

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.

In this paper, we crawl over 7,000 questions about deep learning in Stack-Overflow, and simulate the Attention-based Aspect Extraction^[1] model to extract common aspects embedded in these questions. Since questions have its own distinguishing features such as containing html tags and non-informative words, a specific natural language method is proposed. To figure out the most appropriate number of embedded aspects, we propose to combine the Hierarchical Topic Modeling with Minimal Domain Knowledge^[10]. Finally, we evaluate the model performance qualitatively and visualize the results to facilitate readers' observation.

Keywords: deep learning, unsupervised learning, aspect extraction

Contents

1	INTRODUCTION	5
2	BACKGROUND AND MOTIVATION	11
2.1	Popularity of deep learning	11
2.2	Online Question Analysis	15
3	RELATED WORK	17
4	METHODOLOGY	20
4.1	Data Crawling	21
4.2	Preprocessing	22
4.2.1	Word Formatting	23
4.2.2	Word Filtering	24
4.2.3	Word Replacement	24
4.2.4	HTML Tags Summerization	25
4.2.5	Word Embedding	26
4.3	Aspect Number Detection	27
4.4	Model Description	30
4.4.1	Sentence Embedding with Attention Mechanism	32
4.4.2	Sentence Reconstruction with Aspect Embeddings	33
4.4.3	Training Objective	35
4.4.4	Regularization Term	35
4.5	Visualization	36
5	EXPERIMENTATION	39

5.1	Dataset	39
5.2	Training	40
5.2.1	Training with Tags and Fake Loss	41
5.2.2	Training without Tags	43
5.3	Manually Assigning topic description	43
5.4	Simple Test	49
6	VISUALIZATION	51
6.1	Word Cloud	51
6.2	Issue River	53
7	FUTURE WORK	56
8	REFERENCE	57

1. INTRODUCTION

Deep learning is a class of machine learning algorithms that¹:

- 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 9,000 questions from developers all over the world. Under this situation, it is normal to see questions such as “I’m new to this field, ...”, which to some extent shows that many new developers tend to enter this field and ask some basic questions.

We think it is significant and necessary for the “newbies” to understand the history of development and what this field is really talking about. Also,

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



Figure 1: The “deep-learning” tag in StackOverflow.

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.

Questions posted by developers directly reflect the focus and common concerns of the deep learning field. For example, the “high-vote” (or “high-score”) questions are more likely to be classical topics and recent questions may refer to new emerging issues. As an example, in 2017 October, 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^[2] (LDA) (Blei et al., 2003) and its variants^{[3][4][5][6]} (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”^[7] (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 document, 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-based 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 attention 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)^[10] 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.

For data set, we find that StackExchange has already collected their questions related to deep learning and the data are available on the Internet². This publicly-available data contain “posts”, “tags”, “votes” and many other information about every question from 2016 August to 2018 March, including over 5,500 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. 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” are important to show the quality of the the posts, as good questions

²<https://archive.org/download/stackexchange>

always get a higher votes, but this is not included in this semester due to the time limit.

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 seven 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 7,000 questions about deep learning in StackOverflow
- Use Hierarchical Topic Model to detect appropriate aspect number in a corpus
- Simulate the Unsupervised Neural Attention Model for Aspect Extraction^[1] and learn the aspects embedded in deep learning related questions
- 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 2 and Figure 3 are all generated in Google trends³, 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 4, deep learning related researches occupies a large proportion of all the publications in Tsinghua University. Hundreds of questions about deep learning are asked in the Internet, with a lot of hot discussions.

³<https://trends.google.com/trends/>

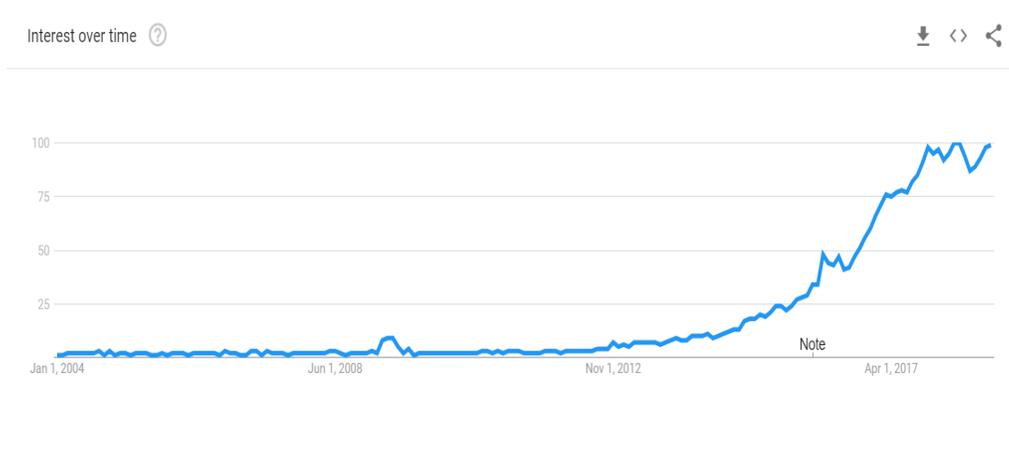


Figure 2: Numbers represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. A score of 0 means there was not enough data for this term.

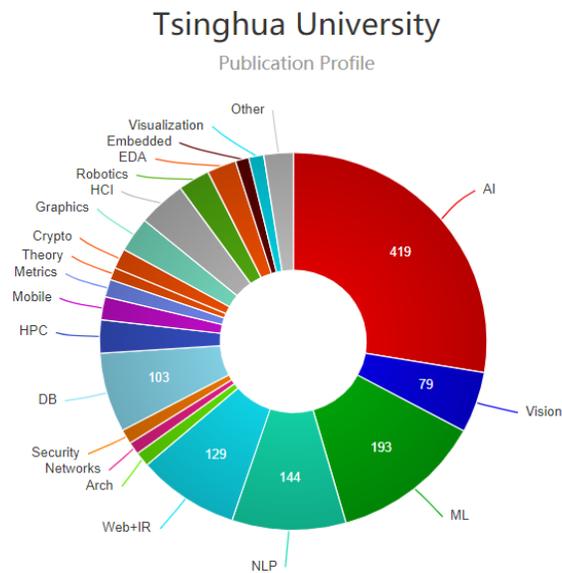


Figure 4: Publication Profile Distribution of Tsinghua University

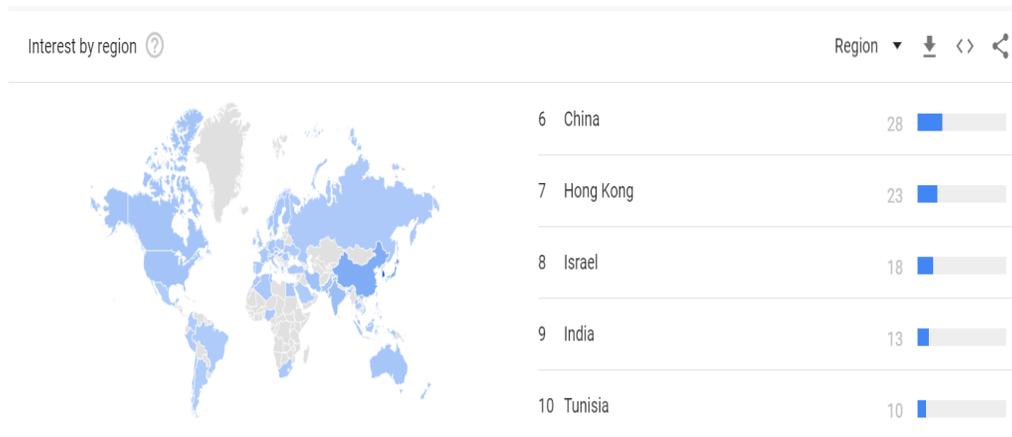


Figure 3: Values are calculated on a scale from 0 to 100, where 100 is the location with the most popularity as a fraction of total searches in that location, a value of 50 indicates a location which is half as popular. A value of 0 indicates a location where there was not enough data for this term.

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 5. And in Figure 5, 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 and image classification. And it is also interesting and helpful to know the developing history of deep learning.

For deep learning experts, knowing the newest information of deep learn-

Deep Belief Networks vs Convolutional Neural Networks

Ad closed by Google

Report this ad

AdChoices 

谷歌广告

-
- 34

I am new to the field of neural networks and I would like to know the difference between Deep Belief Networks and Convolutional Networks. Also, is there a Deep Convolutional Network which is the combination of Deep Belief and Convolutional Neural Nets?
- 23

This is what I have gathered till now. Please correct me if I am wrong.
- ★

For an image classification problem, **Deep Belief networks** have many layers, each of which is trained using a greedy layer-wise strategy. For example, if my image size is 50 x 50, and I want a Deep Network with 4 layers namely

 1. Input Layer
 2. Hidden Layer 1 (HL1)
 3. Hidden Layer 2 (HL2)
 4. Output Layer

Figure 5: Question of deep learning asked by "newbie".

ing 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.

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

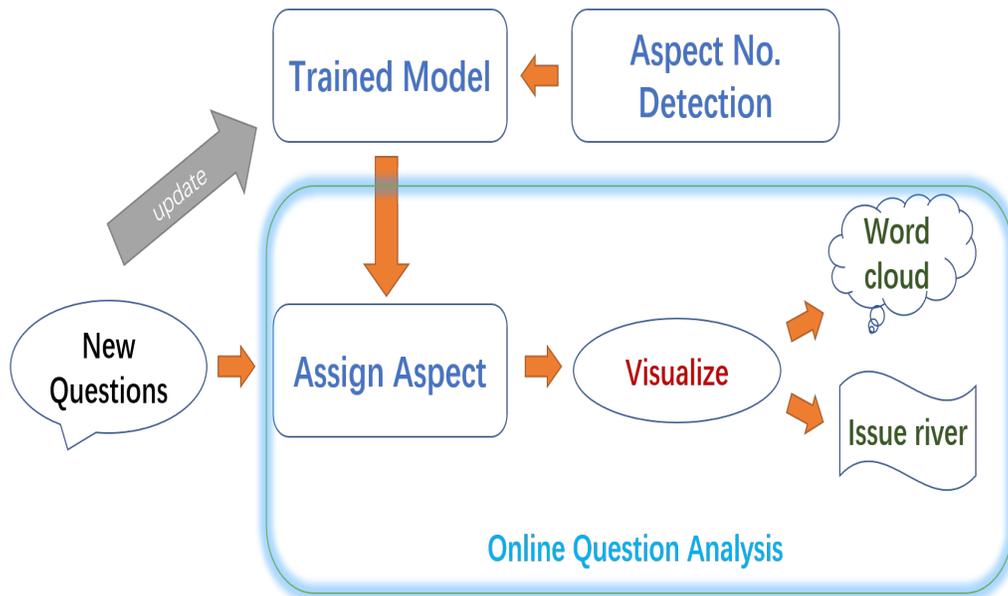


Figure 6: Proposed Online Question Analysis sketch map

To analyze new questions automatically, we want to build an Online Question Analysis tool, which is an automated way to acquire and process developers' questions in real time as questions arrived continuously. As shown in Figure 6, Online Question Analysis takes current questions as input, and test the input through a well trained aspect extraction model to assign aspect names. 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. However, the update part can only be done manually in this semester.

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 detection functionality.

3. RELATED WORK

Many researches have tried various of approaches to developer 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^[11]. 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)^[12] proposed by Jin and Ho (2009) and conditional random fields (CRF)^[13] 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^[14] (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.

To avoid the reliance on labeled data, unsupervised methods are developing 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^[15], 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^[16] (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 influ-

ence 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^[9] (Bahdanau et al., 2015), sentence summarization^[17] (Rush et al., 2015), sentiment classification^[18] (Chen et al., 2016), and question answering^[7] (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.

Compared all these works for aspect extraction, we finally decide to simulate the Attention-based Aspect Extraction model for its effectiveness and suitable for unlabeled data.

4. METHODOLOGY

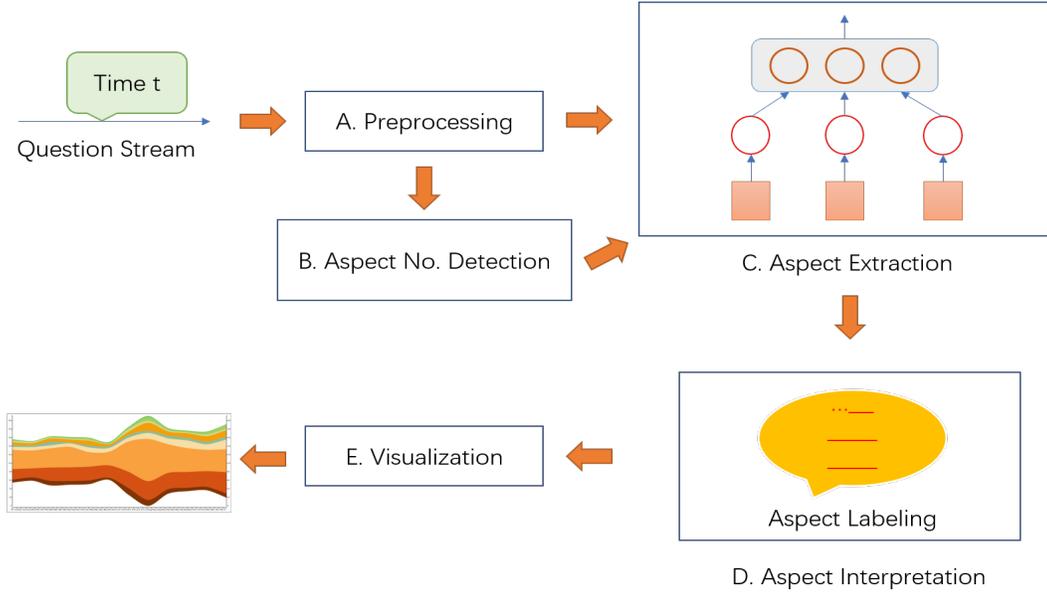


Figure 7: Framework of our model

In this section, we first outline the overall framework of our model in Figure 7. To deal with the raw data (the crawling part is not contained in the Figure 7), the first stage (Part A in Figure 7) 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 sent to the Aspect Number Detection (Part B in Figure 7) first to get the most appropriate number for aspect. After that, the preprocessed data are passed to the Attention-based Aspect Extraction model 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 7). Then, to interpret the topics (Part D in Figure 7), we will manually give each aspect a brief distribution of a single word or phrase. Since this part is really tedious and time consuming, and it can be done by some deep learning approaches like sentence reconstruction, we plan to implement an automatic method in the next semester. Finally, the model visualize the result by word cloud and issue river (Part E in Figure 7).

4.1. Data Crawling

Though we already have over 5,500 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 9,000 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 StackOverflow⁴, 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 8, 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.

The approach we used to catch the whole information of questions was

⁴<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 8: Example of crawling data from home page

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. In this semester, we are not going to use “votes”, “comments” or other information as supplement inputs to our model, so the input file only contains the title and body of each question. And in Figure 9, you can find the whole question “Reduce image dimensions in python”, with the whole “body”, compared to the last question in Figure 8.

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 difficulties in all NLP problems, since deep learning questions always contain codes and terminologies, it has some specific challenges when preprocessing.

```

{"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 9: Example of crawling data after visiting the url

What’s more, unlike sentiment analysis or irony detection 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 (regard as non-informative words in this research). And because the raw data we downloaded and crawled contain html tags (see in Figure 8 and Figure 9), it is difficult for us to determine whether to eliminate these tags (some tags like `...</code>, ... 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 package⁵ for lemmatization. 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:

Predefined Stop Words: cool, fine, hello, alright, poor, plz, pls, thank,old, new, asap, bit,someone, love, like, annoying, beautiful, dear.

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 in the crawled

⁵<https://www.nltk.org/>

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:

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

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 structures, which leads to a “fake” loss, which we will give a detailed analysis in the Section 5.2.1.

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)

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, etc⁶. And Word2Vec is one of the most popular technique to learn word embeddings using shallow neural network. Figure 10 is an illustration of word2vec.

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

⁶<https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa>

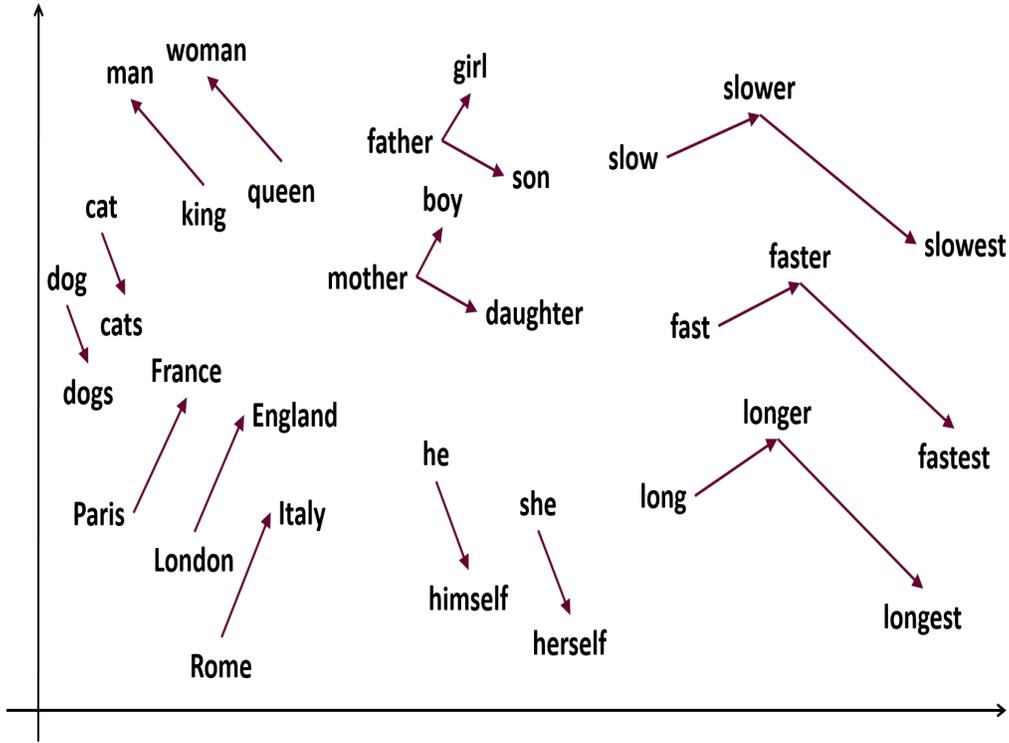


Figure 10: Illustration of word2vec

word embeddings are designed to map words that often co-occur in a context to points that are close by in the embedding space^[8] (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.

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

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^[10]. 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 11.

As you can see in Figure 11, it is not hard to find the tcs decrease greatly when aspect number reaches 4, 7, and 49. For the two numbers less than ten, we think it is too few to summarize all the aspects in such a large data set. Actually, if you calculate the proportion of new aspects 4 or 7, it is still quite large because the overall tcs summation is not big. And from aspect 7 to aspect 48, the tcs are decreasing in a very slow speed. We think since the overall summation becomes bigger and bigger, the proportion of each new aspects becomes further smaller, in other words, the contribution becomes less and less significant. We think it is also reasonable to choose a number around 30. However, we finally find the tcs of aspect 49 is sharply decreasing, so 48 will be the best aspect number to choose.

Since compared to the tcs of aspect 1 to 7, the tcs of other aspects seems small, these aspects may tend to be less significant aspects. In the Section 5.3, we manually assign some labels to each aspect. According to Figure 11, the aspects later should theoretically have fewer frequencies and are more likely to be “Noisy” aspects. This will explained in details later.

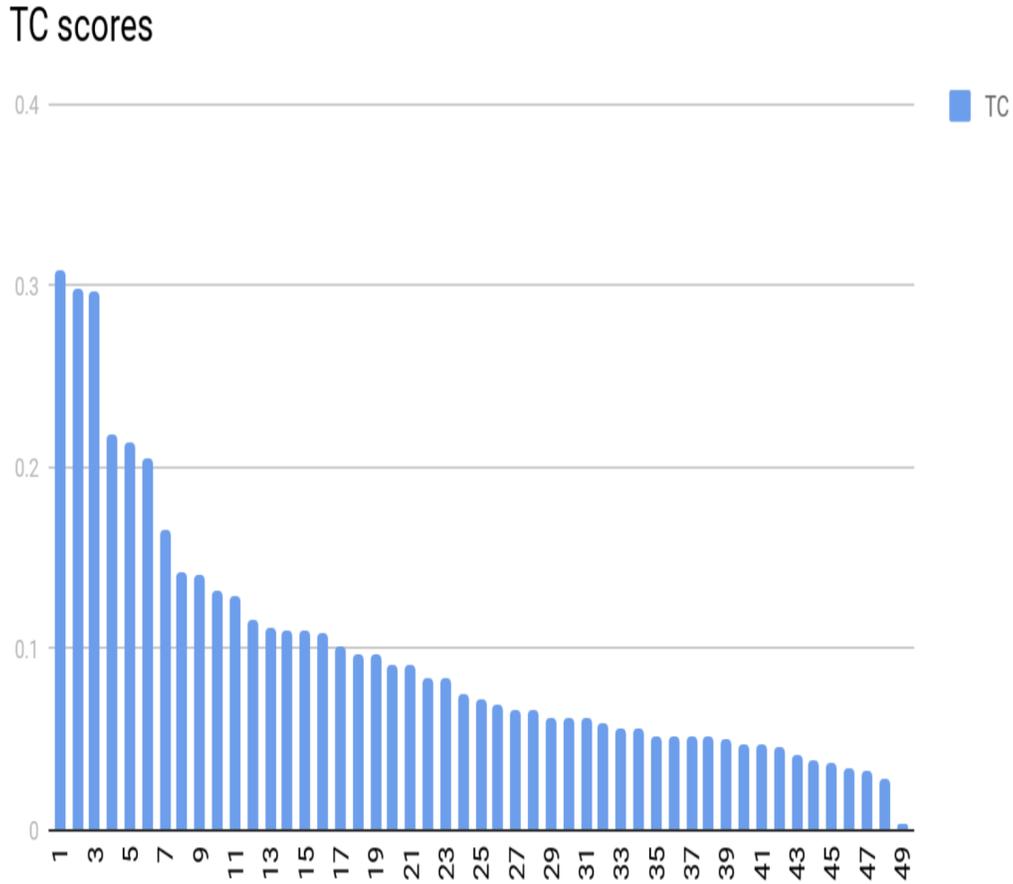


Figure 11: Topic correlation distribution

Every time you update the training data set to input into the Attention-based Aspect Extraction model (showed in Figure 7), you should always perform Aspect Number Detection first.

4.4. Model Description

We describe the Attention-based Aspect Extraction (ABAE) model in this section. The final goal of this 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 model is shown in Figure 12 in page 39.

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 12, 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 embedding \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 12) into their reconstructions (which is \mathbf{r}_s in Figure 12) 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 12.

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} , $i = 1, \dots, n$ corresponding to the word indexes in the question in the first step.

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

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 global context of the question:

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

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

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

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 12, 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 be obtained by dimensionality reduction of \mathbf{z}_s from d dimensions to K dimensions, and apply a softmax non-linear activation function which yields normalized non-negative weights:

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

where \mathbf{W} is the weighted matrix parameter, and \mathbf{b} is the bias vector. 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:

$$\mathbf{r}_s = \mathbf{T}^T \cdot \mathbf{p}_t \quad (6)$$

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^{[19][20][21]} (Weston et al., 2011; Socher et al., 2014; Iyyer et al., 2016). For every input question, we randomly select 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:

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

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:

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

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 :

$$L(\theta) = J(\theta) + \lambda U(\theta) \quad (9)$$

where λ is the weight parameter of the regularization term.

4.5. Visualization

In this part, we introduce how we visualize our results. Since the Attention-based Aspect Extraction 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 wordart⁷ to create word cloud for convenience and

⁷<https://wordart.com/>

it is more vivid and user-friendly. To achieve a totally-automated framework, this step should be done in python in the future.

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”⁸. In 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.

⁸<https://d3js.org/>

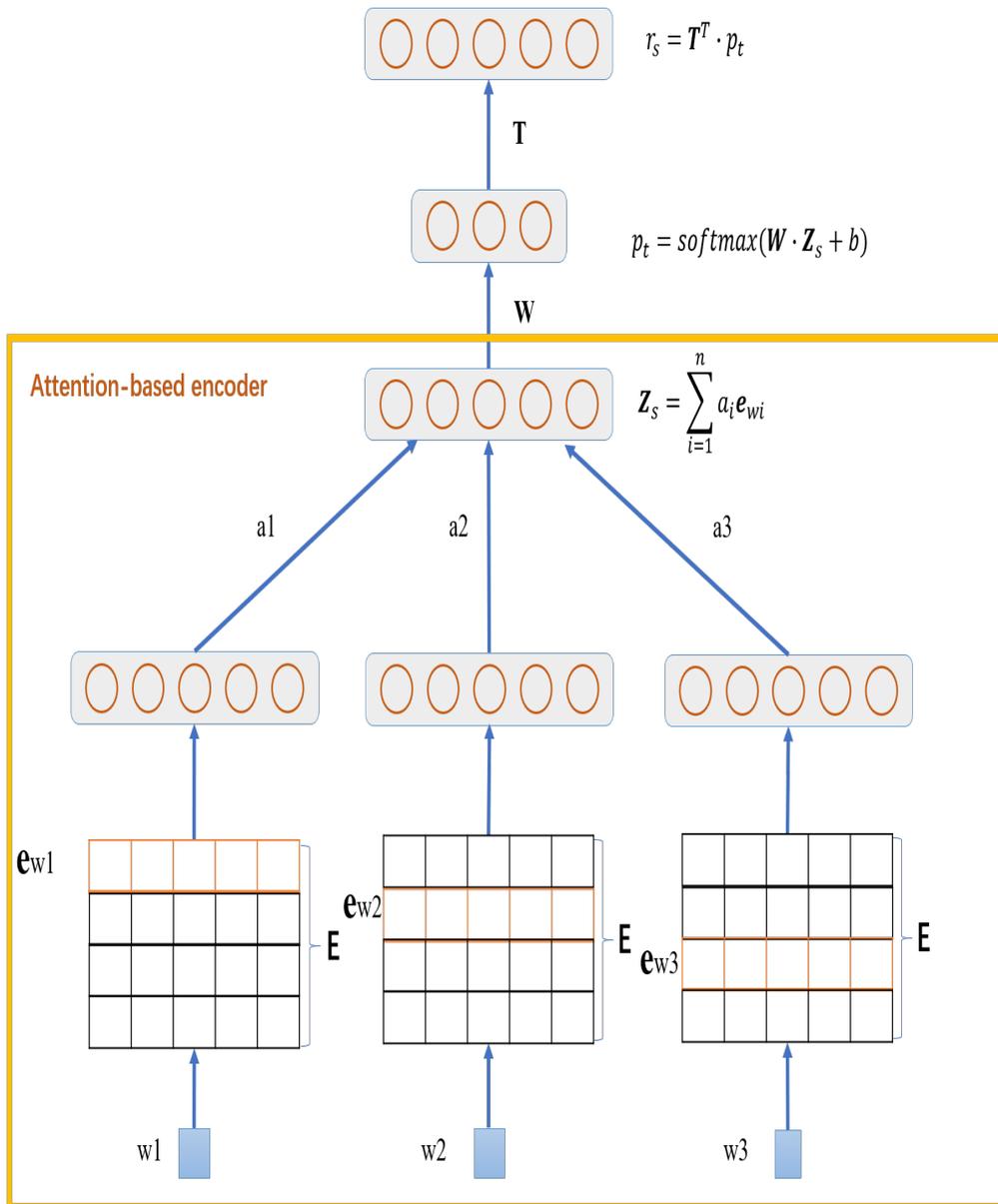


Figure 12: Illustration of ABAE

5. EXPERIMENTATION

This part we evaluate the performance of the Attention-based Aspect Extraction model and the visualization of 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 13. 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

```

<row Id="3629" PostTypeId="1" CreationDate="2017-07-11T19:13:36.097" Score="2" ViewCount="
48" Body="&lt;p&gt;I'm not quite sure how I should go about creating a multi-label image
KNN classifier using python as a lot of the literature I have read does not explicitly
explain this methodology. Specifically, I am not sure how I would be able to potentially
yield multiple labels per image using the KNN classifier architecture. Any insight would
be greatly appreciated! (new to coding by the way)&lt;/p&gt;&#xA;" OwnerUserId="8394"
LastActivityDate="2017-07-18T18:58:50.047" Title="Multi-Label Image Classification using
KNN" Tags="&lt;classification&gt;" AnswerCount="1" CommentCount="0" />
<row Id="3630" PostTypeId="1" CreationDate="2017-07-12T13:46:46.373" Score="2" ViewCount="
64" Body="&lt;p&gt;I am currently in the pre-process of starting an image classification
and extraction project which needs to output multiple softmax and absolute values from a
single image like such:&lt;/p&gt;&#xA;&#xA;&lt;pre&gt;&lt;code&gt;{&#xA; time: &quot;20:20
&quot;, &#xA; teams: [&#xA; {&#xA; red: { goals: 2},&#xA; blue: { goals: 1},&#xA
; },&#xA; {&#xA; scored_by : [&#xA; john: 80%, &#xA; kyle: 51%, &#xA;
darren: 20%&#xA; }&#xA; ]&#xA; }&#xA;}&#xA;&lt;/code&gt;&lt;/pre&gt;&#xA;&#xA;
&lt;p&gt;I can create multiple models which are responsible for different task such as
reading the time from the image as well as the score and eventually combine both. I would
however like to make sure I maximize on efficiency to make sure the process is as fast as
possible.&lt;/p&gt;&#xA;&#xA;&lt;p&gt;Any pointers in the right direction would be
greatly appreciated.&lt;/p&gt;&#xA;&#xA;&lt;p&gt;With kind regards,&#xA;Dennis&lt;/p&gt;&
#xA;" OwnerUserId="8405" LastEditorUserId="8405" LastEditDate="2017-07-12T14:25:41.753"
LastActivityDate="2017-12-05T17:34:04.613" Title="Extracting multiple softmax values from
image" Tags="&lt;neural-networks&gt;&lt;machine-learning&gt;&lt;classification&gt;"
AnswerCount="1" CommentCount="3" />
<row Id="3631" PostTypeId="1" CreationDate="2017-07-12T16:49:42.553" Score="-1" ViewCount=
"255" Body="&lt;p&gt;I have created 22 different Convolutional neural networks that all
test for the presence of unique objects in an image (each one of the classifiers is
unique). &lt;/p&gt;&#xA;&#xA;&lt;p&gt;Each sample in the test set has the output of a
22-long vector that looks something like this [0, 1, 1, 0, 0, 1, ..., 1], the binary
nature of the vector representing the presence/absence of specific objects. &lt;/p&gt;&#xA
&#xA;&lt;p&gt;I have implemented this already in keras and reach around 97% accuracy avg
for the 22 models. Is there any specific ensemble methods that can allow me to combine
all 22 classifiers?&lt;/p&gt;&#xA;" OwnerUserId="7773" LastActivityDate="
2017-07-13T14:17:32.933" Title="Ensemble Learning using Convolutional Neural Networks"
Tags="&lt;neural-networks&gt;&lt;machine-learning&gt;&lt;convolutional-neural-networks&gt;
&lt;classification&gt;&lt;keras&gt;" AnswerCount="3" CommentCount="0" />

```

Figure 13: An example of training data

5.2. Training

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 14, 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 `< /li >`) with “ol” (`< ol >` or `< /ol >`), “alt” and “png” or “jpg” with “img” (`< img scr = ...jpg/png alt = ...`) in Aspect 8 and 9 respectively. What’s more, the 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

```
Aspect 1:
[u'strong', u'h1', u'noreferrer', u'oxforddictionaries', u'ab', u'h2', ...]
...
Aspect 4:
[u'hr', u'answer', u'ask', u'emotion', u'think', u'question', ...]
Aspect 5:
[u'code', u'feature', u'camp', u'variable', u'vector', u'gt', ...]
Aspect 7:
[u'pre', u'en', u'wikipedia', u'rel', u'org', u'convolutional_neural_network', ...]
Aspect 8:
[u'stack', u'img', u'jpg', u'png', u'alt', u'imgur', ...]
Aspect 9:
[u'ol', u'li', u'general', u'human', u'intelligent', u'agi', ...]
Aspect 10:
[u'ul', u'exchange', u'post', u'overflow', u'stackexchange', u'datascience', ...]
...
Aspect 12:
[u'p', u'train', u'use', u'used', u'using', u'network', ...]
```

Figure 14: Aspect terms with tags and noisy words

topics of deep learning. And since html structures are formal and regular, it leads to a fake smaller 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.

5.3. 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.

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
Aspect 5	Deepstack, investigate	Deep learning model
Aspect 6	Collide, commented, weaker, reception	Noise
Aspect 7	Neural, caffe2, stimulate	Deep learning platform
Aspect 8	Move, state, action, player, game	Decision making algorithm
Aspect 9	Learning, urltry, un, divisor, spy	Noise

Order No.	Top words	Label
Aspect 10	Move, state, player, surroundings, current	Decision making algorithm
Aspect 11	dml(deep metric learning), differentiation, curing	Deep learning model
Aspect 12	Cocke(Cocke-Kasami-Younger algorithm), parsing	NLP
Aspect 13	Data, training, set, test ,model, learning, recognition, algorithm	dataset
Aspect 14	Network, canonical, blaise(name)	Deep learning papers
Aspect 15	Showcase, crashing, ongoing	Errors
Aspect 16	Game, play, chess, move, agent, player	Decision making algorithm
Aspect 17	Artificial, irrigation, 25th, num , <i>highlight</i>	Noise
Aspect 18	Eigenvector, pca	Matrix
Aspect 19	Pipelined, cepheus	Learning strategy
Aspect 20	Thrown, wildly, disguise, grained, motivator, antivirus, ipad	Noise
Aspect 21	Car, driving, crash	Self-Driving Cars
Aspect 22	Oppose, thirty, phylogenetic, dismissed	Noise

Order No.	Top words	Label
Aspect 23	Neural, artificial, deep, nutrition, convolutional	Deep learning model
Aspect 24	Melfrequency(MFCCs), recalibrate, electric	Voice recognition
Aspect 25	Answer, question, hyperbolic, stereotype, think	Noise
Aspect 26	Angora, men, pilot, sapiens, sluggo	Image Identification
Aspect 27	Learning, algorithm, procedural, reinforcement	Learning strategy
Aspect 28	Fatal, cybernetics, processing	Learning strategy
Aspect 29	Declassify, architecture, elected, withing, mutate	Noise
Aspect 30	Coalition, disclaimer, lay, thrill, radiation	Noise
Aspect 31	Accumulate, paraphrased, synthesizer	NLP
Aspect 32	ffnns(Feed forward neural networks), network	Learning strategy
Aspect 33	Machine, mentality, supervision, reinforcement	Learning strategy
Aspect 34	Politely, widow, schwefel, singly, dollar	Noise

Order No.	Top words	Label
Aspect 35	nsfw(Not Safe For Work, sexual pic), classification	Image Identification
Aspect 36	<num>, pursuing, eigenvector, 1×1	Matrix
Aspect 37	Potentiality, chief, vacuous, manipulates, appointment	Noise
Aspect 38	Focusing, dehydration, driverless, tantamount	Noise
Aspect 39	Language, natural, formalize	NLP
Aspect 40	Co2, suchs, outstanding, preparing, deer, dexterity	Noise
Aspect 41	Gulp, sheet, spacy, powerpoint	Tools
Aspect 42	<num>, heuristically, adopted, ridiculousness	Noise
Aspect 43	Layer, kaggle, neuron, hidden	Deep learning platform
Aspect 44	Flu, south, sexual, elasticity, noob	Noise
Aspect 45	Instrumental, urlsup, chapel, holland, devi	Noise
Aspect 46	Host, bh, override, toronto, cure, reconciling	Noise
Aspect 47	Reformulating, unfiltered, tesseract, spectral	Image Identification

5.4. *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, so we can not do the evaluations like the formal researches.

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 15. 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.

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 16. 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".

```

<row Id="5487" PostTypeId="2" ParentId="5336" CreationDate="2018-03-02T08:29:16.080" Score="1" Body="<p>Yes, you can train a NN to detect only one type of object like a table. However, you probably will not want to train such a NN from scratch by showing some examples of tables and non-tables. You will need to use <strong>transfer learning</strong> on a model already trained on several image classes and teach it to also recognize your new class. This transfer learning requires a smaller set of desired images. You may need to give it some negative examples also. You should explore transfer learning with <strong>mobilenet</strong>, <strong>inception</strong>, and other <strong>pre-trained Tensorflow models</strong> if you are willing to use <strong>Python</strong> and Tensorflow</p>&#xA;" OwnerUserId="10287" LastActivityDate="2018-03-02T08:29:16.080" CommentCount="0" />
...
<row Id="5570" PostTypeId="1" CreationDate="2018-03-08T06:10:00.340" Score="1" ViewCount="29" Body="<p>In two player games, the exact value of the evaluation function doesn't matter as long as it's bigger for better positions. However, for learning, it's customary when it does change when the best move gets made. This way, the learning can minimize the difference between the directly computed value <code>f(0, p)</code> of a position <code>p</code> and the value obtained from <code>n</code> step minimax <code>f(n, p)</code>. </p>&#xA;&#xA;<p>What I'm missing here is a way to direct the evaluation function to actually winning. For example, a <em>perfect</em> evaluation function for a won position in chess would always return <code>+1</code> without any hint how to progress towards checkmate. In a chess variant without the fifty-move limit, it could play useless turns forever.</p>&#xA;&#xA;<p>I guess, this is a rather theoretical problem as we won't ever have such a good function, but I wonder <em>if there's a way to avoid it</em>?</p>&#xA;" OwnerUserId="12053" LastActivityDate="2018-03-08T07:53:38.680" Title="Game AI evaluation function and making progress towards winning" Tags="<strong>reinforcement-learning</strong>&lt;strong>gaming</strong>" AnswerCount="1" CommentCount="0" />

```

Figure 15: The “Id=5478” post is predicted to be “Decision making algorithm”, while the “Id=5570” post is predicted to be “Deep learning model”.

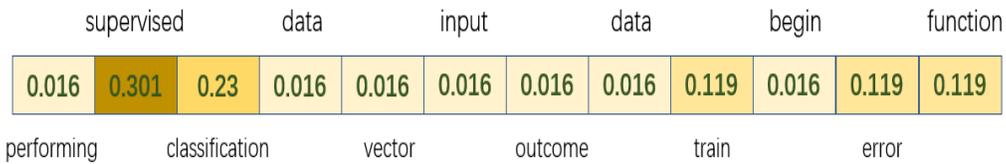


Figure 16: The attention mechanism focus more on “supervised” and “classification” since it is predicted to be “learning strategy”.

6. VISUALIZATION

This section introduces how we visualize our result.

6.1. *Word Cloud*

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

Since the word cloud part is not well developed yet, this semester we will only show a naive version of word cloud, which is shown in Figure 17 below. 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

the number of questions predicted to be certain label in each month. The labels from top to bottom in Figure 19 are in this order: NLP, Image identification, Tools, Deep learning platform, Learning strategy, Deep learning papers, Decision making algorithm, Dataset, Deep learning model.

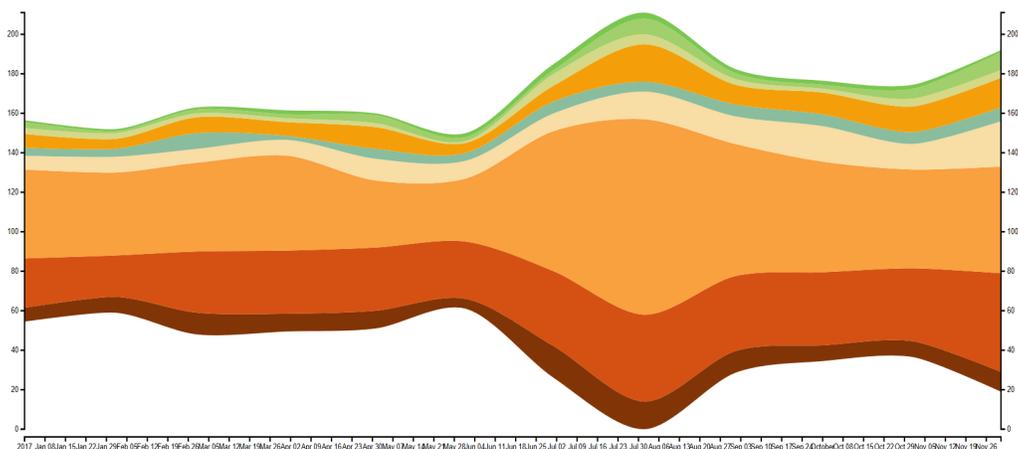


Figure 19: Issue river of 2017

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 20.

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.

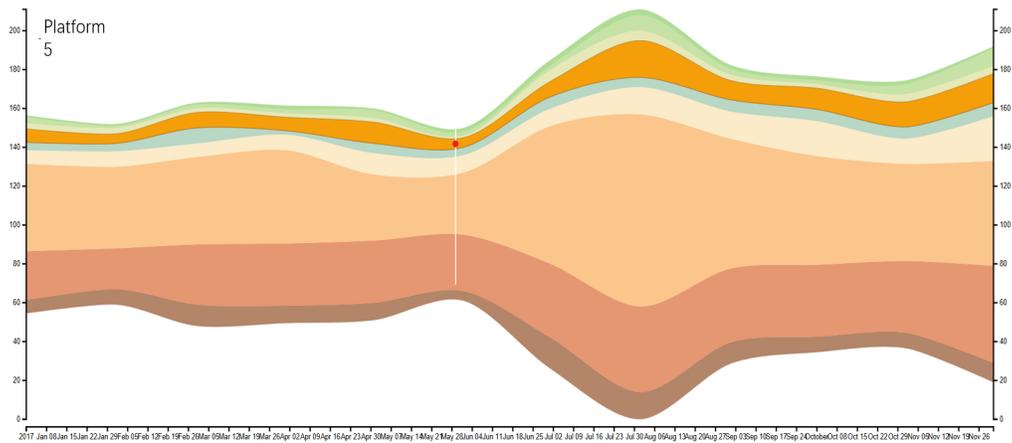


Figure 20: Mouse on red point

7. FUTURE WORK

These works can be done in the future:

- The preprocessing part only deals with single words. However, some words only make sense when they are phrases with other words (e.g. loss function). Phrase extraction can be done in the future. Meaningful phrases based on co-occurrence frequencies can be extracted and linked them with “_”.
- Some emotional words and abbreviations are still remained in the context after preprocessing, since the documents used nltk to extract stop words which may have little differences with questions context. These non-informative words should be collected and build a specific dictionary manually.
- Manually label some test data to further quantitatively evaluate the prediction accuracy of the model.
- Manually give description labels to each aspects consumes a lot of time, an aspect interpretation should be developed, which can automatically convert lists of aspect terms to meaningful phrases or sentences.
- This model cannot detect emerging issues since it do not compare the current input with former input. New model IDEA^[22] can be considered.

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] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research 3.Jan (2003): 993-1022.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.

- [7] 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.
- [8] 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.
- [9] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).
- [10] Gallagher, Ryan J., et al. "Anchored correlation explanation: topic modeling with minimal domain knowledge." arXiv preprint arXiv:1611.10277 (2016).
- [11] 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.
- [12] 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.
- [13] Li, Fangtao, et al. "Structure-aware review mining and summarization." Proceedings of the 23rd international conference on computational linguistics. Association for Computational Linguistics, 2010.

- [14] Wang, Wenyu, et al. "Recursive neural conditional random fields for aspect-based sentiment analysis." arXiv preprint arXiv:1603.06679 (2016).
- [15] 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.
- [16] Mnih, Volodymyr, Nicolas Heess, and Alex Graves. "Recurrent models of visual attention." Advances in neural information processing systems. 2014.
- [17] Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." arXiv preprint arXiv:1509.00685 (2015).
- [18] 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.
- [19] Weston, Jason, Samy Bengio, and Nicolas Usunier. "Wsabie: Scaling up to large vocabulary image annotation." IJCAI. Vol. 11. 2011.
- [20] 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.

- [21] 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.
- [22] Gao, Cuiyun, et al. ”Online app review analysis for identifying emerging issues.” 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). IEEE, 2018.