Capstone Final Report



20150757 CHOI JI HYUK

20154098 YEON SEUNG YOON

20170885 JUNG JAE HYUNG

INDEX

1. The purpose
2. The difference between ABB and Sadari
3. Structure and working process of ABB
4. Implementation of each module
5. Accuracy
6. Conclusion
7. Manual
8. The purpose

There are lots of information at the internet. And we search what we want to know about from there. Among the various formats of delivering information, blog is one of the most familiar and handy method for us to search because of goliath-like famous web portals like Naver. And Naver supports its ‘Naver blog’ very well.

Blog is a kind of a website driven by individual. It gets famous from early 2000’s and usually has been posted ordinary, everyday posts by individuals. What’s more, the visitors also access into blogs and intend to read the everyday posts.

It is no surprise that sellers want to use blog. They pretend like their blogs are ordinary one as usual and advertise their products to visitors. How come the visitors can distinguish it whether the post from blogs they read is advertising or not? Some marketing companies advertise their products on uncertain blogs with tiny image noticing that they are advertising but hard to notice it without focus. Moreover, not small portion of advertisers don’t mind about that, especially individual businessmen don’t.

Of course, Naver has tried to deliver right information to portal users with ‘C-rank’ and ‘DIA’ algorithms but these is only for the recommending orders on the searching page. However, users surf the inside of blogs too. And there is no consciousness that they only check the blogs on the searching pages. They literally flow all around blogs. Then, at some point, they read advertising blogs.

From this idea, our team supposed that there will be some characteristics of advertising posts of blogs and intended to implement this as a chrome extension.

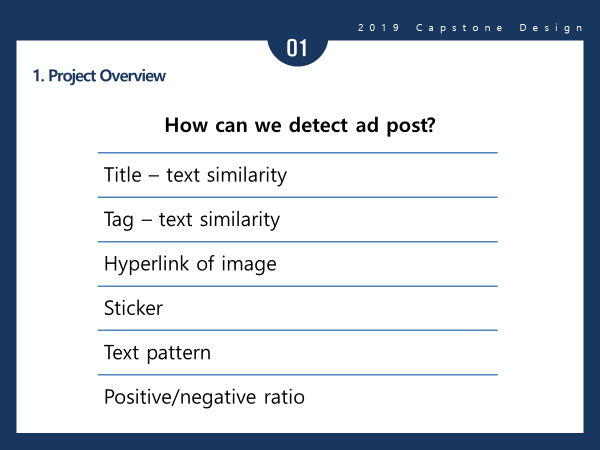
1. The difference between ABB and Sadari

With the notice from teaching assistance, we got to know there is similar program ‘Sadari’ with same purpose with our one. However, thanks to Sadari, we were able to aim the better feature to implement and thought about the new method when we search what feature to implement and how to implement. So, we searched papers and patent.

-Feature of Sadari



-Feature of ABB



As we read the feature of Sadari, it is split by 3 big portions. One is the structure of text content, and others are positive/negative and etcetera which is made of characteristics of text, image and sticker contents.

On the other hand, the feature of ABB, our program has 6 features. One is the consistency between title and content. And the one between #tag and content too. Also, we check the number and percentage of images and stickers which are usually used from advertising posts. What’s more, we use SVM method with tf - idf calculating. It is written as Text pattern on the ‘feature of ABB’ image above. At the last, positive/negative ratio is selected as feature too because most of papers and patent about advertising classification wrote the importance about checking positive/negative ratio.

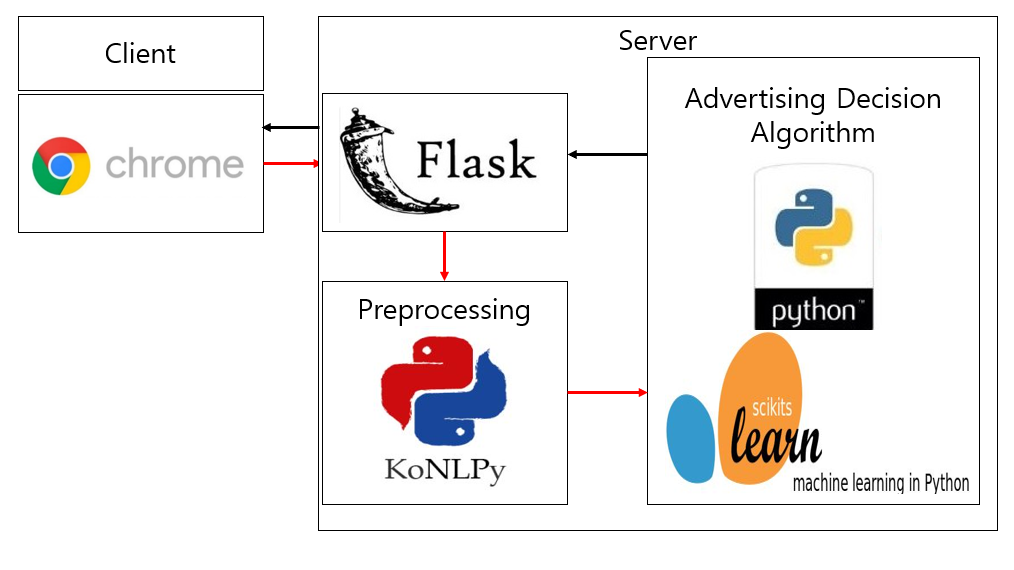
We found some subtle point from the feature of ‘Sadari’.

The first, Sadari’s feature mostly focus on the form of the text content, not on the meaning. And it really makes one wanders whether all the features show meaningful result. For example, as an algorithm member experimented about the structure of blogs, he found that the length of text content depends on individual’s characteristics, not on advertising or non-advertising.

The second, Sadari’s feature cared less about the difference between English text and Korean text. For instance, there is a feature about pronouns. This is what we found from a patent of American, and on the paper, there was an example about the pronoun feature. “Hey, how are you? Did you enjoy this Friday night? I got cold from Thursday and wasn’t able to hang out.” However, the point is that Koreans don’t use pronoun that much, they usually omit pronouns.

With this realization, we made effort on finding better feature and filtering out not appropriate feature. So, we checked about Korean papers more for those reason. Finally, the features chosen are the 6 on the image above.

1. Structure of ABB



The structure of ABB is made of two parts. Originally, we supposed to run the program on local computer, but api for algorithms demands extra computer setting which unprofessional users can’t do. So, we decided to use server during the implementation.

The whole flow of program proceeds from the chrome extension program. The first, the extension program checks the website if it is Naver blog or not. If it is the one, program parses text contents, image and sticker url and send it to server.

The next, the program in the server takes the data and starts calculating each feature with each module.

The third, after calculating, it sends the result through the extension program and user finally can get the notice from the program.

1. Implementation of each module

4-1 Server

The ‘payload.js’ file takes the current url of the Chrome at first. The next, if the current url contains the string ‘blog.naver.com’, the server changes the string belonged to ‘mainframe’ tag to chrome’s current url. If it isn’t the string ‘blog.naver.com’, other string from ‘screenframe’ tag of html tags to current url. If there is no ‘screenframe’ tag neither, halt payload.js, which runs on the Chrome background. Then again, if there is the string ‘blog.naver.com’ included in url, change a string from ‘mainframe’ tag to the Chrome’s current url. If not, deliver blogId and logNo value from current url to ‘popup.js’ file. Cycling this routine is effective to find certain url for parsing in 3 trials.

And the ‘popup.js’ file lets ‘payload.js’ work on the Chrome background. When it takes the url for parsing from Payload.js file, it sends the blog’s blogId and logNo value to Flask Server. After sent, each value is unveiled to user through popup.html file. If the server send an error message, it shows a connection failure to user. The content delivered to the server is written below.

1. Categnum: int type output, Category of the blog (not appeared on UI but for assisting the values below)
2. Emotion: float type output, positiveness/ negativeness of the blog
3. Title: float type output, consistency between the title and content / Tag: float type output, consistency between the tag and content
4. Link: list type output, url of certain images related to advertising
5. Sticker: list type output, url of certain stickers related to advertising
6. Svm: float type output, value decided by pattern from SVM method

4-2 UI

The colored circles on the left side are the signals about each feature. This idea is based on the traffic light. If the color gets red, it means the feature is highly doubtful as an advertisement. If it is yellow, a bit doubtful and if it is green, it won’t be an advertisement, from the decision of algorithm.

The one on the right side is the integrated decision from the algorithm. If it is red, the algorithm thinks that the content of blog is doubtful as an advertisement. And the yellow, green follows as the meaning from the explanation above. This helps for the user to decide whether to read the blog more or not intuitively.

4-3 algorithm: categorizing

From 4-3, the explanation will be about classifying algorithm feature whether the blog is advertising or not.

Categorizing is the first thing to do for the decision. The specific words selected by tf-idf algorithm which show features well are prepared in .txt files. With comparing the txt from blog and those .txt files, check the mostly matched category by the number of same words included. Luckily the words from each category are not same, so there is no need to worrying about misunderstanding. The output will be integer type like number 1 for restaurant category, and 2 for IT category and so on.

4-4 Emotion of the content

With the positive/negative word dictionaries prepared, check the number of words from blog which are same with the words on the dictionaries. After the work is done, the ratio ( the number of positive word/ the number of negative word) will be outputted by float type.

4-5 Consistency between title and text content

With using ‘konlpy api’ which split the text string from blog into ‘morpheme’, the program checks the morpheme from title. And then, it checks the number of morphemes which matches with the morpheme from the title. As a result, it outputs the ratio (the number of morphemes above / the whole number of morphemes from the blog text) with float type.

4-6 Consistency between tag and text content

The process flows as same as 4-5 above, with using the morpheme from tag instead of title.

4-7 Advertising image url

Advertising image url is usually used by online marketing companies which employ famous bloggers and ask them to upload advertising contents on their blogs. The point is, the companies upload specific images which shows they are advertising on the blog. And mostly, Korean words are included in the url.

So, there are two ways for classifying the advertising Image url. First is using ‘advertising image url’ dictionary. The famous companies’ image url are written on here. By comparing the url with the parsed image url, the program outputs the result by list type which includes the matched url with the dictionary and the parsed image url.

Another method is checking whether there is Korean morpheme included in the parsed image url by using ‘Konlpy api’. With this way, we can filter lots of small marketing companies and individual businessmen.

4-8 SVM

ABB has text patter recognizer SVM) as one of advertising determine modules. We use ‘scikit-learn’ for to make this recognizer. Recognizer consists of Token vectorizer, TF-IDF transformer and linear SVM. To discard useless words, chi square stats also used for recognizer.

Token vectorizer make text into vector. Each element of vector is count of word in that text and each row of vector is represent word in total words dictionary. This step make non-numeric text data into numeric form for SVM can use.

Chi square stats means how well each word represents each class. So we cut vectors over 50,000 rows into 100 rows by chi square.

TF-IDF is means the word is how especially important in the text. So we transform token vector into TF-IDF.

SVM use TF-IDF as a feature. SVM determine that the post either advertisement or not by TF-IDF values of words that represent either advertising or not. According to a previous study and our experiments, use kernel trick is not so good for text classification. Thus, our SVM is just linear.

1. Accuracy

When we parse data for experiments, restaurant category has been researched for a long time, so the accuracy differs from each category and the restaurant is the most accurate one with approximately 80% from 16000 parsed data. Also, there are IT, parenting product, fashion categories too, however the accuracy of those aren’t that high as 80% of restaurant category. It will be 65~70% accuracy.

1. Conclusion

In conclusion, for the users not to be swayed by the intention from advertising companies, our team implemented ABB chrome extension program. The program parses the contents of Naver blog, calculating the features blog advertisement from the server, and finally shows the decision by using 3 colors like traffic light whether the blog is highly doubtful as an advertisement or not with the result of 6 features’ colors.

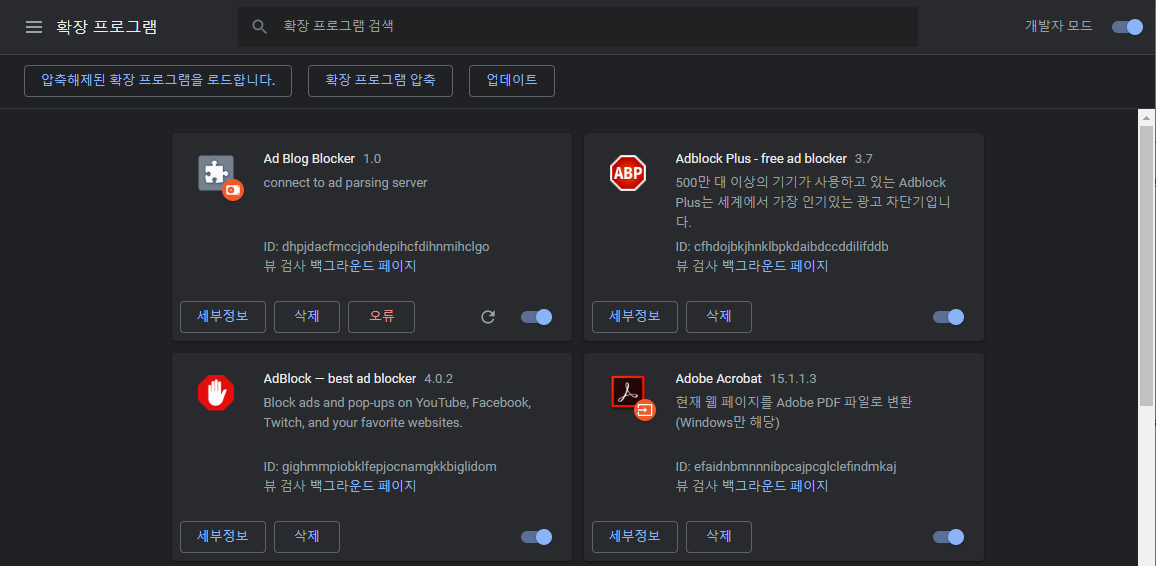
1. Manual

7-1 Github address

<https://github.com/yeomyeom/capston>

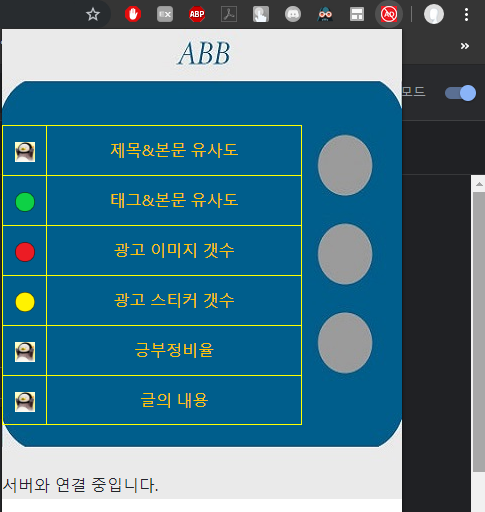
7-2 Way of using

1. Download the zip file from the link above.
2. Extract the file and open the Chrome.
3. Go to <chrome://extensions/> and turn the ‘developer mode’ on.
4. Load the ‘ChromeExtension’ folder from the downloaded file. (The folder where ‘manifest.json’ included)
5. After this process, ABB will be on the browser.



1. Turn on ‘Ad Blog Blocker’ and go inside naver blog.
2. Click the red ‘AD’ button and UI window will be opened.





1. The server isn’t turned on now, so it won’t work in real world.