former give us a smaller "fake" loss with massive aspect terms, which we think it is worth to analyze.

5.2.1. Training with Tags and Fake Loss

We input the training data set into the Aspect Number Detection model and it outputs that the most appropriate aspect number is 14 (which is quite different of Section 4.3, because the input here are with tags while the input in Section 4.3 is without tags, and the latter is what we actually used in the end). After that, we input the training data with aspect number parameter 14 into the Attention-based Aspect Extraction model, and get a loss around 8 in the end. Compared to the loss we get after eliminating all tags, this loss is smaller and intuitively it should provide us with more satisfactory result. However, as shown in Figure 16, the aspect terms contain a lot of noisy words and tags which you can not find out what each aspect is talking about even with manually detection with deep learning experience.

To analyze this problem, we first find that html tags are treated as extremely significant words in some aspects. And the words often tend to occur with the tags in same context are also treated as important words in the same aspects according to tags, for instance, "li" (<li> or) with "ol" (<ol> or), "alt" and "png" or "jpg" with "img" (<img scr=...jpg/png alt=...) in Aspect 8 and 9 respectively. What's more, the

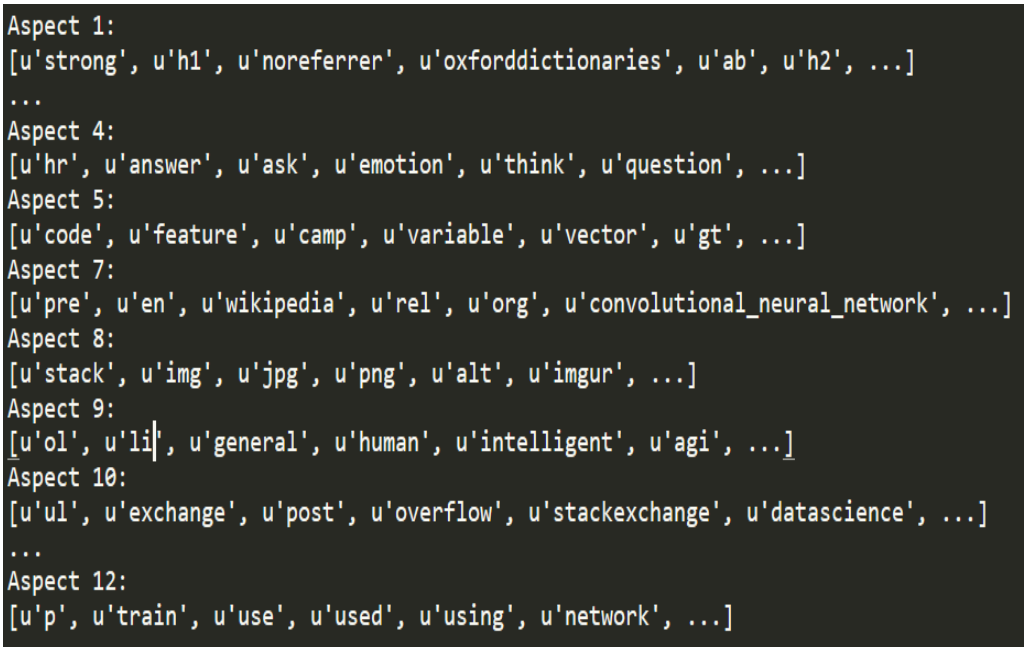

Figure 16: Aspect terms with tags and noisy words

number of aspects we compute in Aspect Number Detection model is similar to the number of different kinds of tags occurred in the data set. Combining all these findings, we claim that because the html tags are maintained in the context with highly regular format, our model tend to treat these tags as golden terms to describe each aspect (actually is html structure). In other words, html structures are learned during this process rather than different topics of deep learning. And since html structures are formal and regular, it leads to a fake lower loss.

To further prove our claim, we input some questions with mainly one certain type of html format (e.g., the questions with a lot of codes and without other html format) as test case, and found that they really output

labels of corresponding aspects distributed to correct html structures as we predicted. Our claim is proved to be true.

This training was abandoned and treated as a lesson to our future reseach.

5.2.2. Training without Tags

After the preprocessing mentioned in Section 4.2 (including eliminating all the html tags), we put the training data set into the Aspect Number Detection model and determined the aspect number to be 48 (see details in Section 4.3). After 20,000 iteration, the loss finally stop at 10. Though the loss was higher than before, we found that the output aspect terms were much better and experienced developers would be able to undertand the meaning of each aspects. Therefore, the result we use in the visualization part are based on the result we achieved here. The details of aspect terms are shown in Section 5.3 below.

5.2.3. Manually Assigning topic description

Since it is not an easy task for newbies to understand aspect only with several describing words which may contain a lot for terminologies, we decide to manually assign topic description to each aspect. It can helpfully decrease the number of duplicated topics and it is much more readable. What's more, this may also help for future work if we want to test the accuracy of topic description, which always needs manually golden aspect assignment as He et al. proposed [1].

Manually assignment is really a tedious work since we should search each word in the topic term and trace them back to the raw training set. Some words tend to have high frequencies in the data set, what we can do is to combine other words in the aspect term and ignore the influence of other unrelated posts. For example, for Aspect 2, "identify" is a common word in developers' questions. However, with the word "graffiti" (which itself has high relation to image problems), we can locate these two words in one certain question: "...Wolfram's Image Identify of graffiti on the wall, but it recognized it as...". After knowing the original sentence, it is much easier for us to understand the true concept of this topic and give a precise description. Similar to Aspect 4, though convnets can also be used in many other fields (which should describe its topic as Deep Learning Model), we find word "smoothness" with it and locate the original question is talking about image classification and mentions "smoothness of color". And combining the fact that convnets are usually used in image related deep learning projects, we decide to describe this topic as "Image Identification". This is how we normally decide to use which label to describe the topics.

However, during the assignment, we find that some aspects seem "useless". It is hard for us to find what they are talking about even with the original questions, or no proper words or phrases can be used to conclude the meaning. For instance, the words with highest score in Aspect 44 is "flu", "south", "sexual", "elasticity", "noob" (not even a word). "Flu" may relate to machine learning prediction, while "south" is hard to locate and "sexual" may related to image classification for sexual pictures. We think we would

better regard it as "Noise" and do not consider visualize this topic in the later Visualization part, so as other topics which are assigned "Noise". And we find that due to the Topic Coherence graph, the topics with larger topic number contributes less and less to the whole coherence, which means that the posterior topics are more likely to be important aspects. In other words, there should be more topics regarded as "Noise" in the posterior part, which is true in the table. What's more, in our prediction, few test cases should be predicted to be "Noise" aspects, since these topics are usually non-sense. This is also the reason why they do not need to show in the visualization part. More information is showed in Section 6.2.

Due to the limitation of space, only some of the important aspect are shown in this paper. For complete 48 aspects please refer to the paper in term 1.

Order No.	**Top words**	**Label**
Aspect 0	Goal, current, player, minimax, state, decision	Decision making algorithm
Aspect 1	Consume, restore, gpu	Storage
Aspect 2	Graffiti, identify	Image Identification
Aspect 3	Artificial, intelligence, resnets, neural	Deep learning model
Aspect 4	Enforcement, convnets, smoothness	Image Identification

Order No.	Top words	Label
Aspect 7	Neural, caffe2, stimulate	Deep learning platform
Aspect 12	Cocke(Cocke-Kasami-Younger algorithm), parsing	NLP
Aspect 13	Data, training, set, test ,model, learning, recognition, algorithm	dataset
Aspect 24	Melfrequency(MFCCs), recalibrate, electric	Voice recognition
Aspect 27	Learning, algorithm, procedural, reinforcement	Learning strategy
Aspect 44	Flu, south, sexual, elasticity, noob	Noise

5.2.4. Simple Test

In many other researchers' study in unsupervised field, they will use some manually labeled data to test the prediction accuracy of their model. However, we find no place to get these kind of labeled data and do not have enough time to label our test data either in term 1, so we do a simple test first in term 1.

To show that our model make sense to some extend, we test the prediction labels of questions in 2018 March as test input, and some of the matching case are shown in Figure 17. The upper post (Id=5478) is predicted to be Aspect 0, which we labeled it as "Decision making algorithm"; and the second post (Id=5570") is predicted to be Aspect 3, denoted as "Deep learning model".

These are only two samples of all the correct predictions.

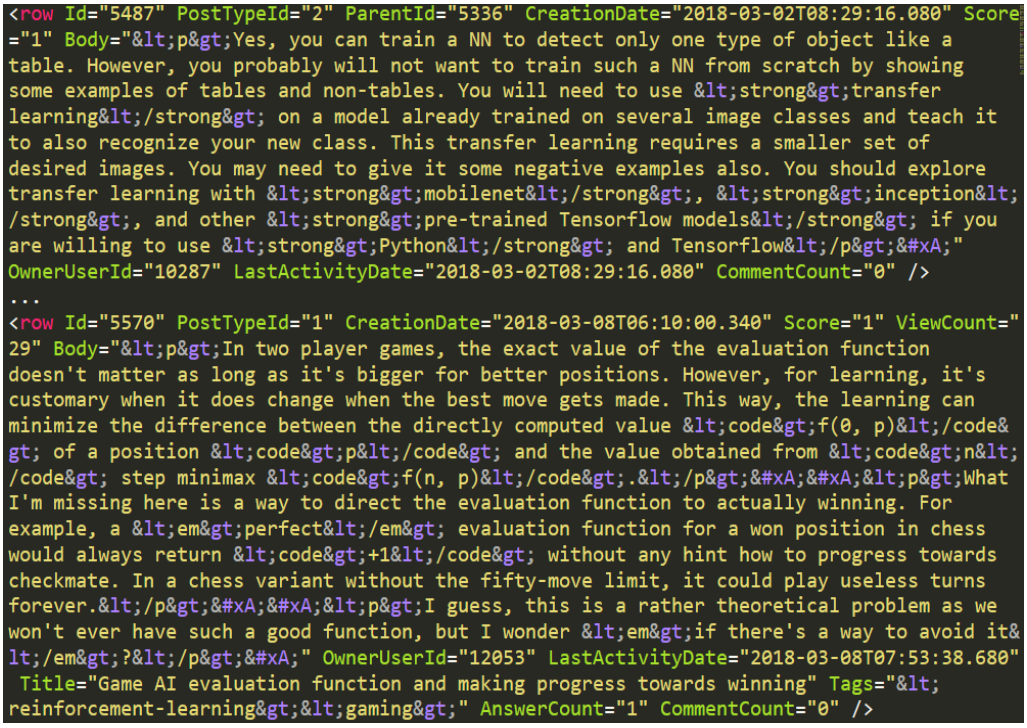

Figure 17: The "Id=5478" post is predicted to be "Decision making algorithm", while the "Id=5570" post is predicted to be "Deep learning model".

To evaluate further, we analyzed the attention weight of test questions to see whether the attention mechanism works well. Important words which may be selected to be aspect terms should have a higher weight. One example is shown in Figure 18. Though "data", "error" both can be important words with high weights in other questions, attention mechanism successfully focus more on "supervised" and "classification" since this question is asking about "learning strategy".

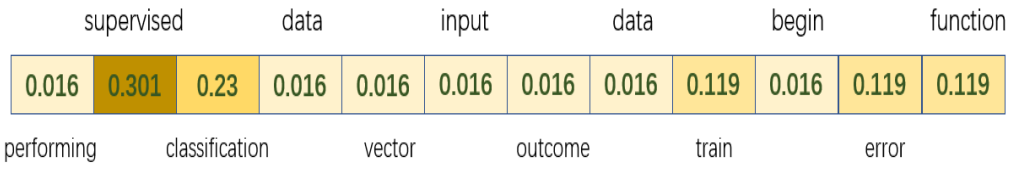

Figure 18: The attention mechanism focus more on "supervised" and "classification" since it is predicted to be "learning strategy".

5.3. Training on IDEA

The training steps are similar to what we have done on ABAE model, but we apply complete test approaches. This section shows the power of IDEA and how our new approaches improve IDEA.

5.3.1. Topic Coherence Analysis

Since we get several words to describe each topic in our result, we use an open source topic coherence analysis tool[12] to analyze the relevance between these trained describing words. Some of the aspects with good describing words are shown below:

(0.53) move_right state_space enough_information next_state initial_state

(1.57) make_decision long_term_goal current_knowledge deep_blue short_term

(0.40) play_chess evolutionary_game_theory complex_task combinatorial_game artificial_intelligence

(0.58) hide_layer input_layer output_layer input_space input_image

[12]https://github.com/jhlau/topic_interpretability

We add exponential decay and views as new features to IDEA model, and the following table shows the improvement of our approach.

ABAE	**IDEA**	**IDEA+decay**	**IDEA+views**	**IDEA+views+decay**
0.069	0.133	0.166	0.208	0.209

This shows adding exponential decay and views as features both improve the performance of the model greatly. The performance of the model reaches the top when exponential decay and views as features are both added. According to the result, we conclude that we propose a new model with better performance compared to IDEA.

5.3.2. Topic embedding Analysis

As mentioned in Section 4.5.2, IDEA will generate a topic embedding to represent the topic distribution of each training data. The topic embedding can be regarded as features and trained in a classifier. In this part, we input over 500 manually labeled data and get the corresponding topic distribution as features, and then put the features and golden label into a SVM classifier to see the result. We randomly select 400 data for training and the remaining for testing. The comparison of the testing accuracy of different models are shown below:

The details of the prediction accuracy and recall of IDEA and IDEA+decay

ABAE	IDEA	IDEA+decay
0.50	0.86	0.87

is shown below:

IDEA:									
	predict_label								
Gold_label		Image	NLP	Game-ai	Self-driving	Programming-lang	Reinforcement-lea	sum	Precision
	Image	8	1	0	0	0	0	9	0.89
	NLP	2	13	1	0	3	0	19	0.68
	Game-ai	1	0	29	2	1	2	35	0.83
	Self-driving	0	1	0	16	0	0	17	0.94
	Programming-lar	0	1	0	0	11	0	12	0.92
	Reinforcement-le	0	1	1	0	0	12	14	0.86
	sum	11	17	31	18	15	14	106	
	Recall	0.73	0.76	0.94	0.89	0.73	0.86		0.86
IDEA+decay:									
	predict_label								
Gold_label		Image	NLP	Game-ai	Self-driving	Programming-lang	Reinforcement-lea	sum	Precision
	Image	7	0	0	0	0	0	7	1.00
	NLP	3	16	1	0	1	1	22	0.73
	Game-ai	1	0	30	0	1	4	36	0.83
	Self-driving	0	0	0	17	0	0	17	1.00
	Programming-lar	0	1	0	1	12	0	14	0.86
	Reinforcement-le	0	0	0	0	0	8	8	1.00
	sum	11	17	31	18	14	13	104	
	Recall	0.64	0.94	0.97	0.94	0.86	0.62		0.87

Figure 19: The testing result of IDEA (upper one) and IDEA+decay (lower one)

We can conclude that IDEA has a much better performance than ABAE. Exponential decay also improves the performance of topic distribution.

6. VISUALIZATION

This section introduces how we visualize our result.

6.1. Word Cloud

The method how we create the world cloud is explained in details in Section 4.5.

For the word cloud generated by ABAE (shown in Figure 20), the input words are from 12 different colors, which represent 12 months in 2017. The size of the words are decided by the overall aspect scores of words in that month, the higher the larger.

In the word cloud, we can find some interesting features. For the several largest words in the word cloud, which are "tensorflow", "ai", "numpy", "caffe", "robotics", "word2vec" and so on, can be regarded as significant words in all the questions posted in 2017, and this obeys our experience of deep learning field. It is interesting that in March (blue words), "panda" is the largest word and we may not understand why without looking back to the raw questions. Actually "panda" is a planning system allows to solve different kinds of planning problems. The planning algorithm for all these problems is a hybrid planning approach, which fuses hierarchical planning with causal reasoning. In March, new "panda" version was proposed with some new features and bugs, so it was asked frequently. Combining the situation that developers asked various kinds of questions in March so questions in other

Figure 20: the word cloud of 2017, where different color means different month

field were sparsely distributed, "panda" was regarded as the most important feature in March.

We want to further mention one word, which is "sophia" in brown (October). The word may be too small to see in the whole word cloud, so you may see it in Figure 21, which is just a part extraction of the larger word cloud. As mentioned in the introduction part, "sophia" is the first AI women robot which was given citizenship by Kingdom of Saudi Arabia in October, which arose a hot discussion about artificial intelligence.

Similarly, we can also build word according to the result of IDEA. For example, for January 2017, we select top 5 words in the top 10 topics to see

Figure 21: word "sophia"

the popular discussion of that month.

Figure 22: the word cloud of January 2017, where different color means different topics

As shown in Figure 22, the hottest topic in that month is image classifier, related with long term goal, non-linear and other information. What's more, in the red circle, we can find a name of game: tic tac toe. This game is always

used as an elementary task in deep learning. This means in that month there was a hot discussion about tic tac toe.

6.2. Issue River

All the questions in 2017 are input into the model as test case to predict the labels manually assigned in Section 5.3. In our assumption, the questions predicted to be "Noise" are regard as wrong prediction, since a question certianly has its meaning in deep learning field and should not be labeled as "Noise". The actual result shows that only a few questions will be labeled as "Noise" in our model, the numbers of which are always smaller than 10. And we find that only 9 aspects ("Decision making algorithm", "Image identification", "Deep learning model", "Deep learning platform", "NLP", "Learning strategy", "Deep learning papers", "Tools", and "Dataset") always occur at least once in each month and occupy the large proportion of all the questions, while the total summation of other aspects including "Noise" occupies only around 10 percent of the whole questions. To demonstrate in a clear and distinct format, only the nine aspects are shown in Figure 19. The x-axis is the time from 2017 January to 2017 December. The width of each river is the number of questions predicted to be certain label in each month. The labels from top to bottom in Figure 23 are in this order: NLP, Image identification, Tools, Deep learning platform, Learning strategy, Deep learning papers, Decision making algorithm, Dataset, Deep learning model.

When you move your mouse on a certain river, the label and number of questions will list in the left-up corner, as shown in Figure 24.

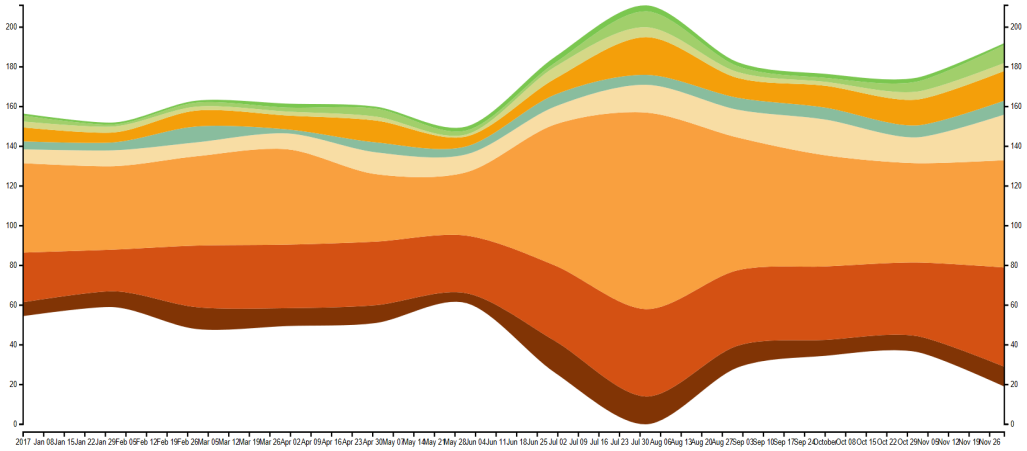

Figure 23: Issue river of 2017

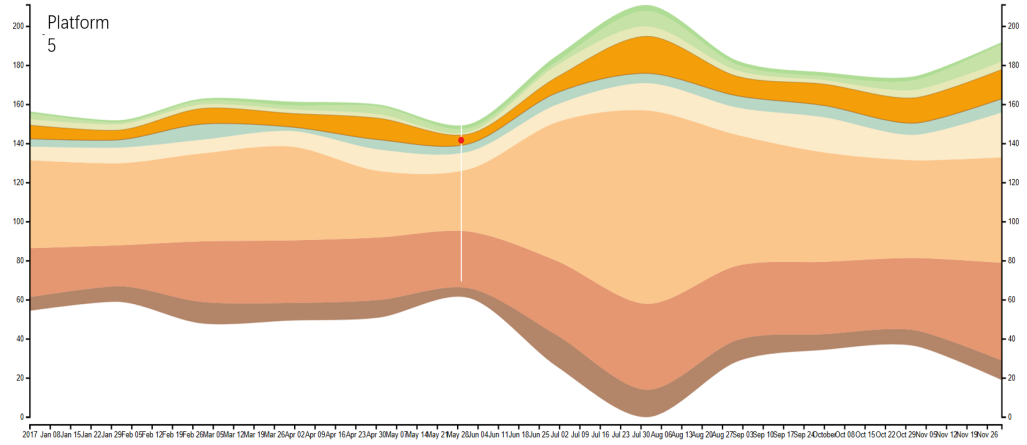

Figure 24: Mouse on red point

In the river you can easily catch the trend of aspects. Most of the questions were about “Decision making algorithm”, “Dataset”, and “Deep learning model”. Developers tended to ask more questions after August and so on.

In term 2, we try to improve our issue river for not only the user interface, but also the emerging issues detected by IDEA. Two cases are shown below.

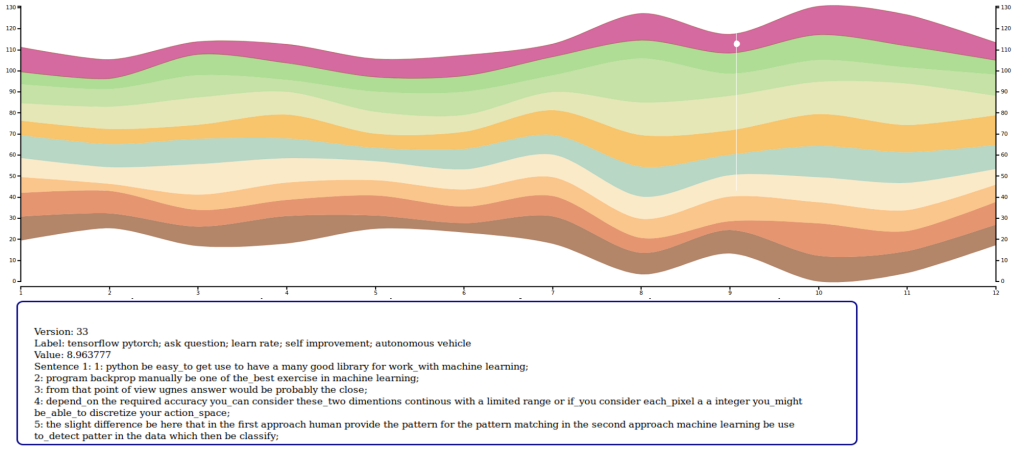

Figure 25: Mouse on the white point (Version 33, purple part). Labels: TensorFlow, PyTorch, learning rate

As shown in Figure 25, the labels are first talking about TensorFlow and PyTorch, which refers to sentence 1 about python libraries for machine learning. The next label is learning rate. It refers to sentence 2 which is about backpropogation. This means this topic contains both information of platforms and learning algorithms. There is no emerging issue in this version.

Figure 26 shows an emerging issue about using swarm intelligence to do self-driving, and compare with human drivers, which is shown in the yellow part.

These two cases shows that issue river can help us analyze the topic in a

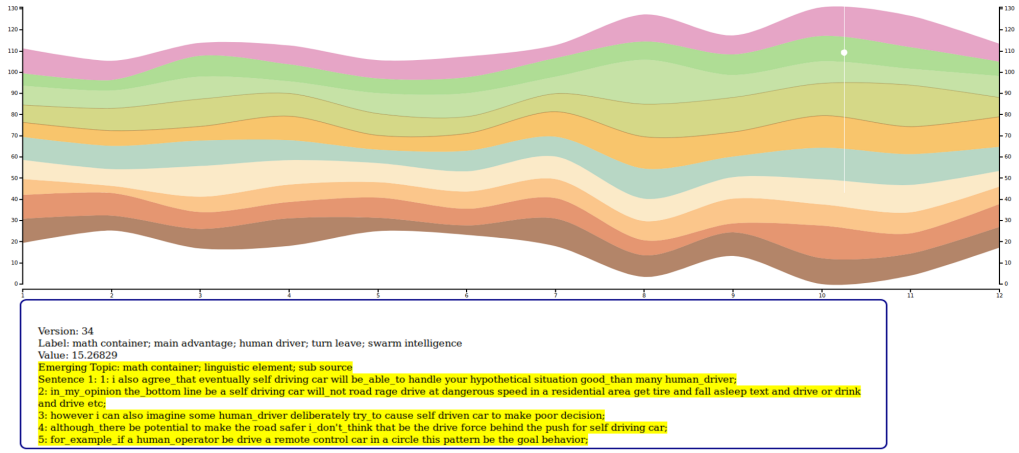

Figure 26: Mouse on the white point (Version 34, green part). Labels: math container, human driver, swarm intelligence

certain time period and find emerging issues easily.

7. FUTURE WORK

These works can be done in the future:

- Try to analyze and provide more details of emerging issues, like showing the accepted answers and links to the origin pages.
- Use our model to deal with other NLP problems, like analyzing reliability of different deep learning tools and platforms.
- Develop a more user-friendly interface.

8. REFERENCE

[1] He, Ruidan, et al. "An unsupervised neural attention model for aspect extraction." Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vol. 1. 2017.

[2] Gao, Cuiyun, et al. "Online app review analysis for identifying emerging issues." 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). IEEE, 2018.

[3] Lau, Jey Han, David Newman, and Timothy Baldwin. "Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality." Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics. 2014.

[4] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research 3.Jan (2003): 993-1022.

[5] Titov, Ivan, and Ryan McDonald. "Modeling online reviews with multi-grain topic models." Proceedings of the 17th international conference on World Wide Web. ACM, 2008.

[6] Brody, Samuel, and Noemie Elhadad. "An unsupervised aspect-sentiment model for online reviews." Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2010.

[7] Zhao, Wayne Xin, et al. "Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid." Proceedings of the 2010 Conference on Empirical

Methods in Natural Language Processing. Association for Computational Linguistics, 2010.

[8] Mukherjee, Arjun, and Bing Liu. "Aspect extraction through semi-supervised modeling." Proceedings of the 50th annual meeting of the association for computational linguistics: Long papers-volume 1. Association for Computational Linguistics, 2012.

[9] Mimno, David, et al. "Optimizing semantic coherence in topic models." Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, 2011.

[10] Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations." Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2013.

[11] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

[12] Gallagher, Ryan J., et al. "Anchored correlation explanation: topic modeling with minimal domain knowledge." arXiv preprint arXiv:1611.10277 (2016).

[13] Hu, Minqing, and Bing Liu. "Mining and summarizing customer reviews." Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004.

[14] Jin, Wei, Hung Hay Ho, and Rohini K. Srihari. "A novel lexicalized HMM-based learning framework for web opinion mining." Proceedings of the 26th annual international conference on machine learning. ACM, 2009.

[15] Li, Fangtao, et al. "Structure-aware review mining and summarization." Proceedings of the 23rd international conference on computational linguistics. Association for Computational Linguistics, 2010.

[16] Wang, Wenya, et al. "Recursive neural conditional random fields for aspect-based sentiment analysis." arXiv preprint arXiv:1603.06679 (2016).

[17] Wang, Linlin, et al. "Sentiment-aspect extraction based on restricted boltzmann machines." Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Vol. 1. 2015.

[18] Mnih, Volodymyr, Nicolas Heess, and Alex Graves. "Recurrent models of visual attention." Advances in neural information processing systems. 2014.

[19] Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." arXiv preprint arXiv:1509.00685 (2015).

[20] Chen, Huimin, et al. "Neural sentiment classification with user and product attention." Proceedings of the 2016 Conference on Empirical Methods

in Natural Language Processing. 2016. [?] Hermann, Karl Moritz, et al. "Teaching machines to read and comprehend." Advances in Neural Information Processing Systems. 2015.

[21] Weston, Jason, Samy Bengio, and Nicolas Usunier. "Wsabie: Scaling up to large vocabulary image annotation." IJCAI. Vol. 11. 2011.

[22] Socher, Richard, et al. "Grounded compositional semantics for finding and describing images with sentences." Transactions of the Association of Computational Linguistics 2.1 (2014): 207-218.

[23] Iyyer, Mohit, et al. "Feuding families and former friends: Unsupervised learning for dynamic fictional relationships." Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016.

[24] Loulwah AlSumait, Daniel Barbar, and Carlotta Domeniconi. On-line LDA: Adaptive Topic Models for Mining Text Streams with Applications to Topic Detection and Tracking. In: Proceedings of the 8th IEEE International Conference on Data Mining, ICDM 2008, December 15-19, 2008, Pisa, Italy. 2008, pp. 312.