# Automated Tagging of Stack Exchange Data Science Posts Using Natural Language Processing

A Capstone Project in Fulfillment of
Springboard's Data Science Career Track Program

By

Laura Elliott

And In Association with

Ajith Patnaik

Springboard Mentor

# Table of Contents

# 1. ABSTRACT / PROBLEM STATEMENT

For many question answering websites (Stack Overflow, StackExchange, Quora, Answerbag, Yahoo, WikiAnswers, etc.), a plethora of questions on many different subjects are submitted each day. Each question requires tagging with words or phrases to provide context for and to categorize that question. These tags are an important ranking factor in most of these website's search algorithms, so it is important for them to be as accurate as possible in order to ensure the best customer experience. The more quickly and accurately the tags can be assigned, the better the service can be about sending the right question to the right set of people for answering and the quicker the turnaround time for an answer to that question. The tags also provide the users with the ability to sort and filter through questions and answers on their topic of concern. Therefore automated tagging has become a necessity among these websites. The objective of this capstone is to build an algorithm to predict the tags using the text in the questions themselves.

The client could be a StackExchange (or other question answering entity) site developer who is looking for a faster and more accurate way to automatically tag questions. Based on the results of this analysis, the client may decide to deploy this new algorithm for auto-tagging questions in order to improve the efficiency of his team and the customers' experience.

## 1. Data Description

Questions and tags from the Data Science Stack Exchange database were used for this analysis. The Data Science Stack Exchange website is a question and answer site for Data Science professionals, Machine Learning specialists, and those interested in learning more about the field. It is one of 173 Stack Exchange Q&A communities (which includes the largest and most well-known site for programmers - Stack Overflow). The website link is:

https://datascience.stackexchange.com/

The dataset was extracted from the Data Science Stack Exchange database on July 22, 2020 via the Stack Exchange data dump site using this link:

https://archive.org/details/stackexchange

These data were archived on May 31, 2020, so contain information up to that point in time.

The data is formatted as a separate archive consisting of XML files zipped via 7-zip using bzip2 compression. Both the datascience.meta.stackexchange.com.7z and datascience.stackexchange.com.7z files were downloaded and extracted. The archive includes Badges, Comments, Posts, PostHistory, PostLinks, Tags, Users and Votes. The entire data schema is provided in Appendix 1.

For preliminary examination of the xml files, a free application called XML ValidatorBuddy was used to view the file contents and structure. This application was particularly helpful for

reviewing the larger xml files which could not be opened with other text readers due to size limitations.

The Tags.xml file in the meta archive was quite useful in evaluating the distribution and number of tags. The Posts.xml (79 MB) file contained within the datascience.stackexchange.com.7z contained all the necessary information for the analysis. Therefore, no joining of separate xml files was required.

The following features from the Posts.xml file were loaded for the analysis.

| *Features Name* | *Data Type* | *Description* |
|---|---|---|
| Id | int | unique post identifier |
| PostTypeId | tinyint | 1 = question; 2 = answer |
| CreationDate | datetime | Date of post |
| Score | int | sum of upvotes - downvotes |
| ViewCount | int | count of number of views |
| Body | nvarchar | question / answer body text |
| OwnerUserId | int | id of owner/user |
| LastActivityDate | datetime | date of last post activity |
| Title | nvarchar | brief description of question |
| Tags | nvarchar | comma-separated list of the tags associated with the question |
| AnswerCount | int | count of the number of answers |
| CommentCount | int | count of the number of comments |
| FavoriteCount | int | count of the number the post has been marked as a favorite |
| ClosedDate | datetime | date the post was closed |
| ContentLicense | nvarchar | description of content license |

*Table 1: Feature Description from the Posts.xml file*

The title feature was not included in the analysis.

# 2. DATA WRANGLING

The following sections outline the steps involved in the data wrangling process.

## 2. Conversion of XML file to Pandas DataFrame

The initial process of converting the Posts.xml file into a pandas dataframe was conducted in the "CP2_01_XML to DF.ipynb" Jupyter Notebook. The following pseudocode outlines the process:

1. Load and Parse the Posts.xml file using the xml.dom.minidom package.
2. Test whether the parsing worked by getting a sample of items from the file.
3. Create a header row from the items in each row of the xml; `df_cols = ['Id','PostTypeId','CreationDate','Score','ViewCount','Body','OwnerUserId','LastActivityDate','Title','Tags','AnswerCount','CommentCount','FavoriteCount','ClosedDate','ContentLicense']`
4. Create an empty list to append data to.
5. Write a for loop to extract all items from each row of the xml making sure to write an N/A for those fields that are empty.
6. Create a pandas dataframe with the extracted data and headers.
7. QC the dataframe and export as a pickle file.

Input is the original Posts.xml file from the Stack Exchange data dump. Output is a pandas dataframe called posts_df and associated pickle file called posts_df_09112020.pickle. The resulting posts_df dataframe consisted of 51395 rows and 15 columns (features listed in the table above).

For this notebook, only 3 libraries were used: xml.dom.minidom, pandas, and pickle.

## 3.

## 4. Data Conditioning, FIltering, and Cleaning

Additional conditioning and cleaning steps were performed in the "CP2_02_Load DF Clean and Analyze.ipynb" Jupyter notebook. The following pseudocode outlines the preliminary filtering and cleaning steps :

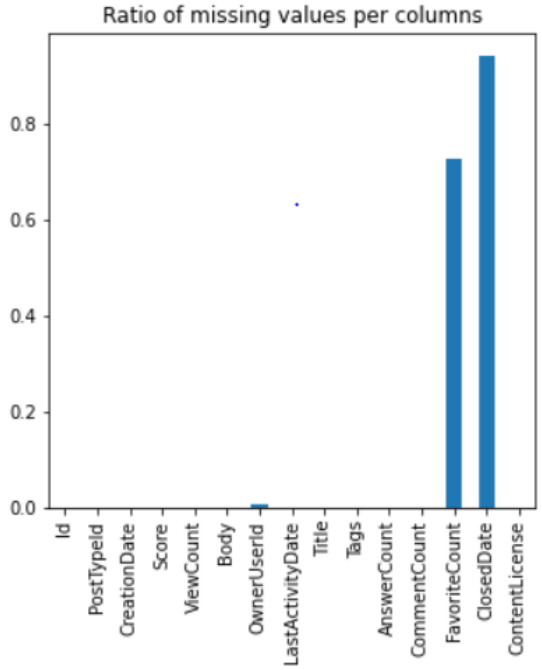
1. Load the posts_df_09112020.pickle file created in the first Jupyter Notebook.
2. Examine contents to assure everything transferred properly.
    a. Identified some cells filled with only white space. These were replaced with nans.
3. Filter down to just questions (PostTypeId = '1')
4. Convert objects to appropriate integer and date dtypes (text attributes Body, Title, and Tag were converted in later steps).
5. Check for missing values.
6. Check for and remove duplicates.
7. Preprocess tags - remove html tags and convert to string.
8. ...

Input is the pandas dataframe called posts_df and associated pickle file called posts_df_09112020.pickle from the first Jupyter Notebook.

For these steps, the following libraries were used: pandas, pickle and numpy.
For visualizations, the following libraries were used: matplotlib.

## 1. *Check of Missing Data*

The following bar chart and table describe the number of missing values in each column. The chart is shown as a ratio of missing to total values per column.




```
Int64Index: 24363 entries, 0 to 51393
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Id                24363 non-null  object
 1   PostTypeId        24363 non-null  object
 2   CreationDate      24363 non-null  object
 3   Score             24363 non-null  object
 4   ViewCount         24363 non-null  object
 5   Body              24363 non-null  object
 6   OwnerUserId       24248 non-null  object
 7   LastActivityDate  24363 non-null  object
 8   Title             24363 non-null  object
 9   Tags              24363 non-null  object
 10  AnswerCount       24363 non-null  object
 11  CommentCount      24363 non-null  object
 12  FavoriteCount     6710 non-null   object
 13  ClosedDate        1422 non-null   object
 14  ContentLicense    24363 non-null  object
dtypes: object(15)
memory usage: 3.0+ MB
```

*Fig 1: Ratio of missing to total values in each column (plotted with matplotlib) and results of dataframe.info query providing a count of non-null values.*

The results indicate that the critical columns have no missing values. Only the OwnerUserId, FavoriteCount and ClosedDate columns have missing values. The missing values in the latter 2 features are expected, since not all question posts will be identified as having a Favorite answer and not all will be Closed. The OwnerUserId has a few missing values, but was not used in the analysis.

## 2. *Check of Duplicates*

The dataframe was checked for duplicates of both the question (Body) and Title and 10 duplicate entries were found. Two of the duplications are shown below.

| Id | PostTypeId | CreationDate | Score | ViewCount | Body | OwnerUserId | LastActivityDate | Title | Tags | AnswerCount | CommentCount | FavoriteCount | ClosedDate | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 461 | 1 | 2014-06-18 22:10:58.497 | 12 | 1336 | <p>There's this side project I'm working on wh... | 986 | 2017-11-06 08:07:09.260 | Preference Matching Algorithm | <bigdata><text-mining><recommender-system> | 2 | 0 | 2 | NaT | |
| 477 | 1 | 2014-06-18 22:15:43.820 | 2 | 561 | <p>There's this side project I'm working on wh... | 986 | 2014-06-19 10:30:43.993 | Preference Matching Algorithm | <algorithms> | 1 | 3 | NaN | 2014-06-26 05:31:41.193 | |
| 2313 | 1 | 2014-10-20 06:38:16.490 | 5 | 1156 | <p>Where is the difference between one-class, ... | 4717 | 2015-03-03 14:21:51.033 | Machine Learning - Where is the difference bet... | <machine-learning><data-mining><classification... | 2 | 2 | 5 | NaT | |
| 2316 | 1 | 2014-10-19 21:03:35.390 | 1 | 47 | <p>Where is the difference between one-class, ... | 4717 | 2014-10-20 10:38:59.437 | Machine Learning - Where is the difference bet... | <machine-learning><classification><data-mining> | 0 | 1 | NaN | 2014-10-21 14:09:06.617 | |

*Fig 2: Partial results of check for duplication in Title and Body.*

The results illustrate that these duplicates were created by the same owner but have slightly different creation dates and only one of the grouped pairs has a closed date. Additionally Score, LastActivityDate, Tags, and ViewCount, AnswerCount, CommentCount, and FavoriteCount are different. For this reason, the second entry with lower counts and closed dates were removed. Removal of these duplicates reduced the total row count of the dataframe from 24363 to 24353 entries.

### *3. Preprocess Tags*

As noted in Figure 2 above, each of the tags in the Tags column are separated by html <> characters. These were removed and the text was converted to a space delimited string in preparation for further analysis.

## Preprocess Body (Questions)

Many cleaning steps were required to condition the question text ('Body' attribute) for analysis. These steps were performed in a third Jupyter notebook "CP2_03_Clean_Body_Text_Analyze.ipynb".

The following pseudocode outlines the steps taken:

1. Load the pickled pandas dataframe from 02 notebook
2. Examine the feature 'Body' (which are the questions) to identify cleaning tasks
3. …
4. ...
5. Cleaning tasks:
    a. Remove code snippets
    b. Remove html formatting
    c. Expand contractions
    d. Language detection to make sure everything is in English
    e. Remove special characters
    f. Simple lemmatization
    g. Named Entity Recognition
    h. Part Of Speech Tagging
    i. Convert to lowercase
    j. Remove task remove stop words
6. Export the dataframe with cleaned text and additional features for further analysis

Input is the pandas dataframe called questions_df and associated pickle file called questions_df_09252020.pickle from the second Jupyter Notebook.

For these steps, the following libraries were used: collections, string, textwrap, unicodedata, contractions, fasttext, spacy, nltk.

To identify cleaning tasks, a sampling of the questions were printed for examination. Figure 3 shows a few examples.

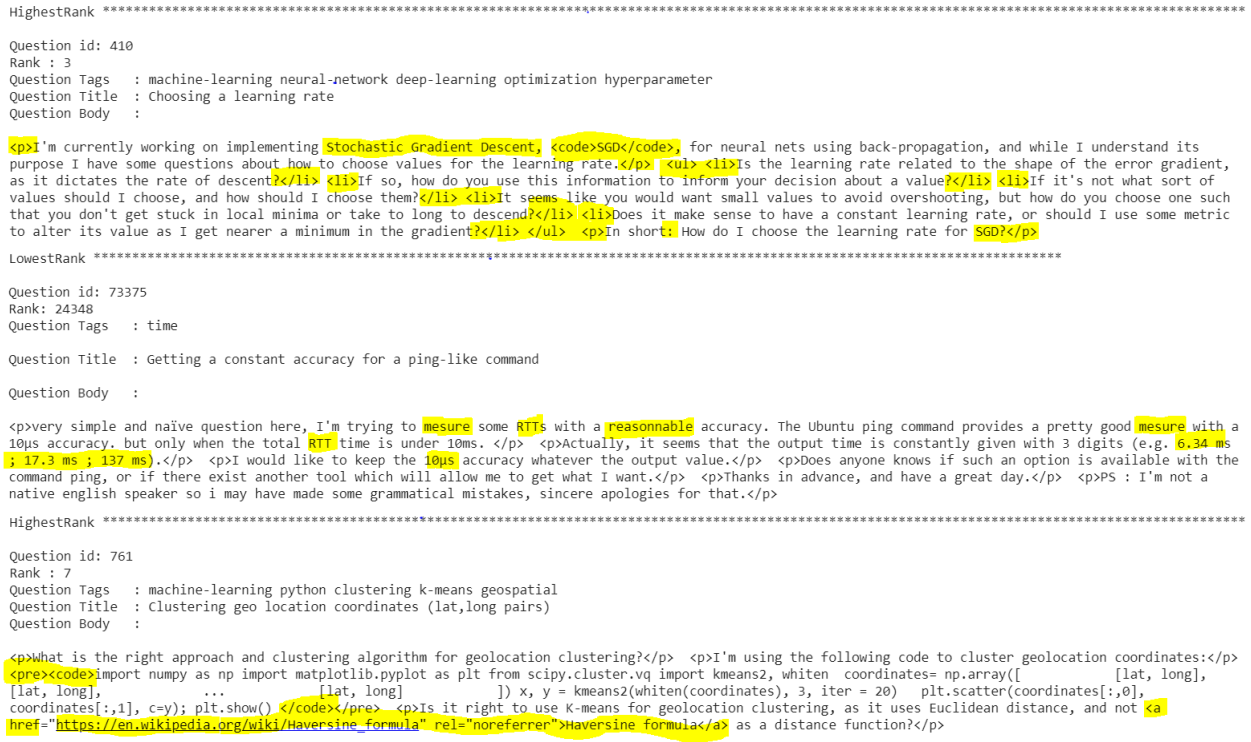



*Fig 3: Question examples showing the raw question "Body" text, with various text highlighted to show important items identified for cleaning. Text was formatted using 'textwrap' for increased readability.*

The first thing that stands out as an issue requiring cleaning is the html formatting, such as the   and <li> </li> pairs that denote paragraph and line formatting. In some cases, text is bolded for emphasis. There are also hyperlinks and code snippets that tend to clutter the overall question but are provided as examples or reference. Standard special character removal and contraction expansion is also required.

Note in the examples shown that oftentimes more than one question is asked within the Body or sometimes no questions is asked at all. Question length can be quite variable also. Therefore, prior to performing the cleaning steps, certain features were extracted from the Body (such as the number of code snippets, number of question marks, body text length, and number of bolded text items).

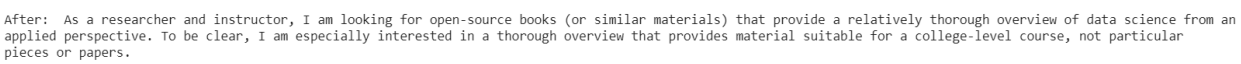
It appears that acronyms like LDA, PCA, SGD, RTT are important to recognize, which indicates that we should attempt a Named Entity Recognition and Part of Speech tagging process as a part of the text processing.

Some misspellings were also identified, but have been kept in place. Some examples are shown above. In one case, "but" was misspelled as "bot", the latter being an important word with a very different meaning in the data science world.

Beautiful Soup was then used to identify and remove the code snippets and then to remove the remaining html formats. Figure 4 shows an example of a question before and after this process.


*Fig 4: Example question before and after removal of code snippets and html formatting.*

The next step was to expand contractions, using the contractions module. Figure 5 shows a before and after example of this process.



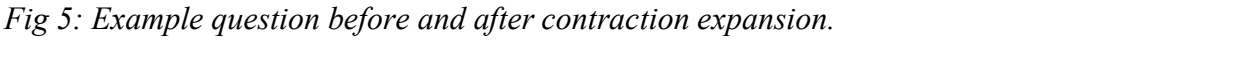




*Fig 5: Example question before and after contraction expansion.*

The next step was a check to see if all questions are in English, using Facebook's fasttext library and their prebuilt model (lid.176.bin). Only 3 questions returned as a different language (fr,ja,kn) but a review of these rows indicates that the processing of the questions done in previous steps reduced them to just one word or character, and were therefore producing erroneous language results. A preliminary pass of the English language detection routine prior to cleaning the code snippets identified some string examples within the text that were in an alternate language, but these were eliminated by first removing the code snippets. To summarize, all text from the Stack Exchange data dump is in English, although some examples provided within the question code snippets were in a different language. These examples were all removed as a part of the code snippet removal process.

The next step was to remove all special characters in the text. In order to identify what characters should be removed, further analysis was performed using the nltk dictionary 'punct_only'. There are 329 unique special characters in the Body text. Most of the special characters are punctuation and symbols and these were removed:

!"#$%&\'()*+,./:;-<=>?@[\\]^_`{|}~

However Greek characters such as α, β, θ, σ, ⅄,Ψ, and μ were maintained because they have special significance in the data science field.

NLTK's WordNetLemmatizer was then used to perform a simple lemmatization of the text. Sample results of the special character removal and lemmatizing process are shown in Figure 6 below.

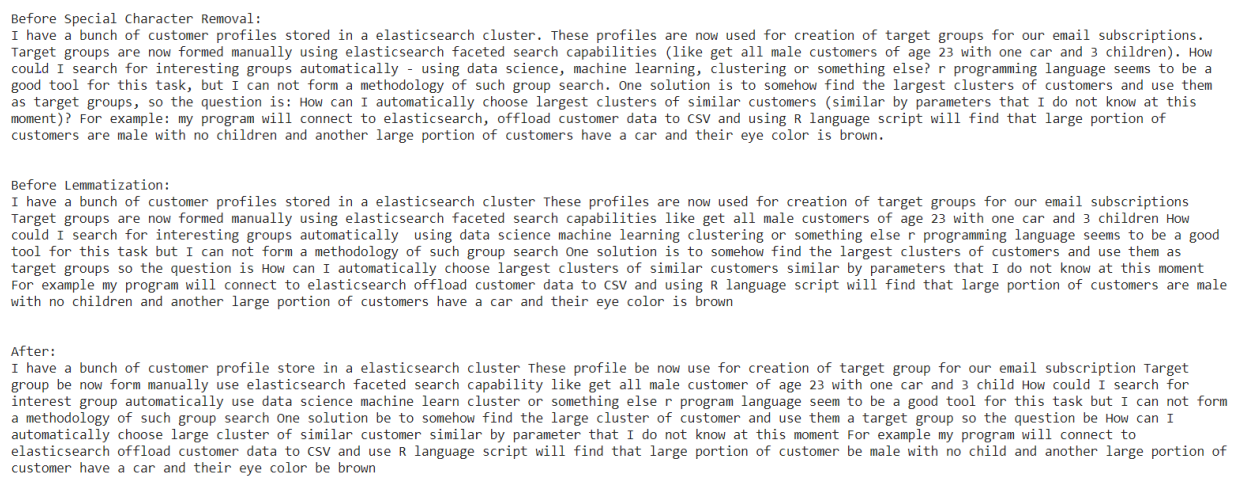



*Fig 6. Sample showing questions text before and after special character removal and lemmatizing.*

An attempt was made to recognize, extract, and label named entities using Spacy's Named Entity Recognition and corresponding corpus 'en_core_web_sm'. Running this process, even with this small dataset takes quite a bit of time, and while effective at recognizing and extracting the named entities, the labelling results were poor. Figure 7 below shows that terms like "Libsvm", "Liblinear", and "MapReduce" are identified erroneously as a Geopolitical Entity (GPE), an Event, and a Person, respectively. This is a result of the fact that the Spacy algorithm is trained on the general corpus but our dataset is quite technical. Further work to improve the labelling results was deemed beyond the scope of this capstone project.

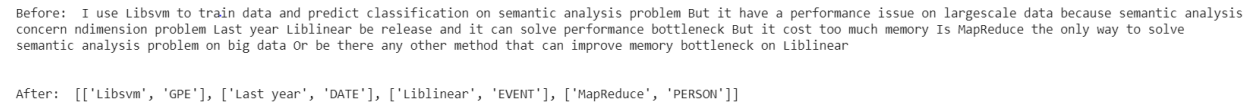



*Fig 7. Sample showing poor labelling results for technical data science terms during the NER process.*

Nouns were also recognized and extracted using NLTK's pos_tag module. The results will be examined and discussed further in the exploratory analysis section of this report.

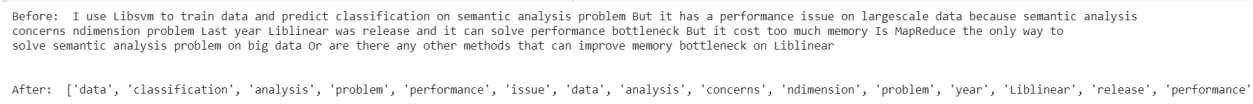



*Fig 8. Same sample as in Figure 7 showing results of POS tagging for nouns. Some, but not all, named entities are recognized also through this process.*

The next cleaning step for the question text was to convert all to lowercase. It was determined via experimentation that performing the lowercase step prior to NER and POS tasks produced less satisfying results. The final step of text cleaning was to remove stop words. The following words were removed from the stopwords list since they are meaningful for data scientists - 're',

'r', and 'q'. A before and after sample of the lowercase conversion and stop word removal is shown below.

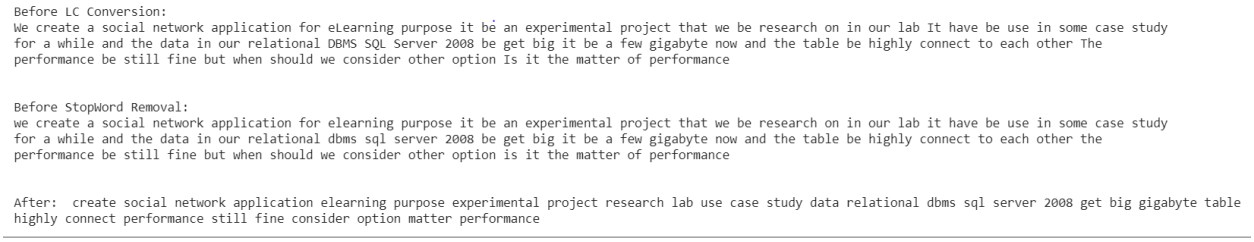



*Fig 9. Sample showing results of lowercase conversion and stop word removal.*

Final Output is a pandas dataframe called questions_df consisting of 24363 entries and 63 columns. The associated pickle file exported is named questions_df_ner_results_10282020.pickle.

# 3. EXPLORATORY DATA ANALYSIS

## Tag Distribution Analysis

The following pseudocode outlines the process used to analyze the tag distributions. These steps were performed in the last portion of the "CP2_02_Load DF Clean and Analyze.ipynb".

7. ...
8. Get a list of all the Tags and Frequency Counts
9. Exploratory analysis / visualization of tags related to their frequency
10. Exploratory analysis / visualization on other features such as ViewCount and Score
11. Create TopTagsList based both on frequency and ranking by ViewCount and Score
12. Write out the filtered data frame with new features

For these steps, the following libraries were used: collections, Beautiful Soup, nltk, and re. For visualizations the following libraries were used: matplotlib, seaborn, and wordcloud.

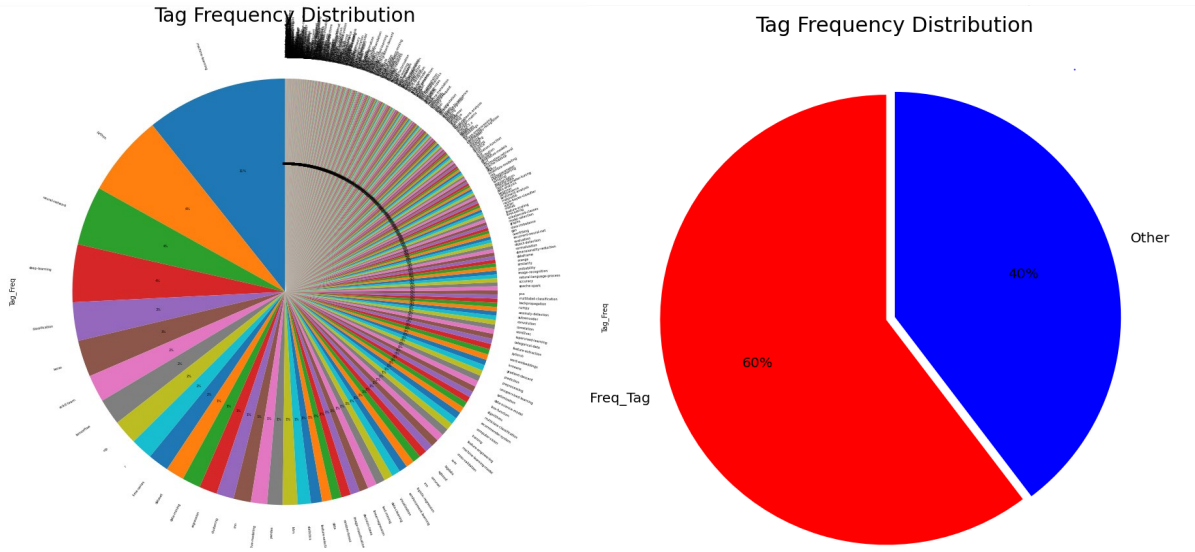



*Fig 10: Pie chart showing the frequency distribution of all of the tags. There are a total of 590 unique tags in the dataset, but the top 32 tags represent 60% of the data.*

As shown in Figure 10, of a total of 590 unique tags, the top 32 most frequent tags encompass 60% of the total data.

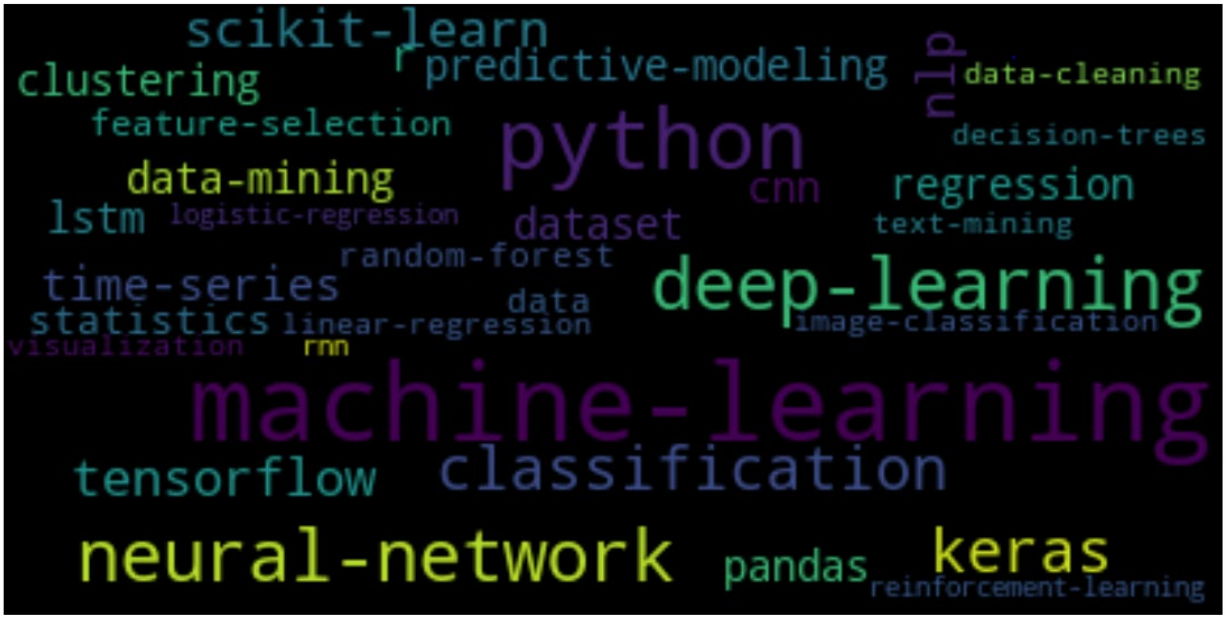



*Fig 11: Wordcloud of top 32 tags, showing the popularity of machine and deep learning topics.*

Figure 11 shows the top 32 tags as a wordcloud. You can see that machine-learning, deep-learning, and neural-networks and their associated software and libraries such as keras and tensorflow are the hottest topics in the forum.

Additional exploratory analysis was also performed on a newly created feature, the number of tags per question (TagCount), ViewCount, and Score, and the correlations between them.

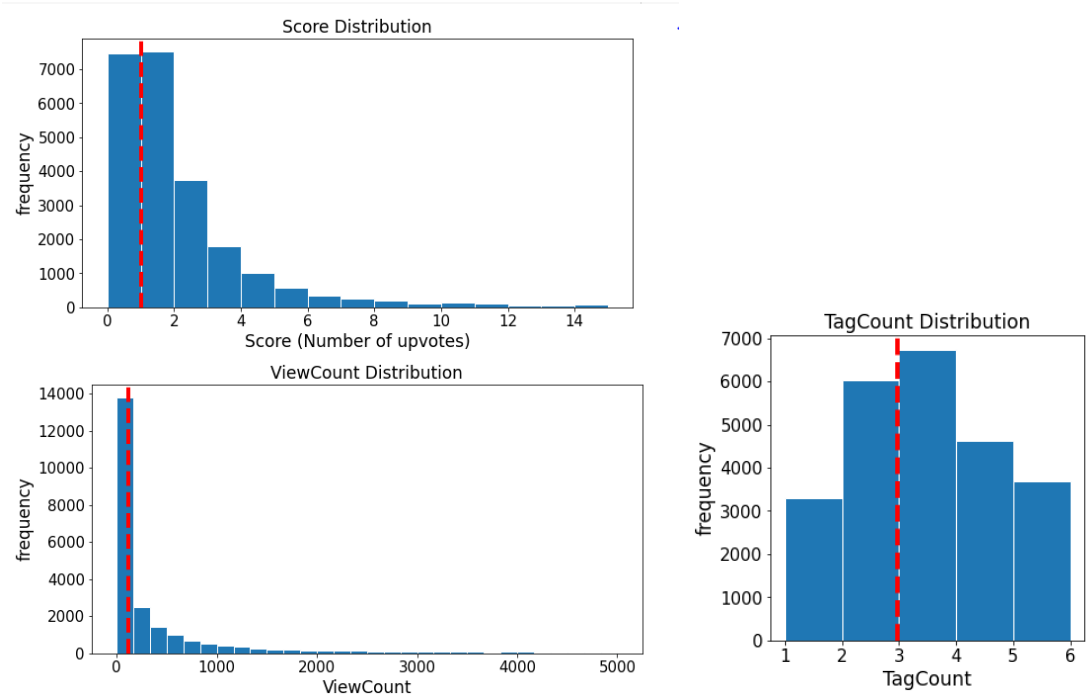



*Fig 12: Frequency Distribution of Score, ViewCounts, and TagCount.*

| | Score | ViewCount | TagCount |
|---|---|---|---|
| **count** | 24353.000000 | 24353.000000 | 24353.000000 |
| **mean** | 2.005872 | 1387.329651 | 2.974705 |
| **std** | 5.433220 | 6895.850492 | 1.257866 |
| **min** | -6.000000 | 2.000000 | 1.000000 |
| **25%** | 0.000000 | 36.000000 | 2.000000 |
| **50%** | 1.000000 | 114.000000 | 3.000000 |
| **75%** | 2.000000 | 578.000000 | 4.000000 |
| **max** | 214.000000 | 234355.000000 | 5.000000 |

*Table 2: Feature Statistics*

As would be expected for count features and shown in Figure 12 and Table 2, both Score and ViewCount have a skewed poisson distribution with mean values of 2 and 1387 respectively. TagCount has a normal distribution with min of 2, and maximum number of tags per question of 5.

Further exploratory analysis was performed to identify whether the questions with top tags have higher Scores, ViewCounts or TagCounts on average than the others. The questions were split into 2 groups, those with tags on the TopTagsList and those without, and means for these attributes were calculated for each group. The results are shown in Figure 13 below.

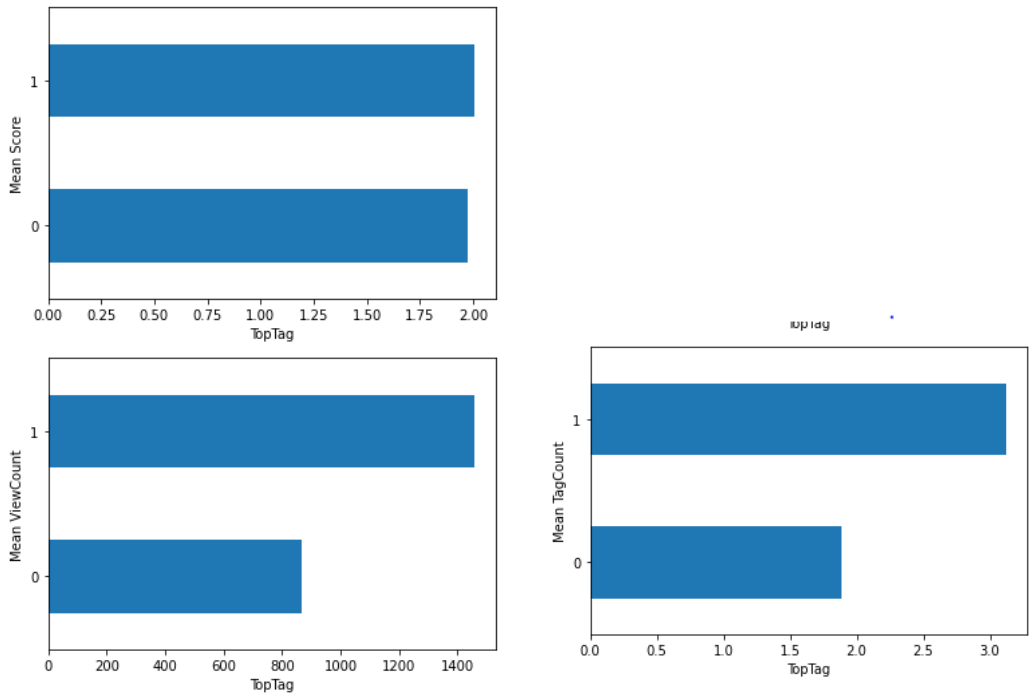



*Fig 13: Mean Score, ViewCount, and TagCount by TopTag Grouping*

The results indicate that the mean difference of the Score between the questions with Top Tags and those without is not statistically significant; however the ViewCounts and TagCounts are statistically significant being almost twice the value where questions are tagged with one or more of the top 32 most frequent tags.

Based on the strong correlations between these latter two attributes and the top tags, the tags were further high-graded, and re-ranked using not only the tag frequency but also the ViewCount and Score attributes. Rank order was determined by tag frequency + ViewCount + TagCount.

During this analysis, it became clear that use of all tags (up to 5 per question) produced more scatter and variation in the TopTag results. Using only the first tag for each question and excluding all the remaining tags (plus rank ordering) produced a list of 33 TopTags that comprised 88% of the data (as opposed to the initial 60%). It was therefore decided to use only the first (and most frequent) tag for each question for tag prediction.

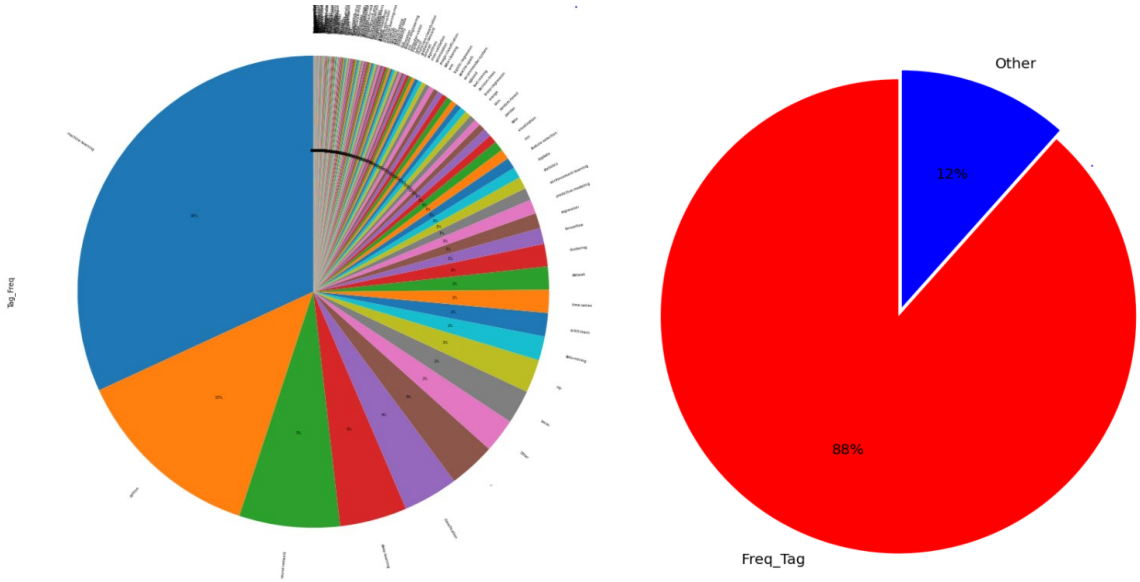



*Fig 14: Pie chart showing the frequency distribution of the ranked ordered first tags. There are a total of 349 unique first tags in the dataset, but the top 33 tags represent 88% of the data.*

As shown in Figure 14, of a total of 349 unique first tags, the top 33 most frequent tags now encompass 88% of the total data.

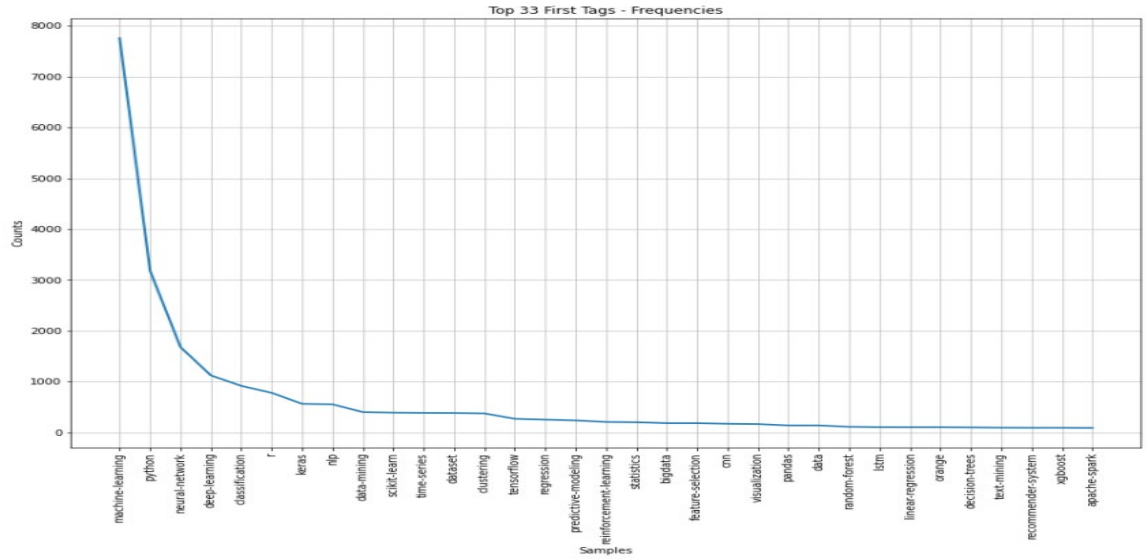



*Fig 15: Revised Frequency distribution of the ranked ordered first 33 tags.*

The revised 33 first top tags were used for the prediction. All other tags were renamed to 'OTHER'.

Output is a pandas dataframe called questions_df consisting of 24363 entries and 40 columns.
The associated pickle file exported is named questions_df_09252020.pickle.

## Question Text "Body" Analysis

The following pseudocode outlines the process used to analyze the 'Body' attribute - the question text. Some of these steps were performed in the preliminary portion of the "CP2_03_Clean_Body_Text_Analyze.ipynb"

2. …
3. Examine the feature 'Body' which are the questions to gain statistical insights
4. Get counts of certain special characters to add to features prior to cleaning
    a. Question Marks
    b. Bolded Text
    c. Number of Paragraphs
    d. Code Examples

Additional exploration and visualization steps were performed in the "CP2_04_CP2_04_Analyze_Features.ipynb" notebook.

1. Load the pickled pandas dataframe from 03 notebook
2. Examine contents to assure everything transferred properly
3. Analyze length of body (both characters and words) against other features and id any correlations
4. Analyze number of question marks against other features and id any correlations
5. Analyze text bolding against other features and id any correlations
6. Analyze number of paragraphs against other features and id any correlations
7. Analyze number of code examples and length of those examples and id any correlations
8. Analyze NER results against tags for correlation
9. Create Bag of Words and get word frequency count by tag
10. Export the dataframe with cleaned text and additional features for further analysis

Input is a pandas dataframe called questions_df consisting of 24363 entries and 63 columns.
The associated pickle file is named questions_df_ner_results_10282020.pickle.

For these steps, the following libraries were used: pandas, numpy, pickle, collections.
For visualizations the following libraries were used: matplotlib and seaborn.

Freq distribution of word and character counts

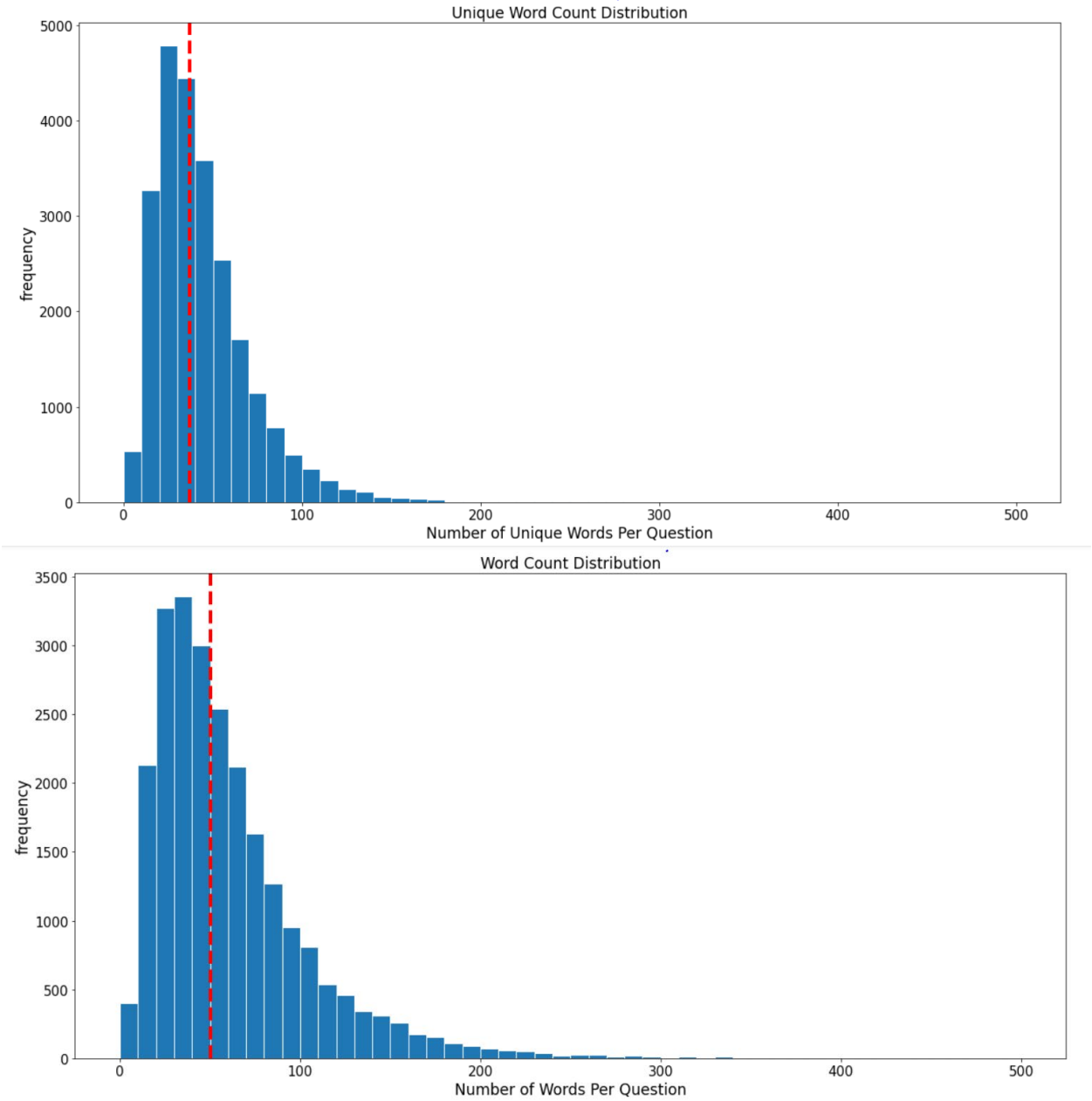



*Fig 16: Frequency distribution of the number of unique words and number of words per question in the dataset, showing the skewed results (poisson distribution) as would be expected by a count feature Red dashed line denotes the median, which is 44 and 62 respectively.*

No correlation between whether question tag is a top tag and question length.

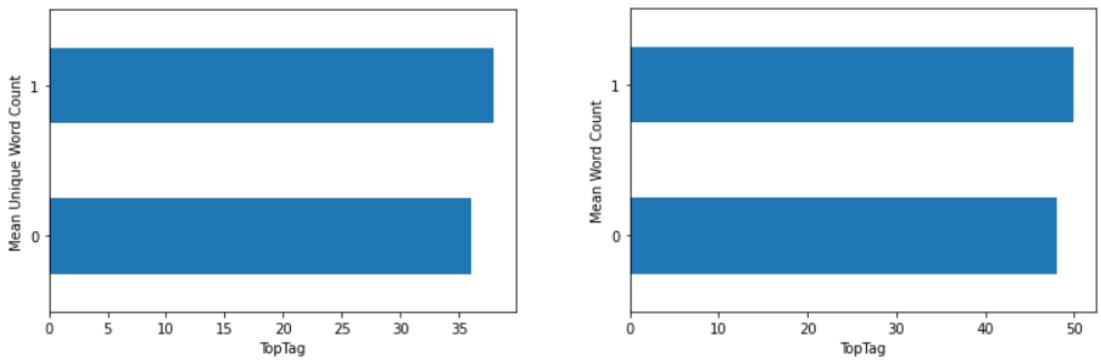


*Fig 17: Mean unique word / word counts between questions with Top Tags (1) and without Top Tags (0). Questions with Top Tags tend to have slightly more words but is a very weak correlation. Statistically significant?*

Inverse relationship between both ViewCount/ Score and question length.

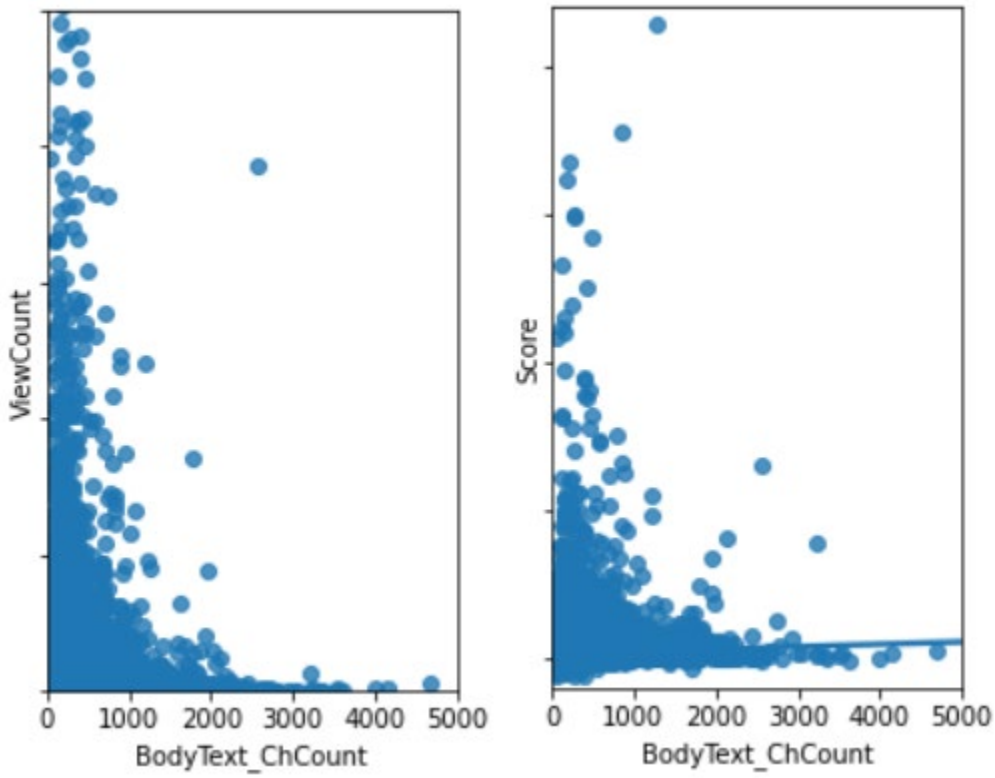



*Fig 18: Crossplots of ViewCount and Score versus character count per question showing the inverse relationship between these features and the counts.*

Inverse relationship between number of questions asked, bold count, and paragraph counts, body word count; this may speak to different styles of the question askers;
some are big questions askers, other are big on code examples, or bolding text for emphasis, but they tend to fit in separate categories or perhaps are related more specifically to tag topics. Also inverse relationship between ViewCount/Score and a large amount of questions, bold count, and paragraph counts.

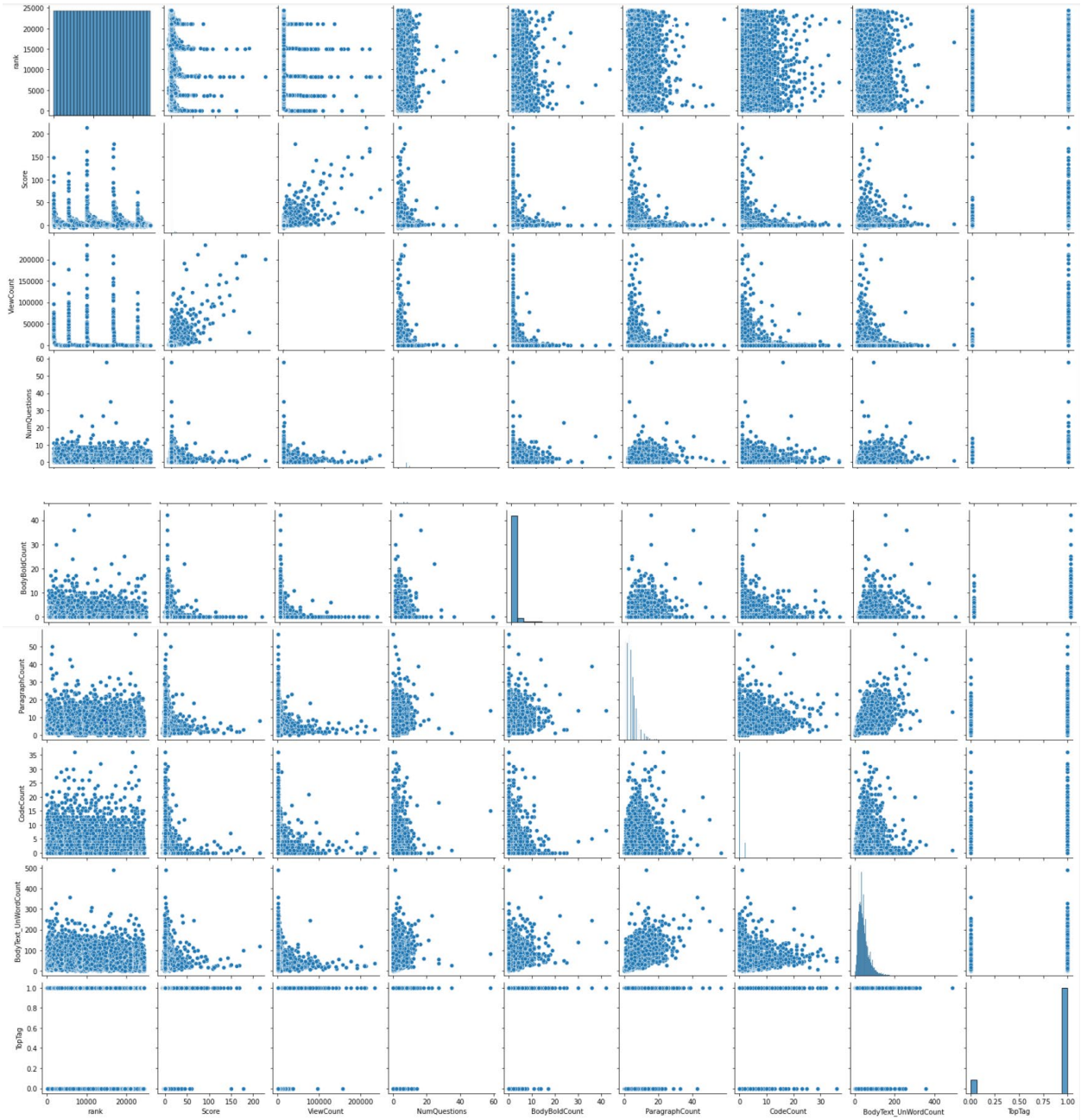



*Fig 19: Seaborn pairplot of counts of features.*

2nd pass of EDA will be performed and then this section enhanced after pass 1 of modelling.

## Final DataFrame

Output is 2 pandas dataframes - one has columns removed to count vectorization called …. The associated pickle file exported is named questions_df_ner_results_10282020.pickle.
The other is an archive of the complete dataframe with all feature / columns created called …. The associated pickle file exported is named questions_df_ner_results_10282020.pickle.

# 4. COUNT VECTORIZATION, TF-IDF and DOCUMENT EMBEDDINGS

The final step in data conditioning is to convert the text to a numerical format. Three different methods were employed and will be compared. The first process is count vectorization, the second is tf-idf. ELABORATE
And finally a more sophisticated method of vectorizing the text to create document embeddings will be employed. We will then be able to present each question as a vectors in feature space
The following pseudocode outlines the process to perform these tasks. These steps were performed in the "CP2_05_CountVectorization.ipynb" notebook.

# 5. FEATURE ENGINEERING

| *Features* | *Description* |
| --- | --- |
| *1.* | |

# 6. MACHINE LEARNING

# 7. REFERENCE

# 8. APPENDIX

## 8.1 Stack Exchange Data Science Data Schema