Data Analytics

Blog Authorship Analysis

Yunke Xiong

March, 2023

Table of content

Introduction

The rise of social media and online communication has led to an explosion in user-generated content since the 21st century. This has created a wealth of data that can be analyzed to gain insights into how people communicate and express themselves online. The Blog Authorship Corpus is one such dataset that contains a large amount of blog posts from a diverse set of authors, each with their age, gender, zodiac sign, and occupation.

In this project, we explore the potential of natural language processing (NLP) techniques to analyze this dataset and uncover insights into the demographic characteristics of the authors, as well as their sentiment and writing style. We begin by performing exploratory data analysis to gain a better understanding of the dataset, including the distribution of age, gender, and occupation among the authors, as well as any patterns that might show up in people's writing habits.

Next, we use various NLP techniques, including text preprocessing, feature extraction, and machine learning algorithms, to predict the demographic characteristics of the authors based on their writing style. Specifically, we aim to predict the gender, age group, and sentiment of the authors from their blog posts. We evaluate the performance of our models using various metrics and compare them to baseline models to determine their effectiveness.

Plan

1. Collect data from the appropriate source
2. Load, clean and transform data in Python
3. Export clean data to MySQL
4. Construct Database in MySQL
5. Basic exploratory data analysis
6. Deep dive EDA (with ML results)
7. Sentiment Analysis
8. Author profiling
9. Review Model

Data Collection

The dataset Blog Authorship Corpus is downloaded from the open source platform Kaggle in csv format and is the only dataset used in this analysis. Due to the large size of the dataset (304 MB with 681,288 blogs totalling over 140 millions words) no additional data sources have been retrieved using API or web scraping, as text processing and mining already take an enormous amount of time.

Dataset Overview

Blog Authorship Corpus

The raw data exported from kaggle consists of 7 columns:

Field Name	Description
id	id of the blog author
gender	self provided gender, male or female
age	integer from 13 to 48
topic	occupation of blog author
sign	zodiac sign of blog author
date	date of blog published
text	full text of blogs

Data Cleaning and Transformation

To construct the database, we first clean up the data and do basic transformation in Python. The final output will be a table with around 430,000 rows with no missing data and two additional columns – blog_id and word_count.

The above table will be loaded to MySQL and be the raw table that other tables base on.

Clean up

The dataset is pretty sane and complete with only a few NaN values in date, but from a closer look at the 'topic' column we can see that there are a lot of 'indUnk' which should be the abbreviation of industry unknown. As occupation would be used in following exploratory data analysis, we remove these rows together with NaN values in date and reset the index.

```
# remove rows without topic and date and reset index
data = data.loc[data['topic']!='indUnk',:]
data = data.dropna()
data = data.reset_index(drop = True)
```

Transform

A basic transform is performed before loading the dataset to MySQL, more transformation will be done with SQL query while creating relevant new tables.

First we need to give each blog a unique id so we can refer to it later. We also do a word count for each blog so we can have a grasp of the length of the blogs.

```
# create column for word count

data.loc[:,'word_count'] = data.loc[:,'text'].apply(lambda x: len(x.split()))



# create unique id for each blog

data.loc[:, 'blog_id'] = data.index.astype(str)

data.loc[:, 'blog_id'] = data.blog_id.apply(lambda x : '1'+x.zfill(6))
```

Date format cleaning

The original date format looks like dd-MM-YYYY (i.e. 14-May-2004) and is not super SQL friendly so we convert them into standard YYYY-mm-dd format. But when cleaning up with ***pd.to_datetime***() we found the dates are actually in multiple languages, for instance there's:

> '18,julio,2004' (spanish)
> '20,mars,2004' (french)
> '20,Junho,2004' (portuguese),
> …

To parse this kind of date properly we need to use the ***parse*** method from the *dateparser* library instead of ***to_datetime*** from *pandas*. This method is much more powerful and can parse datetime in different languages then return a datetime.datetime object. However it also takes much more time. Given the size of our dataset, error handling is also added to prevent any loss of progress. When there's any error, simply print out the location of the datapoint, the error message and then continue:

```
    for ind, row in data.iterrows():

        try:

            date_raw = row['date']

            clean_date = dateparser.parse(date_raw)

            data.loc[ind,'date_clean'] = clean_date

        except Exception as e:

            print(ind, e)
```

Text Processing

Text processing is the core of the machine learning part of the report since we rely on NLP techniques such as bag of words to do classification. For text cleaning we used Spacy and regex library.

```
nlp = spacy.load("en_core_web_sm")
```

```
stop_words = nlp.Defaults.stop_words

punctuations = string.punctuation
```

After importing the tokenizer and stopwords, we created a function to apply to the whole dataset. After tokenizing the text, we clean up the tokens, texts are converted into lower case, stripped of spaces and lemmatized so only the root words are kept.

Then we removed everything that's not-alphanumeric, all extra whitespace and all digits.

```
    # create token object

    doc = nlp(sentence)



    mytokens = [word.lemma_.lower().strip() for word in doc] # lemmatize token
and convert into lower case

    mytokens = [re.sub(r'\W+',' ',token) for token in mytokens] # replace
everything non-alpahnumeric by ' '

    mytokens = [re.sub(r'\s+',' ',token) for token in mytokens] # replace one or
more whitespaces by  ' '

    mytokens = [re.sub(r'\d+',' ',token) for token in mytokens] # replace one or
more digits by ' '



    # remove stopwords, punctuation and empty string

    mytokens = [word.strip() for word in mytokens if word not in stop_words and
word not in puntcuations]

    mytokens = [word for word in mytokens if word]

    clean_text = " ".join(mytokens)
```

In the end the punctuations and stop words are also removed as they do not carry any meanings.

Here are some texts before and after cleaning:

before:

Info has been found (+/- 100 pages, and 4.5 MB of .pdf files) Now i have to wait untill our team leader has processed it and learns html.

after:

info find page mb pdf file wait untill team leader process learn html

before:

One thing I love about Seoul (and I mean this about Korea in general...I just happen to be a little Seoul-centric) is the street sellers. I don't really trust the food they sell on the side of the road (except ice cream) but virtually everything else is fair game for me. For example, to get ready for my trip to Canada and generally stock up, in the last two weeks I bought: 2 plants in a can for my nieces (8,000 won) 2 lightweight sports shirts for inlining (10,000 won for both) 1 pair of shorts for inlining (20,000 won) 3 bags of dried 고구마 (goguma, sweet potatoes or yams, they were selling 1 for 3,000 won , 2 for 5,000 but I got 3 for 6,000) 1 tie (at an amazing price of 5,000 won, or USD 4...and I can't really tell how it's worse than the ones I bought for USD 100 back home.) 10 disposible razors (for 1,000 won, or USD 0.80 FOR ALL TEN) 1 noise-making toy hammer for boy #2 (1,000 won) 3 Disney photo albums (but I'm sure ol' Walt didn't make a penny on these....just 1,000 won each) The clothes-seller guy spoke pretty good English...I know because he held my hostage for 5 minutes as we talked about how Korean men are getting fatter (hence his stock of larger sizes for husky guys like me) and how he learned English working for the US Army about 20-30 years ago. The goguma-guy didn't know a lot of English, but he did speak Spanish owing to the fact that he lived in Argentina for a few years. Unfortunately, Spanish is not one of my languages...I know a fair bit of French from my school-days, a smattering Japanese and, of course, some Korean. Anyways, when I passed the goguma-guy later in the week I gave him a big 'hola!' (Spanish for hello, and the extent of my proficiency) and he returned one as well...wow, bridging the cultures with another one...how UN of me. Below is a picture of the famous Yeouido tie-truck. This guy stops in the hotspots and unloads silk and polyester ties on neckwear-hungry salarymen. urlLink Here they are: 2 for 5,000 won (USD 4.50)!

after:

thing love seoul mean korea general happen little seoul centric street seller trust food sell road ice cream virtually fair game example ready trip canada generally stock week buy plant niece win lightweight sport shirt inline win pair short inline win bag dry 고구마 goguma sweet potato yam sell win tie amazing price win usd tell bad buy usd home disposible razor win usd noise toy hammer boy win disney photo album sure old walt penny win clothe seller guy speak pretty good english know hold hostage minute talk korean man fat stock large size husky guy like learn english work army year ago goguma guy know lot english speak spanish owe fact live argentina year unfortunately spanish language know fair bit french school day smatter japanese course korean anyways pass goguma guy later week big hola spanish hello extent proficiency return wow bridge culture un picture famous yeouido tie truck guy stop hotspot unload silk polyester tie neckwear hungry salaryman urllink win usd

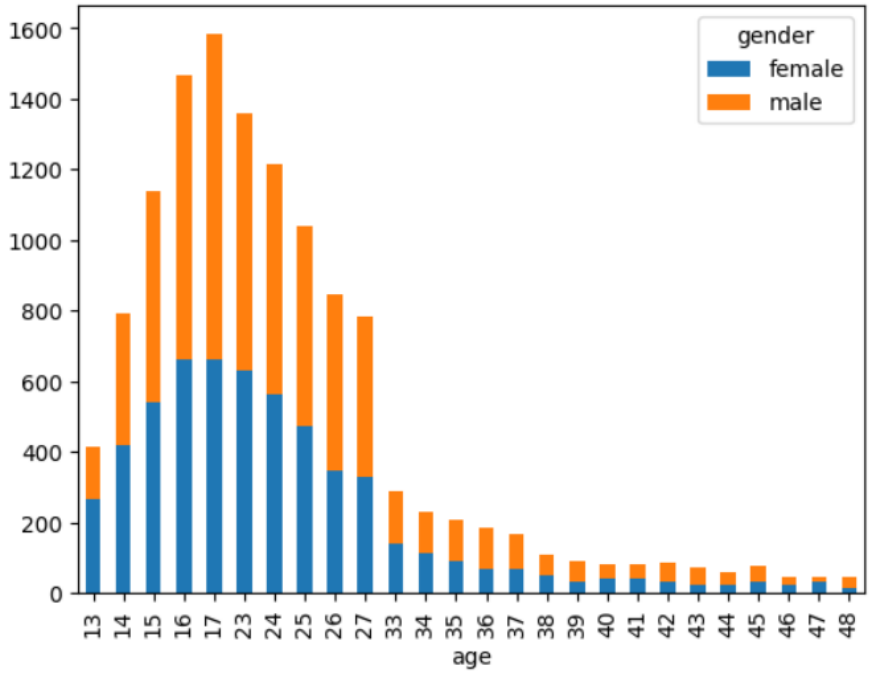
Exploratory Data Analysis

How old are the bloggers ?

Now that the data is ready, we can take a look at the composition of bloggers !
First we can see that the bloggers are mainly young people, most of them are under 30 with the youngest being only 13.

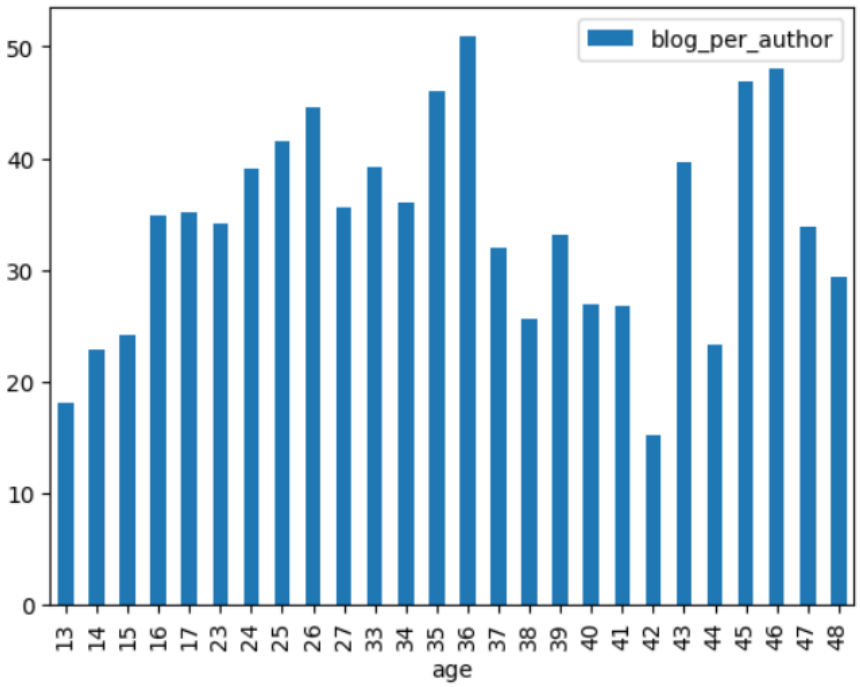
The authors can roughly be put into 4 age groups:
13-17 (teenagers)
23-27 (young professionals)
33-39 (adults)
40-48 (mid age)

The gender composition is quite balanced in each age group with roughly half men half women.



Are the youngest the most productive ?

But is the age group with most bloggers also the most active, or say productive ? To answer the question we can examine the number of blogs posted by each blogger, and we have the below results:

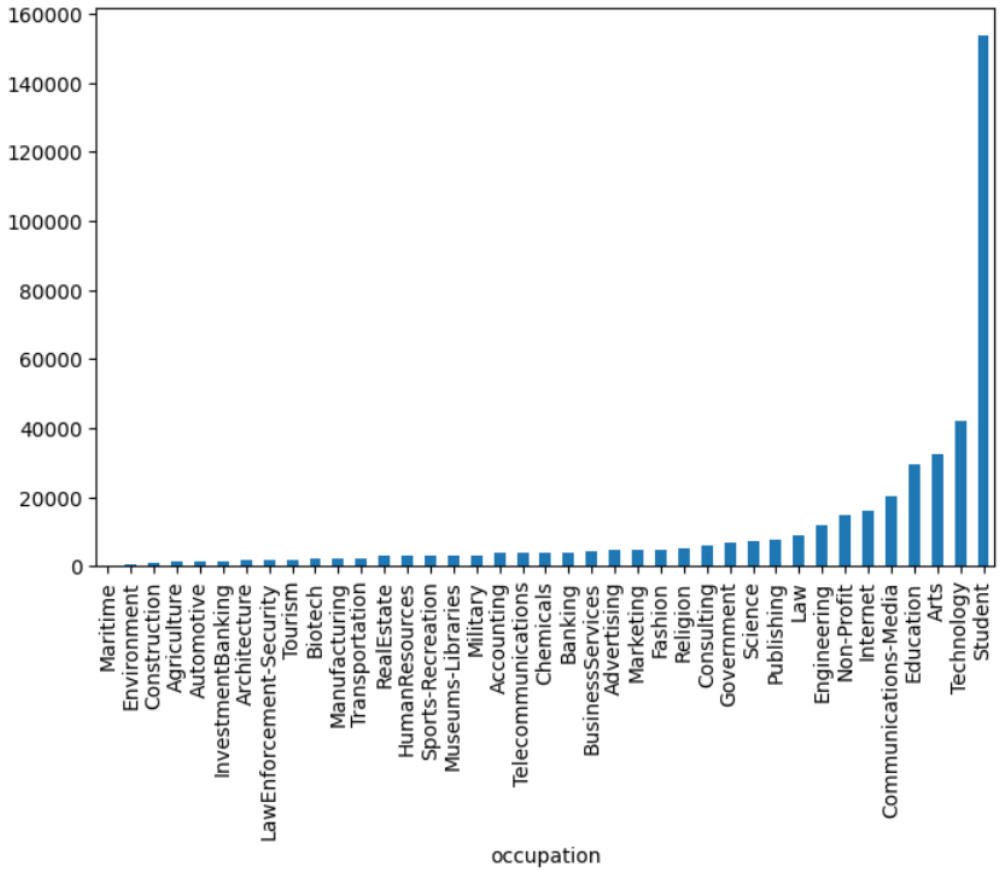


From the graph we can see that although teenagers constitute the largest proportion of bloggers, they were found to be less productive. Individuals at the age of 36 were observed to write the most number of blogs, while those aged 26, 45, and 46 were also noted to be quite productive as well. People from 23-27 and 33-36 seem to be the most productive group of people in terms of number of blogs.

What do bloggers do ?

Upon analyzing the age group, let us now turn our attention to the occupation of the bloggers. As expected, the majority of bloggers are students, which aligns with our previous findings. Among the non-student bloggers, the most common professions are technology, arts, education, media, and internet-related fields.

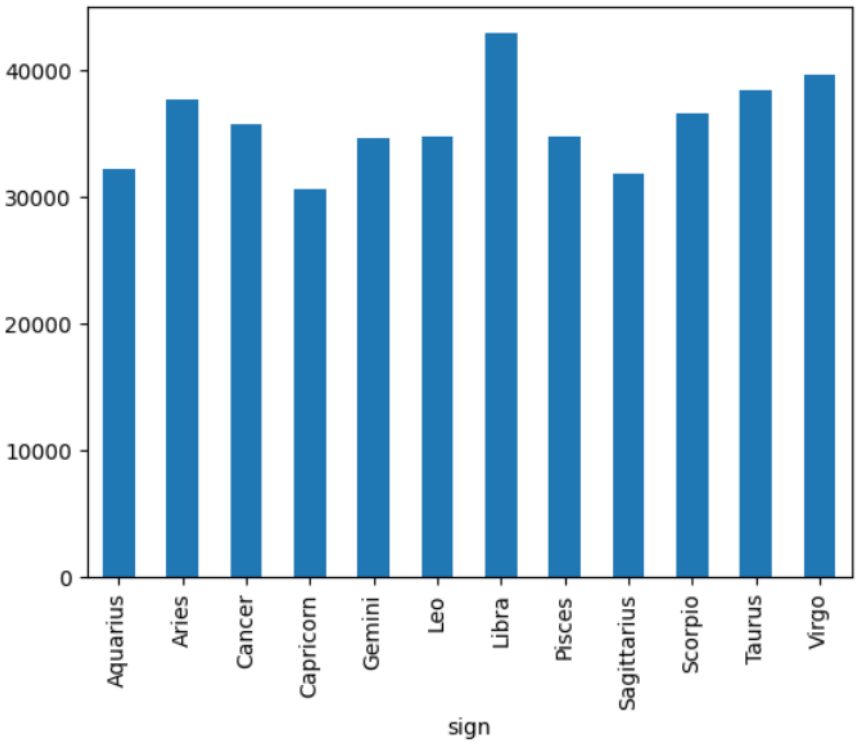
This outcome is reasonable given the time frame in which the dataset was created - back in 2004, when blogging was just beginning to emerge as a mainstream activity. Students, being young and eager to try new things, were likely among the early adopters of blogging, as were individuals in technology and internet-related industries who had early exposure to blogging platforms. The presence of individuals in arts and media industries is also unsurprising, as blogging is a form of social media and has become a means of spreading artistic content and connecting people with traditional media.



In case you believe in the Zodiac sign …

According to the findings, there are approximately 30,000 to 40,000 bloggers from each zodiac sign, indicating a **relatively even distribution** across the signs.

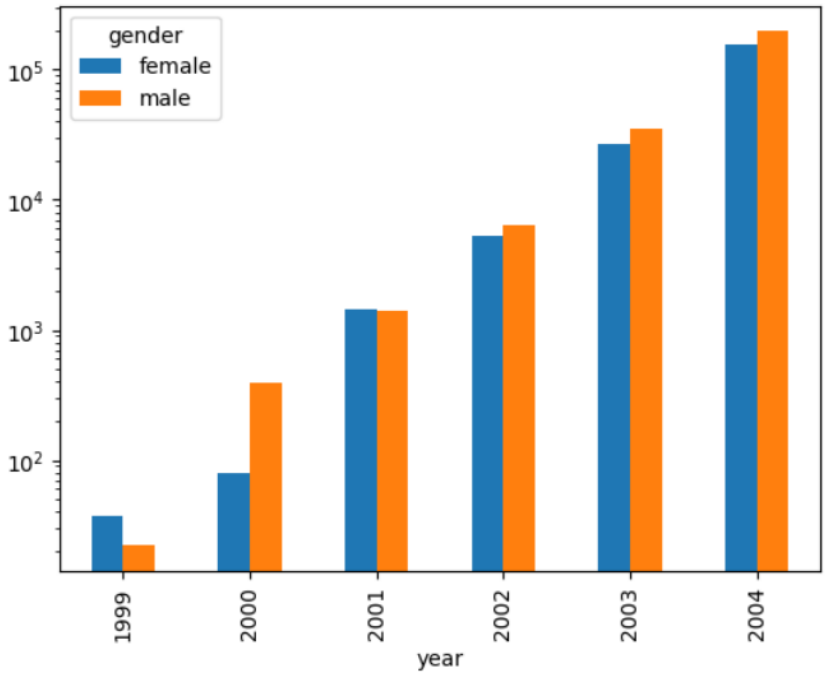
However, upon closer examination, there is a noticeable disparity between the largest and smallest groups. Specifically, the smallest group is the Capricorn sign, with approximately 30,000 bloggers, while the largest group is the Libra sign, with just over 40,000 bloggers. This represents a roughly **30% difference** in the number of bloggers between the two groups.



Evolution of weblogs

There was a significant surge in the number of weblog users between 1999 and 2004. The increase was exponential (that's why logarithmic y-scale is used). In 1999, there were a mere 59 bloggers, which rose to 480 in the following year, and increased even further to 2,800 in 2001. By 2002, the number of bloggers had risen to 12,000, and in 2003 it had jumped to 62,000. Finally, in 2004, there were a staggering 350,000 bloggers, marking a drastic increase over the years.

With regards to the gender composition of bloggers, in the first year, there were more female bloggers than male bloggers. However, in 2000, the number of male bloggers far exceeded that of female bloggers. In 2001, the distribution was relatively even between male and female bloggers. From 2002 to 2004, the number of male bloggers was roughly 24% higher than that of female bloggers.



Database & ERM

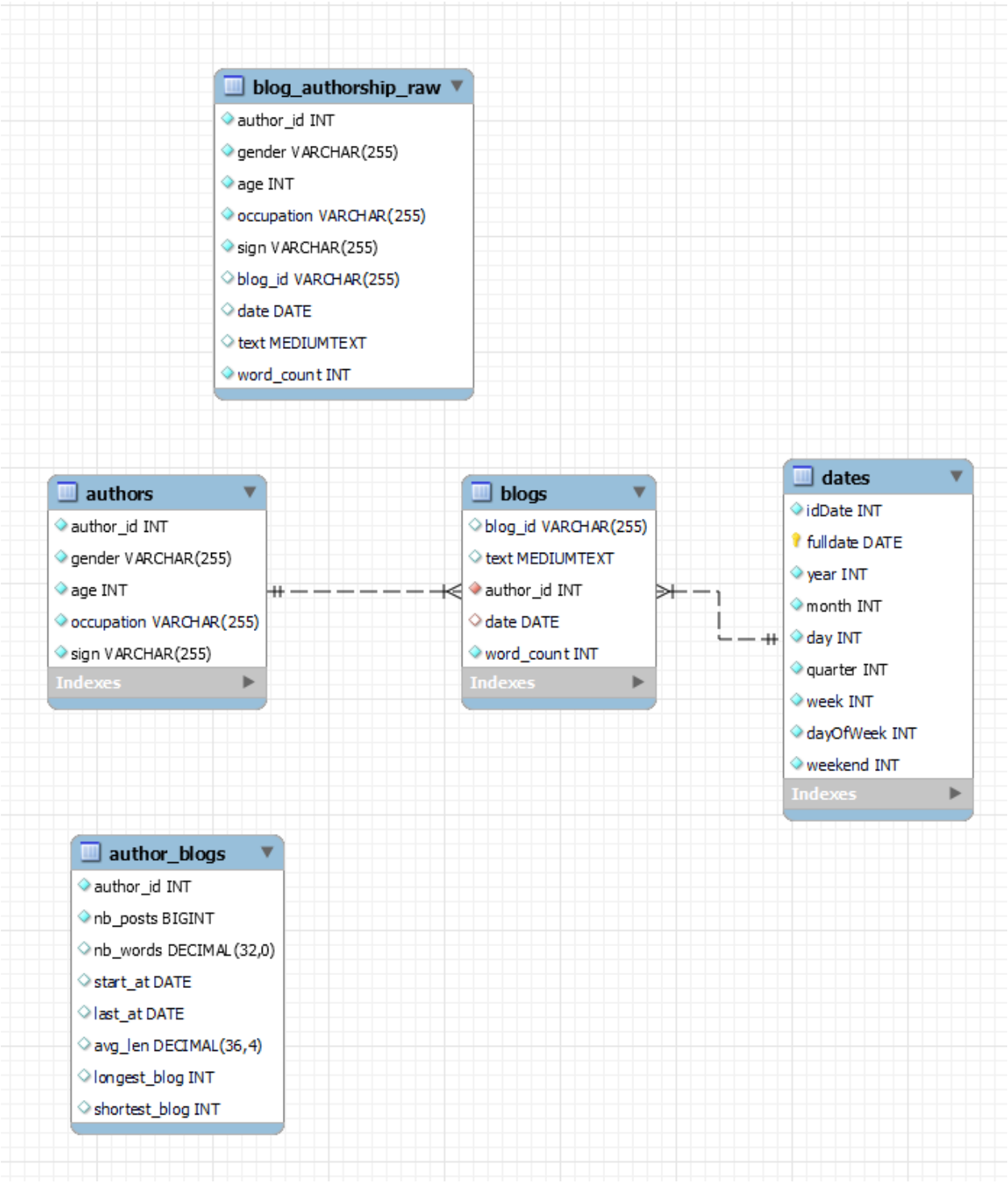
Why SQL ?

For the next step, I decided to create a database to store my cleaned and structured data. I opted to use SQL because it allows me to perform queries at a small level and link different tables.

When deciding between SQL and NoSQL databases, I ultimately chose SQL for a couple of reasons. First, SQL databases are relational, whereas NoSQL databases are non-relational. SQL databases also have a predefined schema, while NoSQL databases have dynamic schemas for unstructured data. Besides, SQL databases are vertically scalable, whereas NoSQL databases are horizontally scalable. Fourth, SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores and don't really suit my tabular data. Finally, SQL databases are better suited for multi-row transactions, while NoSQL databases are better suited for unstructured data such as documents or JSON.

ERM Diagram

The diagram for our entity relation model can be seen as follows,


blog_authorship_raw
author_id INT
gender VARCHAR(255)
age INT
occupation VARCHAR(255)
sign VARCHAR(255)
blog_id VARCHAR(255)
date DATE
text MEDIUMTEXT
word_count INT
authors
author_id INT
gender VARCHAR(255)
age INT
occupation VARCHAR(255)
sign VARCHAR(255)
Indexes
blogs
blog_id VARCHAR(255)
text MEDIUMTEXT
author_id INT
date DATE
word_count INT
Indexes
dates
idDate INT
fulldate DATE
year INT
month INT
day INT
quarter INT
week INT
dayOfWeek INT
weekend INT
Indexes
author_blogs
author_id INT
nb_posts BIGINT
nb_words DECIMAL(32,0)
start_at DATE
last_at DATE
avg_len DECIMAL(36,4)
longest_blog INT
shortest_blog INT

Database Construction

As mentioned in previous sections, we start with the raw dataset downloaded from Kaggle, then use Python to clean it up and do some enrichment. Then we output to the final table **blog_authorship_raw** to sql workbench, from this table we build **authors** and **blogs**, then a summary table **blog_per_au**, then a **dates** table that acts as a calendar tool. So only three tables are related, **blogs, authors** and **dates.**

blog_authorship_raw:

First step is to build an empty table to store the final output from Python and then load data from csv. However, probably due to the huge number of rows and large text, the Import Wizard only imported 7 rows even if the data was clean. So I found a work around using the command line. To make this work the file must be put at the location that shows with command SHOW VARIABLES LIKE “secure_file_prov”. In the beginning I also used Text as data type for blog text, then the loading failed due to some extra large text so I changed it to MEDIUMTEXT.

```
# create an empty table to load cleaned data
CREATE TABLE IF NOT EXISTS blog_authorship_raw (
author_id INT NOT NULL,
gender VARCHAR(255) NOT NULL,
age INT NOT NULL,
occupation VARCHAR(255) NOT NULL,
sign VARCHAR(255) NOT NULL,
blog_id VARCHAR(255),
date DATE,
text MEDIUMTEXT,
word_count INT NOT NULL
);

SHOW VARIABLES LIKE "secure_file_priv";

# load data into table
LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/blog_author_clean.csv"
INTO TABLE blog_authorship_raw
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

authors & blogs:

After the data is successfully loaded, the **authors** table was created using SELECT query, containing only authors related information to keep each table light. Table blogs are created in the same manner.

```
# create author table containing all author infomation
DROP TABLE IF EXISTS authors;
CREATE TABLE IF NOT EXISTS authors
SELECT DISTINCT author_id,
       gender,
       age,
       occupation,
       sign
FROM blog_authorship_raw ;
```

```
# create blog table containing all blog infomation
DROP TABLE IF EXISTS blogs;
CREATE TABLE IF NOT EXISTS blogs
SELECT blog_id ,
       text,
       author_id,
       date,
       word_count
FROM blog_authorship_raw ;
```

blogs_per_au:

A summary table blogs_per_au is created with granularity at author level, containing the number of posts, number of words, starting and last time they post a blog, average length of their blog etc.

```
# create author_blog table with aggregated information about each author
DROP TABLE IF EXISTS blogs_per_au;
CREATE TABLE IF NOT EXISTS blogs_per_au
SELECT author_id,
       COUNT(blog_id) AS nb_posts,
       SUM(word_count) AS nb_words,
       MIN(date) AS start_at,
       MAX(date) AS last_at,
       ROUND((SUM(word_count)/ COUNT(blog_id)),4) AS avg_len,
       MAX(word_count) AS longest_blog,
       MIN(word_count) AS shortest_blog
FROM blog_authorship_raw
GROUP BY author_id;
```

dates:

Lastly a dates table is created to act as a util table, first an empty table is created, then a procedure to populate dates and other needed information iterating through the timespan specified.

```
DROP TABLE IF EXISTS dates;
CREATE TABLE dates (
idDate INTEGER NOT NULL, -- year10000+month100+day
fulldate DATE PRIMARY KEY,
year INTEGER NOT NULL,
month INTEGER NOT NULL, -- 1 to 12
day INTEGER NOT NULL, -- 1 to 31
quarter INTEGER NOT NULL, -- 1 to 4
week INTEGER NOT NULL, -- 1 to 52/53
dayOfWeek INTEGER NOT NULL, -- 1 to 7
weekend INTEGER NOT NULL,
UNIQUE td_ymd_idx (year,month,day),
UNIQUE td_dbdate_idx (fulldate)

) Engine=innoDB;
```

```
DROP PROCEDURE IF EXISTS fill_date_dimension;
DELIMITER //
CREATE PROCEDURE fill_date_dimension(IN startdate DATE,IN stopdate DATE)
BEGIN
DECLARE currentdate DATE;
SET currentdate = startdate;
WHILE currentdate < stopdate DO
INSERT INTO dates VALUES (
YEAR(currentdate)*10000+MONTH(currentdate)*100 + DAY(currentdate),
currentdate,
YEAR(currentdate),
MONTH(currentdate),
DAY(currentdate),
QUARTER(currentdate),
WEEKOFYEAR(currentdate),

CASE DAYOFWEEK(currentdate)-1 WHEN 0 THEN 7 ELSE DAYOFWEEK(currentdate)-1 END ,
CASE DAYOFWEEK(currentdate)-1 WHEN 0 THEN 1 WHEN 6 then 1 ELSE 0 END);
SET currentdate = ADDDATE(currentdate,INTERVAL 1 DAY);
END WHILE;
END
//
DELIMITER ;



TRUNCATE TABLE dates;

CALL fill_date_dimension('1999-01-01','2005-01-01');
OPTIMIZE TABLE dates;
```

SQL Queries

Here are some analysis done using SQL query in the MySQL Workbench

1. Average number of blogs and blog length by zodiac sign

The total number of blogs is quite in line with the number of bloggers we saw earlier in the EDA section, we have less Capricorn bloggers and more Libra.

What's interesting here is that cancer people seem to be more fond of long blogs(300 words per blog on average) and Scorpio people are the opposite (261 words per blog on average).

```sql
WITH authors AS (
SELECT author_id, sign FROM authors),
blog_au AS (
SELECT author_id, nb_posts, nb_words, avg_len
FROM blogs_per_au)

SELECT sign,
SUM(nb_posts) AS total_posts,
SUM(nb_words) AS total_words,
AVG(avg_len) AS avg_words
FROM blog_au bau
LEFT JOIN authors a
    ON bau.author_id = a.author_id
GROUP BY sign
ORDER BY total_posts;
```

sign	total_posts	total_words	avg_words
Capricorn	30650	6482017	299.69967046
Sagittarius	31925	6431876	271.06576423
Aquarius	32236	6848887	284.85271751
Gemini	34633	7342535	288.25685188
Leo	34784	7521297	274.91009914
Pisces	34808	6613032	262.68724777
Cancer	35779	8052076	300.31573842
Scorpio	36574	7018427	261.60201158
Aries	37716	7254899	278.43669942
Taurus	38496	7684025	271.08620659
Virgo	39731	8235135	275.15979121
Libra	42913	8316706	282.03754013

2. Gender difference in posting

We can see that excluding the impact of gender composition we discovered in the EDA section earlier, there's no significant difference in terms of average posts or average blog length gender wise. The length of female bloggers are only slightly longer than the male counterparts (283 vs 275).

```sql
WITH authors AS (
SELECT author_id, gender FROM authors),
blog_au AS (
SELECT author_id, nb_posts, nb_words, avg_len
FROM blogs_per_au)

SELECT gender,
COUNT(bau.author_id) AS nb_bloggers,
SUM(nb_posts) AS total_posts,
(SUM(nb_posts) /COUNT(bau.author_id)) AS avg_posts,
AVG(avg_len) AS avg_words
FROM blog_au bau
LEFT JOIN authors a
    ON bau.author_id = a.author_id
GROUP BY gender
ORDER BY total_posts;
```

gender	nb_bloggers	total_posts	avg_posts	avg_words
female	5699	190032	33.3448	283.19381142
male	6793	240213	35.3618	275.48827727

3. Blog posts across the year

It's interesting how most of the blogs are posted during summer, way more than in any other season. Blogs are also slightly longer compared to the ones in other months of the year.

```sql
WITH blogs AS (
SELECT blog_id, date, word_count
FROM blogs)
SELECT month,
COUNT(blog_id) AS blog_posted,
SUM(word_count) AS words_posted,
SUM(word_count)/COUNT(blog_id) AS blog_len
FROM dates
LEFT JOIN blogs
    ON dates.fulldate = blogs.date
GROUP BY 1
ORDER BY 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

month	blog_posted	words_posted	blog_len
1	13708	2740101	199.8906
2	15553	3140192	201.9027
3	18985	4022171	211.8605
4	21344	4478359	209.8182
5	49821	10376058	208.2668
6	81149	16428860	202.4530
7	106093	22529173	212.3531
8	84322	16862241	199.9744
9	7604	1423524	187.2073
10	10480	1797727	171.5388
11	10774	1977322	183.5272
12	10412	2025184	194.5048

4. Blog posts across the week 1/

When trying to find patterns in terms of week of days, surprisingly there’re more blogs posted on Monday.

```sql
WITH blogs AS (
SELECT blog_id, date, word_count
FROM blogs)
SELECT dayOfWeek,
COUNT(blog_id) AS blog_posted,
SUM(word_count) AS words_posted,
SUM(word_count)/COUNT(blog_id) AS blog_len
FROM dates
LEFT JOIN blogs
    ON dates.fulldate = blogs.date
GROUP BY 1
ORDER BY 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

dayOfWeek	blog_posted	words_posted	blog_len
1	72023	14918421	207.1341
2	69071	13784942	199.5764
3	64111	12792760	199.5408
4	65424	13240489	202.3797
5	56246	11773978	209.3301
6	41975	8907528	212.2103
7	61395	12382794	201.6906

5. Blog posts across the week 2/

This query examine almost the same pattern as previous one except it only looks at the workday/weekday. Consistent with previous finding, people actually post more blogs on average during the week than over weekend, which is a bit counter-intuitive as we would assume people might tend to write more over the weekend when they're free.

```sql
WITH blogs AS (
SELECT blog_id, date, word_count
FROM blogs)
SELECT weekend,
COUNT(blog_id) AS blog_posted,
CASE WHEN weekend = 0 THEN COUNT(blog_id) / 5 ELSE COUNT(blog_id) /2 END AS daily_blog_posted,
SUM(word_count)/COUNT(blog_id) AS blog_len
FROM dates
LEFT JOIN blogs
    ON dates.fulldate = blogs.date
GROUP BY 1
ORDER BY 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

weekend	blog_posted	daily_blog_posted	blog_len
0	326875	65375.0000	203.4741
1	103370	51685.0000	205.9623

Conclusion

In summary, our analysis of weblog data from 1999 to 2004 provides insights into the demographics of bloggers, their productivity, and the growth of the blogosphere during this time period. We found that bloggers were primarily young people, with the majority being under 30 years old. The age groups of 23-27 and 33-36 were the most productive in terms of the number of blogs written, with individuals at the age of 36 writing the most. Most bloggers were students, with technology, arts, education, media, and internet-related fields being the most common professions among non-student bloggers.

Interestingly, we also observed a relatively even distribution of bloggers across the zodiac signs. However, there was a noticeable disparity between the largest and smallest groups, with the Libra sign having just over 40,000 bloggers and the Capricorn sign having only 30,000 bloggers. The surge in the number of weblog users between 1999 and 2004 was exponential, with the number of bloggers increasing from just 59 in 1999 to a staggering 350,000 in 2004.

Another notable finding was the gender composition of bloggers, which varied over time. While there were more female bloggers than male bloggers in the first year of the study, the number of male bloggers far exceeded that of female bloggers in 2000. In 2001, the distribution was relatively even between male and female bloggers, but from 2002 to 2004, the number of male bloggers was roughly 24% higher than that of female bloggers. These findings suggest that the gender gap in blogging may have widened during this time period.

Overall, our analysis provides valuable insights into the demographics and productivity of bloggers, as well as the growth and gender composition of the blogosphere during the early years of the 21st century. These insights may be useful for researchers and practitioners interested in understanding the role of blogging in online communication and self-expression, as well as its impact on society more broadly.

With the current cleaned data other insights can be generated with some NLP techniques, for example with sentiment analysis we can see if one occupation is happier than other, do people's sentiment change over time etc. We can also try to predict the demographic characteristics of the author and see if our model is accurate.