Web Scraping and Cluster Analysis

Applicant name: Julieta Matevosyan

Web scraping:

I used Selenium's Chrome Driver to automatically access the company's technology blog by providing the page URL to the driver.

Then, using beautiful soup library, I conducted the following steps to get the article's text data:

- collect all the URLs of the buttons that are used to navigate to the next page in the blog
- collect all the URLs of the articles from every blog page URLs by inspecting the location of the URLs in the HTML tags in advance
- remove tags storing unneeded information for the analysis, such as page metadata, code scripts, etc.
- lowercase and strip the text, using regex, remove special characters, URLs that existed in the text, digits, strings with certain patterns that were common to every article (e.g., dates, blog category), English and Japanese punctuation marks, etc.

Finally, I stored a total of 54 article URLs and their corresponding preprocessed texts into a pandas dataframe.

Cluster analysis:

I decided to use topic modeling to uncover topics within each article that can highlight the article's content and help readers to find the required articles faster.

After analyzing the blog carefully, I noticed that the main distinguishing factor for the article's content is its title. However, for people who are less tech-savvy, it can be hard to quickly look up the article they are interested in which can potentially lead to frustration. Thus, I decided to use topic modeling to find the main keywords associated with each article and suggest using them as hashtags on the article's webpage. Additionally, by knowing what kind of topics a particular website visitor prefers, later on this information can be used to recommend a similar type of content to that specific user or a group of like-minded users.

To implement topic modeling, I conducted the following steps:

- made a function that can tokenize article's text using the 'sudachipy' module while also removing common Japanese language stopwords and words written in hiragana
- made a function that returns topics for each article using the 'gensim' library

In this case, I limited the number of topics to 1 since adding more topics did not result in more distinct words within the topics.

Discussion:

For each article, the words with highest probabilities within the topic can be used as hashtags to be added to the article's webpage and thus highlight its content. This can help the users or readers of the website to form an idea about the article's main content and navigate through articles that interest them faster.