Wieloskalowa analiza danych z forum internetowego przy użyciu usług chmury AWS

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Michał Kamiński

Master of Science in Biology
Biodiversity, Evolution and Ecology
Department of Biological Sciences
University of Bergen

24.01.2023

Cover page
Beckwithia glacialis on Snøhetta.

Spis treści

Streszczenie

Słowa kluczowe: Big Data, Spark, AWS, EMR, S3

Summary

Keywords: Big Data, Spark, AWS, EMR, S3

Cześć I

Wstęp

Technologie big data

Ilość przetwarzanych danych cyfrowych na świecie rośnie w tempie logarytmicznym i obecnie podawana jest w dziesiątkach zettabajtów [2]. Wraz ze wzrostem ilości danych niezbędne jest optymalizowanie bądź zwiększanie miejsca potrzebnego na ich przechowywanie oraz ulepszanie algorytmów pozwalających na dokonanie analiz w czasie umożliwiającym na wyciąganie wniosków i podejmowania stosownych akcji. Znaczącym usprawnieniem procesu analitycznego było opracowanie algorytmu `MapReduce` [3], wykorzystującego równoległe przetwarzanie zbiorów danych w klastrach komputerowych. Opiera się na stosowaniu dwóch "kroków". Kroku `map` odpowiada za wykonaywanie zadań w węzłach roboczych (niezależnie od siebie) a następnie kroku `reduce`, który zbiera dane z węzłów roboczych i dokonuje kroku redukcji danych poprzez np. agregację.

Klastry komputerowe mogą być wykorzystywane nie tylko do obliczeń, ale również do przechowywania danych. Platformą, która wykorzystuje rozproszony system plików jest np. Apache Hadoop [1].

Formaty danych

Chmury

Cel pracy

Celem niniejszej pracy jest utworzenie infrastruktury w chmurze obliczeniowej AWS pozwalające na wielkoskalową analizy danych w sytemie rozproszonym (ang. *Big Data*).

Do stworzenia przykładowego projektu wykorzystano dane ze strony *Stack Exchange* zawierającej zestawy danych pochodzące z forów społecznościoowych. Analizę ograniczono do danych pochodzących z forum o nazwie *Beer, Wine and Spirits*.

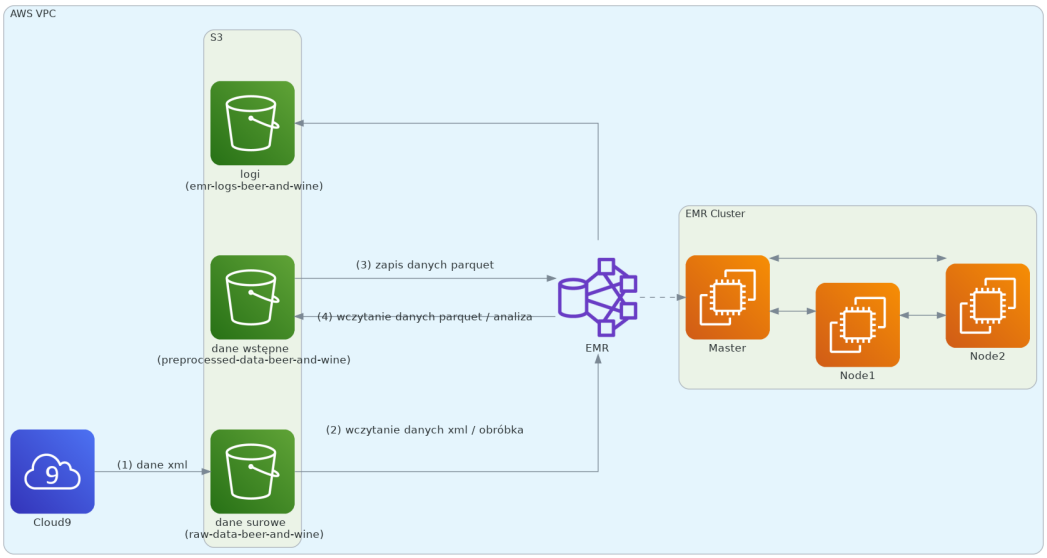
W niniejszej pracy … US Patent 7,650,331: "System and method for efficient large-scale data processing

Cześć II

Wyniki i Dyskusja

1 Schemat infrastruktury

```
Warning: node '9d224b90ad234d2390ad6c328134370e', graph '%3' size too small for label
Warning: node 'dd15172140ca43f8a22baf91a9712825', graph '%3' size too small for label
Warning: node '10c973252a2f4356a5b729b55c3a902d', graph '%3' size too small for label
Warning: Orthogonal edges do not currently handle edge labels. Try using xlabels.
```



Rysunek 1.1: Schemat rozwiązania

W celu rozwiązania postawionego problemu analitycznego stworzono infrastrukturę wyłącznie w obrębie chmury AWS, której ogólny schemat przedstawiono na Rysunek 1.1

1.1 Ekstrakcja

Do etapu ekstrakcji danych wykorzystano usługę `Cloud9`, która zapewnia dostęp do terminala maszyny wirtualnej z systemem linux (platforma `Amazon Linux 2`, typ instancji `t2.micro`). Z użyciem tej usługi dane zostały pobrane ze źródła w binarnym formacie `7z` a następnie pliki zostały wyekstrahowane w formacie `xml` przy pomocy programu `p7zip`. Dane w formacie `xml` zostały następnie skopiowane do

serwisu S3, gdzie utworzono koszyk danych (ang. *bucket*) o nazwie `raw-data-beer-and-wine`, którego przeznaczeniem jest przetwymywanie danych nieprzetworzonych.

Powyższe operacje zostały wykonane przy użyciu poniższych poleceń:

```
#| eval: false
#| echo: true

# instalacja programu p7zip
sudo yum install p7zip.x86_64

# pobranie danych
wget https://archive.org/download/stackexchange/beer.stackexchange.com.7z

# ekstrakcja danych do folderu raw-data
7za e  beer.stackexchange.com.7z -oraw-data

# zapis danych do koszyka S3 przy użyciu programu `AWS CLI`
aws s3 cp $(pwd)/raw-data s3://raw-data-beer-and-wine/ --recursive  --include "*.xml"
```

1.2 Przygotowanie danych wstępnych

W celu przygotowania danych do analizy, dane surowe zostały wstępnie przetworzone oraz zapisane w formacie `parquet`, co pozwoli na wydajniejsze wczytywanie danych podczas uruchomień programu. Podczas etapu wstępnego przetwarzania danych, oprócz zmiany formatu plików, zdefiniowane zostały także schematy danych, które zapewnią, że kolumny danych będą posiadały odpowiednie typy oraz, że krytyczne dane nie będą zawierały pustych wartości. Dodatkowo kolumny z wartościami tekstowymi, niesłownikowanymi zostały oczyszczone z tagów `html` oraz poddane standardowej procedurze oczyszania tekstu.

Powyższe czynności zostały wykonane w notatniku typu Jupiter (ang. ***`Jupyter Notebook`***) w serwisie `AWS EMR`. Stworzono klaster `EMR` (wersja 6.8.0) z instalacją `Hadoop` 3.2.1, `Jupyter Hub` oraz `Spark` 3.3.0, składający się z 1 instancji typu *master* oraz 2 instancji typu *core*, każda typu `m4.large`. W celu ograniczenia kosztów jako opcję zakupu wybrano typ `spot` z limitem maksymalnym ceny odpowiadającej typowi `on-demand`. Wielkość dysków `EBS` stworzonych instancji wynosiła 32 GiB dla każdej istancji w klastrze.

Polecenie programu AWS CLI odpowiadające za utworzenie klastra zajduje się w sekcji Sekcja 4.1.1.

Dostęp do `Jupyter Notebook` w utworzonym klastrze jest możliwy poprzez połącznie przez przeglądarkę z środowiskiem graficznym `Jupyter Hub` wykorzystując adres DNS instancji *master* i port 9443.

1.3 Budowa infrastruktury

Wszystkie serwisy AWS na potrzeby tego projektu zostały utworzone w sposób programatyczny przy użyciu programu `AWS CLI` (poza `Cloud9`, który został utworzony z poziomu konsoli zarządzającej). Wykorzystane polecenia dostępne są w sekcji Rozdział 4.

2 Wstępna obróbka danych

2.1 Konfiguracja aplikacji

W celu przygotowania danych do analizy zostały one wstępnie przetworzone. Pierwszym etapem wstępnego przetwarzania jest wczytanie danych do środowiska analitycznego. Dane surowe, przechowywane w koszyku `raw-data-beer-and-wine` znajowały się w mało przyjaznym dla analiz formacie `xml`. Wczytanie tego typu danych wymagało załadowania dodatkowego pakietu `jar` o nazwie `spark-xml_2.12:0.15.0` pobranego z repozytorium `maven`.

W serwisie `EMR` można dodać tego typu pakiety wykorzystując specjalne polecenia typy `Sparkmagic` rozpoczynające się od znaków `%%`. W tym przypadku użyto `%%configure`:

```
%%configure -f
{
    "conf": {
        "spark.jars.packages": "com.databricks:spark-xml_2.12:0.15.0"
    }
}
```

2.2 Schematy danych

W celu zapewnienia wczytania danych o oczekiwanych typach oraz zapewnieniu że nie ma tam danych brakujących utworzono schematy danych, wykorzystywane w kroku wczytywania z plików `xml`. Poniżej przedstawiono schematy dla każdego z przetwarzanych plików oraz przykładowe wiersze.

2.2.1 Plik `Users`

```
users_schema = StructType([
    StructField('_AboutMe', StringType(), True),
    StructField('_AccountId', IntegerType(), True),
    StructField('_CreationDate', TimestampType(), True),
    StructField("_DisplayName", StringType(), True),
    StructField("_DownVotes", IntegerType(), True),
    StructField("_Id", IntegerType(), True),
    StructField("_LastAccessDate", TimestampType()),
    StructField("_Location", StringType(), True),
    StructField("_ProfileImageUrl", StringType(), True),
    StructField("_Reputation", IntegerType(), True),
    StructField("_UpVotes", IntegerType(), True),
    StructField("_Views", IntegerType(), True),
    StructField("_WebsiteUrl", StringType(), True)
])
```

```
-RECORD 0-------------------------------------------------------------------------
 _AboutMe         | <p>Hi, I'm not really a person.</p>\n\n<p>I'm a backgroun...
 _AccountId       | -1
 _CreationDate    | 2014-01-21 17:45:53.587
 _DisplayName     | Community
 _DownVotes       | 478
 _Id              | -1
 _LastAccessDate  | 2014-01-21 17:45:53.587
 _Location        | on the server farm
 _ProfileImageUrl | null
 _Reputation      | 1
 _UpVotes         | 2
 _Views           | 5
 _WebsiteUrl      | http://meta.stackexchange.com/
only showing top 1 row
```

2.2.2 Plik `Tags`

```
tags_schema = StructType([
    StructField('_Count', IntegerType(), True),
    StructField('_ExcerptPostId', IntegerType(), True),
    StructField('_Id', IntegerType(), True),
    StructField("_TagName", StringType(), True),
    StructField("_WikiPostId", IntegerType(), True)
])
```

```
+------+--------------+---+-----------+-----------+
|_Count|_ExcerptPostId|_Id|   _TagName|_WikiPostId|
+------+--------------+---+-----------+-----------+
|    17|          5062|  1|       hops|       5061|
|    85|          7872|  2|    history|       7871|
|    69|          4880|  4|    brewing|       4879|
|    37|          5109|  5|    serving|       5108|
|    31|           304|  6|temperature|        303|
+------+--------------+---+-----------+-----------+
only showing top 5 rows
```

2.2.3 Plik `Votes`

```
votes_schema = StructType([
    StructField('_BountyAmount', IntegerType(), True),
    StructField('_CreationDate', TimestampType(), True),
    StructField('_Id', IntegerType(), True),
    StructField("_PostId", StringType(), True),
    StructField("_UserId", IntegerType(), True),
    StructField("_VoteTypeId", IntegerType(), True)
])
```

```
+-------------+--------------------+---+-------+-------+-----------+
|_BountyAmount|       _CreationDate|_Id|_PostId|_UserId|_VoteTypeId|
+-------------+--------------------+---+-------+-------+-----------+
```

```
|          null|2014-01-21 00:00:00|  1|      1|   null|          2|
|          null|2014-01-21 00:00:00|  2|      1|   null|          2|
|          null|2014-01-21 00:00:00|  3|      4|   null|          2|
|          null|2014-01-21 00:00:00|  4|      1|   null|          2|
|          null|2014-01-21 00:00:00|  5|      4|   null|          2|
+--------------+-------------------+---+-------+-------+-----------+
only showing top 5 rows
```

2.2.4 Plik `Posts`

```
posts_schema = StructType([
    StructField('_AcceptedAnswerId', IntegerType(), True),
    StructField('_AnswerCount', IntegerType(), True),
    StructField('_Body', StringType(), True),
    StructField("_ClosedDate", TimestampType(), True),
    StructField("_CommentCount", IntegerType(), True),
    StructField("_CommunityOwnedDate", TimestampType(), True),
    StructField("_ContentLicense", StringType(), True),
    StructField("_CreationDate", TimestampType(), True),
    StructField("_FavoriteCount", IntegerType(), True),
    StructField("_Id", IntegerType(), True),
    StructField("_LastActivityDate", TimestampType(), True),
    StructField("_LastEditDate", TimestampType(), True),
    StructField("_LastEditorDisplayName", StringType(), True),
    StructField("_LastEditorUserId", IntegerType(), True),
    StructField("_OwnerDisplayName", StringType(), True),
    StructField("_OwnerUserId", IntegerType(), True),
    StructField("_ParentId", IntegerType(), True),
    StructField("_PostTypeId", IntegerType(), True),
    StructField("_Score", IntegerType(), True),
    StructField("_Tags", StringType(), True),
    StructField("_Title", StringType(), True),
    StructField("_ViewCount", IntegerType(), True),
])
```

```
-RECORD 0-------------------------------------------------------------------------
 _AcceptedAnswerId      | 4
 _AnswerCount           | 1
 _Body                  | <p>I was offered a beer the other day that was reportedly...
 _ClosedDate            | null
 _CommentCount          | 0
 _CommunityOwnedDate    | null
 _ContentLicense        | CC BY-SA 3.0
 _CreationDate          | 2014-01-21 20:26:05.383
 _FavoriteCount         | null
 _Id                    | 1
 _LastActivityDate      | 2014-01-21 22:04:34.977
 _LastEditDate          | 2014-01-21 22:04:34.977
 _LastEditorDisplayName | null
 _LastEditorUserId      | 8
 _OwnerDisplayName      | null
 _OwnerUserId           | 7
 _ParentId              | null
 _PostTypeId            | 1
```

```
 _Score                 | 21
 _Tags                  | <hops>
 _Title                 | What is a citra hop, and how does it differ from other hops?
 _ViewCount             | 2434
only showing top 1 row
```

2.2.5 Plik `Post Links`

```
links_schema = StructType([
    StructField("_CreationDate", TimestampType()),
    StructField("_Id", IntegerType()),
    StructField("_LinkTypeId", IntegerType()),
    StructField("_PostId", IntegerType()),
    StructField("_RelatedPostId", IntegerType())
])
```

```
+----------------------+---+----------+-------+-------------+
|_CreationDate         |_Id|_LinkTypeId|_PostId|_RelatedPostId|
+----------------------+---+----------+-------+-------------+
|2014-01-21 21:04:25.23 |25 |3         |29     |25           |
|2014-01-21 21:42:09.103|89 |1         |83     |50           |
|2014-01-21 21:50:41.313|95 |1         |86     |2            |
|2014-01-21 22:07:35.783|101|3         |47     |99           |
|2014-01-21 22:13:51.38 |102|1         |74     |3            |
+----------------------+---+----------+-------+-------------+
only showing top 5 rows
```

2.2.6 Plik `Post History`

```
history_schema = StructType([
    StructField("_Comment", StringType()),
    StructField("_ContentLicense", StringType()),
    StructField("_CreationDate", TimestampType()),
    StructField("_Id", IntegerType()),
    StructField("_PostHistoryTypeId", IntegerType()),
    StructField("_PostId", IntegerType()),
    StructField("_RevisionGUID", StringType()),
    StructField("_Text", StringType()),
    StructField("_UserDisplayName", StringType()),
    StructField("_UserId", IntegerType()),
])
```

```
-RECORD 0-------------------------------------------------------------------------
 _Comment           | null
 _ContentLicense    | CC BY-SA 3.0
 _CreationDate      | 2014-01-21 20:26:05.383
 _Id                | 1
 _PostHistoryTypeId | 2
 _PostId            | 1
 _RevisionGUID      | a17002a0-00b0-417b-a404-0d8864bbbca5
 _Text              | I was offered a beer the other day that was reportedly ma...
```

```
 _UserDisplayName   | null
 _UserId            | 7
only showing top 1 row
```

2.2.7 Plik `Badges`

```
badges_schema = StructType([
    StructField("_Class", IntegerType()),
    StructField("_Date", TimestampType()),
    StructField("_Id", IntegerType()),
    StructField("_Name", StringType()),
    StructField("_TagBased", BooleanType()),
    StructField("_UserId", IntegerType()),
])
```

```
+------+----------------------+---+-------------+---------+-------+
|_Class|_Date                 |_Id|_Name        |_TagBased|_UserId|
+------+----------------------+---+-------------+---------+-------+
|3     |2014-01-21 20:52:16.97|1  |Autobiographer|false    |1      |
|3     |2014-01-21 20:52:16.97|2  |Autobiographer|false    |2      |
|3     |2014-01-21 20:52:16.97|3  |Autobiographer|false    |6      |
|3     |2014-01-21 20:52:16.97|4  |Autobiographer|false    |7      |
|3     |2014-01-21 20:52:16.97|5  |Autobiographer|false    |9      |
+------+----------------------+---+-------------+---------+-------+
only showing top 5 rows
```

2.3 Czyszczenie kolumn tekstowych

Pliki `Users`, `History` oraz `Posts` zawierają koumny tekstowe z danymi wpisywanymi przez użytkowników oraz często zawierające znaki specjalne czy tagi html. W związku z tym kolumny `_AboutMe` (plik `Users`), `_Text` (plik `History`) oraz `_Body` (plik `Posts`) zostały poddane procesowi oczyszczania.

W tym celu utworzono funkcję `UDF` o nazwie `tags_remove()` Sekcja 4.2.1, która odpowiada za usunięcie tagów html.

Dodatkowo znaki `\n`, `\t`, `\r` oraz podwójne spacje zastąpiono pojedynczymi znakami spacji a także usunięto znaki spacji z początków i końców wartości tekstowych.

Poniżej zaprezentowano przykłady kolumn nieoczyszczonych oraz po ich oczyszczeniu (zawierające końcówkę `_clean`):

- Dla pliku `Users`:

```
[Stage 7:>                                                          (0 + 1) / 1]
```

```
-RECORD 0-------------------------------------------------------------------
 _AboutMe       | <p>Hi, I'm not really a person.</p>\n\n<p>I'm a backgroun...
 _AboutMe_clean | Hi, I'm not really a person. I'm a background process tha...
-RECORD 1-------------------------------------------------------------------
```

```
 _AboutMe       | <p>Dev #2 who helped create Stack Overflow currently work...
 _AboutMe_clean | Dev #2 who helped create Stack Overflow currently working...
-RECORD 2-------------------------------------------------------------------
 _AboutMe       | <p>Former Stack Exchange employee</p>\n
 _AboutMe_clean | Former Stack Exchange employee
-RECORD 3-------------------------------------------------------------------
 _AboutMe       | \n<p>Developer at Stack Overflow focusing on public Q&amp...
 _AboutMe_clean | Developer at Stack Overflow focusing on public Q&A. Russi...
-RECORD 4-------------------------------------------------------------------
 _AboutMe       | <p><strong>BY DAY:</strong> I execute projects on both mo...
 _AboutMe_clean | BY DAY: I execute projects on both mobile and on the web ...
only showing top 5 rows
```

- Dla pliku `History`:

```
-RECORD 0-------------------------------------------------------------------
 _Text       | I was offered a beer the other day that was reportedly ma...
 _Text_clean | I was offered a beer the other day that was reportedly ma...
-RECORD 1-------------------------------------------------------------------
 _Text       | What is a citra hop, and how does it differ from other hops?
 _Text_clean | What is a citra hop, and how does it differ from other hops?
-RECORD 2-------------------------------------------------------------------
 _Text       | <hops>
 _Text_clean |
-RECORD 3-------------------------------------------------------------------
 _Text       | As far as we know, when did humans first brew beer, and w...
 _Text_clean | As far as we know, when did humans first brew beer, and w...
-RECORD 4-------------------------------------------------------------------
 _Text       | When was the first beer ever brewed?
 _Text_clean | When was the first beer ever brewed?
only showing top 5 rows

/config/workspace/env/lib/python3.10/site-packages/bs4/__init__.py:435: MarkupResemblesLocatorW
  warnings.warn(
```

- Dla pliku `Posts`:

```
-RECORD 0-------------------------------------------------------------------
 _Body       | <p>I was offered a beer the other day that was reportedly...
 _Body_clean | I was offered a beer the other day that was reportedly ma...
-RECORD 1-------------------------------------------------------------------
 _Body       | <p>As far as we know, when did humans first brew beer, an...
 _Body_clean | As far as we know, when did humans first brew beer, and w...
-RECORD 2-------------------------------------------------------------------
 _Body       | <p>How is low/no alcohol beer made? I'm assuming that the...
 _Body_clean | How is low/no alcohol beer made? I'm assuming that the be...
-RECORD 3-------------------------------------------------------------------
```

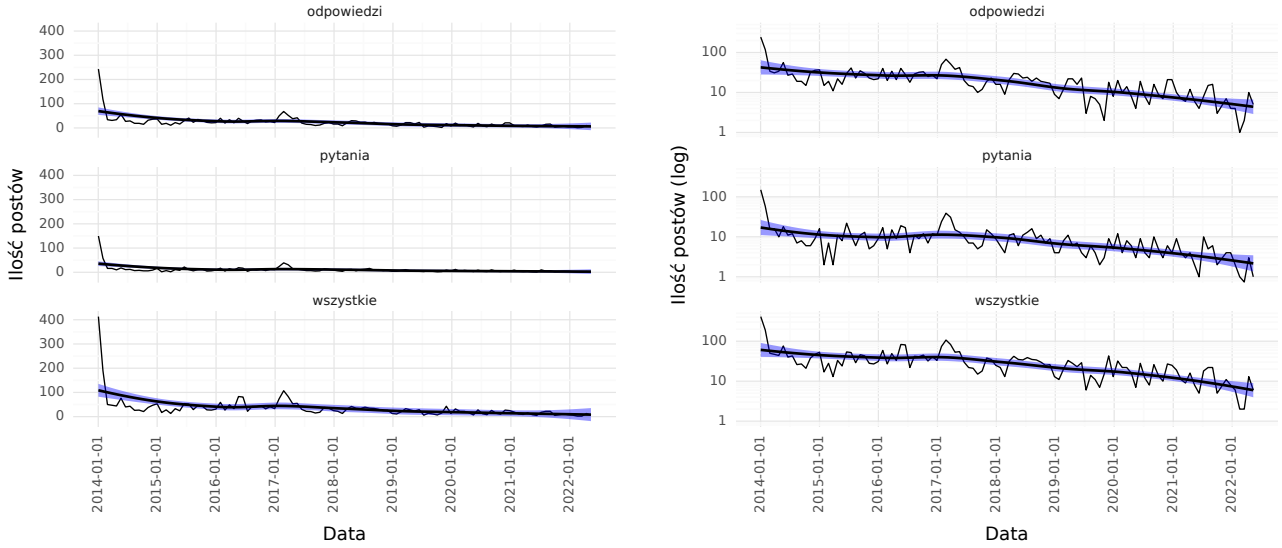
```
 _Body       | <p>Citra is a registered trademark since 2007. Citra Bran...
 _Body_clean | Citra is a registered trademark since 2007. Citra Brand h...
-RECORD 4-----------------------------------------------------------------
 _Body       | <p>In general, what's the best way to work out the temper...
 _Body_clean | In general, what's the best way to work out the temperatu...
only showing top 5 rows
```

Tak przygotowane dane, w celu optymalizacji miejsca zajmowanego w koszyku S3, oraz w celu przyspieszenia procesu wczytywania danych zapisano w formacie `parquet`. Zaskutkowało to redukcją zajmowanego miejsca w koszuku o około 50%.

3 Analiza danych

3.1 Analiza aktywności użytkowników na forum

Jako pierwsze postanowiono zbadać czy forum jest aktywne. W tym celu liczbę postów zagregowano w miesięczne insterwały i zliczono ich ilość. Pierwsza wiadomość pojawiła się na forum 2014-01-21 natomiast ostatnia 2022-06-05. Na przestrzeni tych ~8,5 roku pojawiło się 3769 postów. Z Rysunek 3.1 widać, że zainteresowanie forum spadało w czasie. W rekordowych pierwszych 2 miesiącach umieszczano ich odpowiednio 413 oraz 190, natomiast pod koniec badanego okresu wartości często nie przekraczały 10.

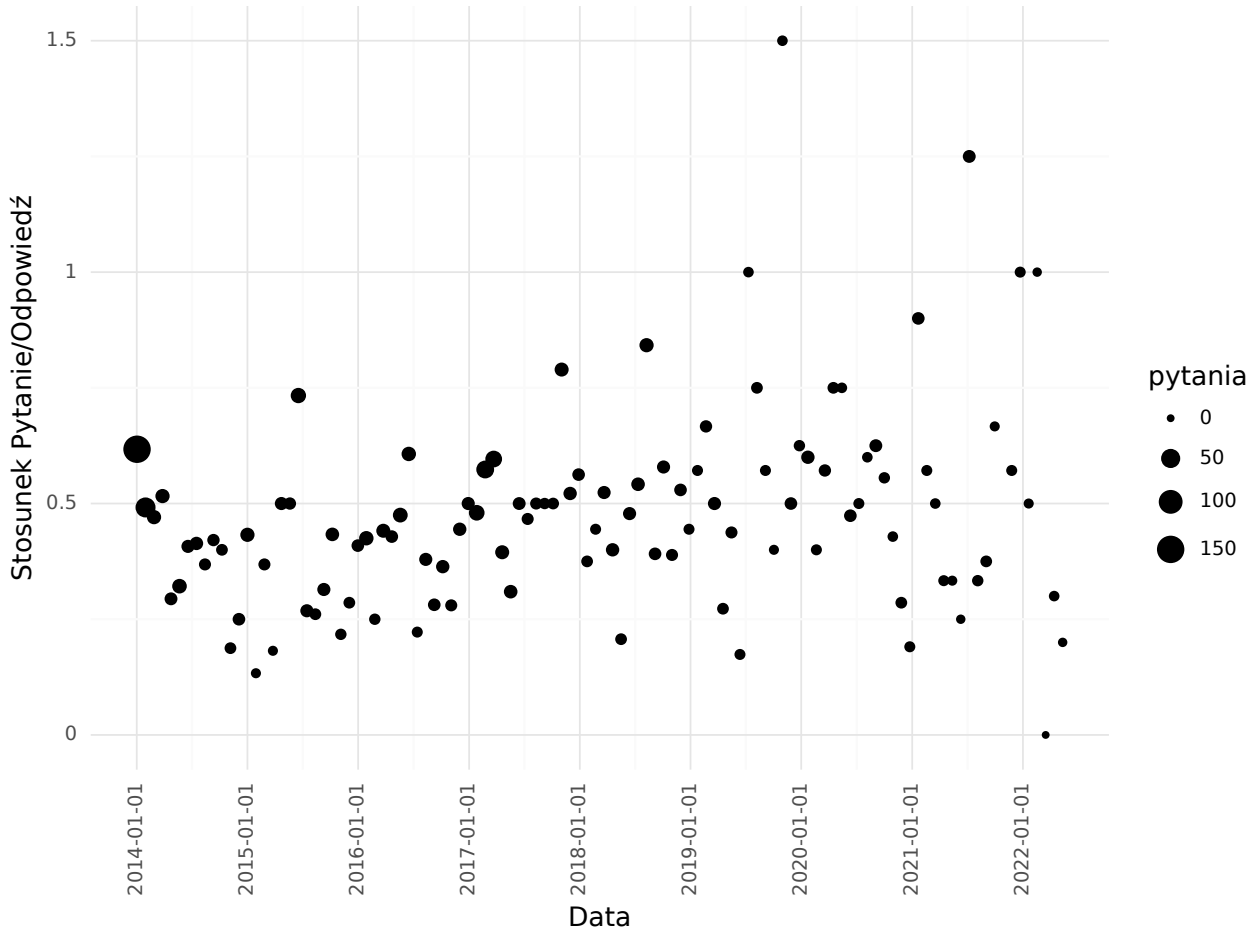


Rysunek 3.1: Liczba postow w czasie z podzialem na pytania i odpowiedzi. Lewy panel wskazuje surowe wartości liczby postów, prawy wartości logarytmiczne o podstawie 10.

Stosunek ilości odpowiedzi do zadawanych pytań rośnie w czasie, lecz ma na to wpływ zmniejszająca się ilość zadawanych pytań (maleje próba badana) co zostało przedstawione na Rysunek 3.2. Patrząc na wartości średnie tylko co drugie pytanie uzyskiwało odpowiedź (ratio 0.48 +/- 0.22). Ogólne statystyki stosunku pytań do dopowiedzi w czasie zostały przedstawione w Tabela 3.1.

Tabela 3.1: Statystyki stosunku ilości odpowiedzi na pytania. Bez uwzględnienia czy było ono zaakceptowane czy też nie.

	średnia	odchylenie standardowe	min	max	mediana
0	0.476316	0.219824	0.0	1.5	0.444444



Rysunek 3.2: Stosunek ilości odpowiedzi na zadane pytania w czasie, uwzględniając ilość zadanych pytań

Tabela 3.2: Statystyki ocen odpowiedzi zaakceptowanych w stosunku do pozostałych

	is_accepted	avg_score	std_score	min_score	max_score	count(a_id)
0	None	2.755169	3.181837	-5	30	919
1	True	6.395044	5.915949	0	46	686
2	False	2.584169	2.735329	-4	26	897

Tabela 3.3: Statystyki ocen odpowiedzi zaakceptowanych w stosunku do pozostałych

	avg(time_to_accept_min)	stddev_samp(time_to_accept_min)	quantiles
0	25244.943571	123338.63251	[141.53, 753.25, 3605.8]

3.2 Dynamika oraz statystyki udzielanych odpowiedzi

Twórcy pytań na forum mają możliwość wybrania odpowiedzi, która jest najtrafniejsza i zawiera poprawną odpowiedź. Te odpowiedzi nazywane są zaakceptowanymi odpowiedziami. Zbadano jak często najwyżej oceniona odpowiedź nie była zaakceptowaną odpowiedzią.

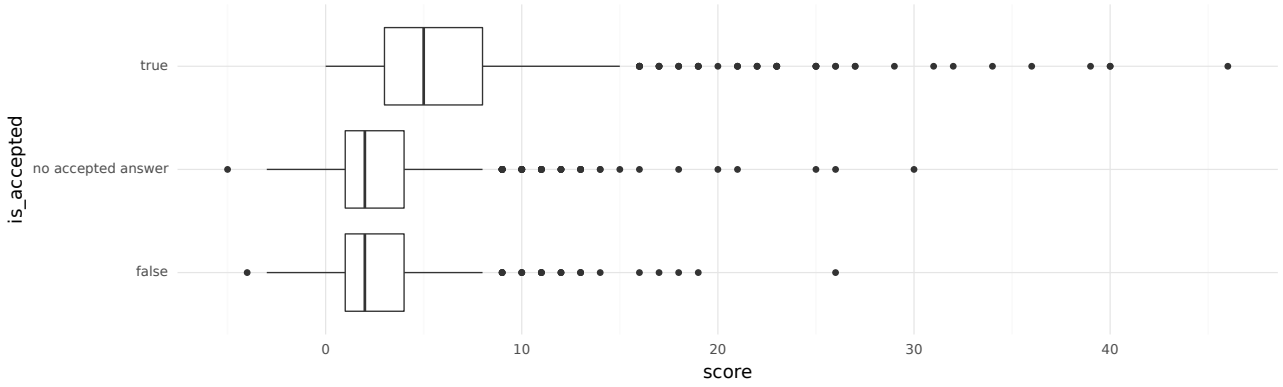
Okazało się, że w przypadku pytań, które posiadały jakąkolwiek odpowiedź około 12.7% (646) przypadków najlepiej ocenianych odpowiedzi nie było tą, która została zaakceptowana przez autora.

Następnie porównano oceny odpowiedzi zaakceptowanych z pozostałymi odpowiedziami a statystyki przedstawiono w Tabela 3.2 oraz Rysunek 3.3.

Pośród odpowiedzi na zadawane pytania wyróżniono 3 kategorie:

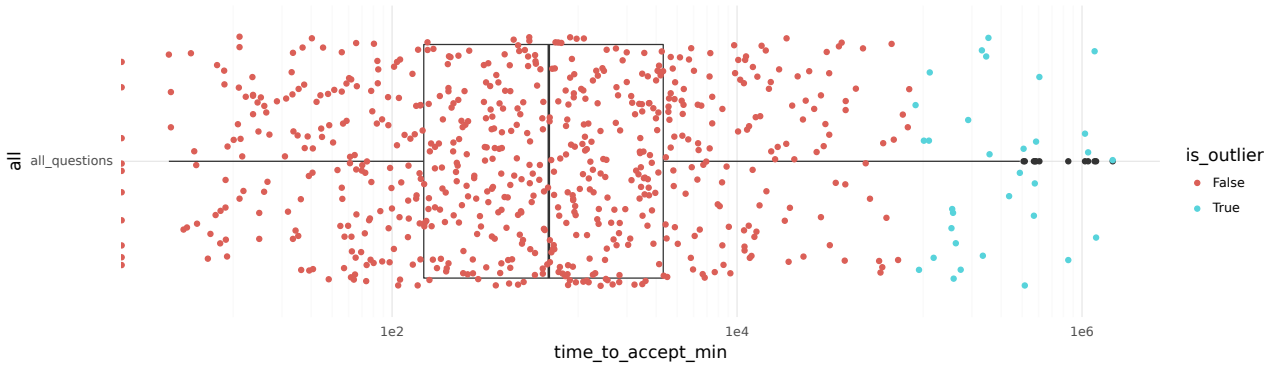
1. odpowiedź zaakceptowana (`is_accepted=True`)
2. odpowiedź niezaakceptowana (`is_accepted=False`)
3. odpowiedź niezaakceptowana ale równocześnie brak jest zaakceptowanej odpowiedzi na to pytanie (`is_accepted=None`)

Z analizy wynika, iż zaakceptowane odpowiedzi mają średnio wyższe oceny użytkowników (6,39) niż pozostałe oceny (2.75 - `is_accepted=None`; 2.58 - `is_accepted=False`), co było oczekiwanym wynikiem.



Rysunek 3.3: Rozkład ocen pytań zaakceptowanych w porównaniu do pozostałych

Następnie zbadano jak szybko od pojawienia się pytania, pojawia się zaakceptowana odpowiedź. Dla wszystkich pytań na tym forum jest to średnio 25244 minuty, ale wartość środkowa (753) sugeruje,



Rysunek 3.4: Rozkład czasu od pojawienia się pytania do zaakceptowanej odpowiedzi

Tabela 3.4: Statystyki ocen odpowiedzi zaakceptowanych w stosunku do pozostałych po odrzuceniu wartości odstających > 100000

	avg(time_to_accept_min)	stddev_samp(time_to_accept_min)	quantiles
0	4769.308792	12694.627624	[127.72, 644.93, 2922.82]

że średnia może być zaburzona. W celu obliczenia średniej nie zaburzonej tak dużymi wartościami odstającymi odrzucono wartości powyżej 100000 minut. Tym razem uzyskano wartość 4769 minut (~3 dni), przy wartości środkowej 644 (~10 godzin).

Rozkłady wartości przedstawiono na Rysunek 3.4 oraz w Tabela 3.3 dla całego zbioru danych oraz Tabela 3.4 po odrzuceniu wartości odstających.

3.3 Retencja użytkowników

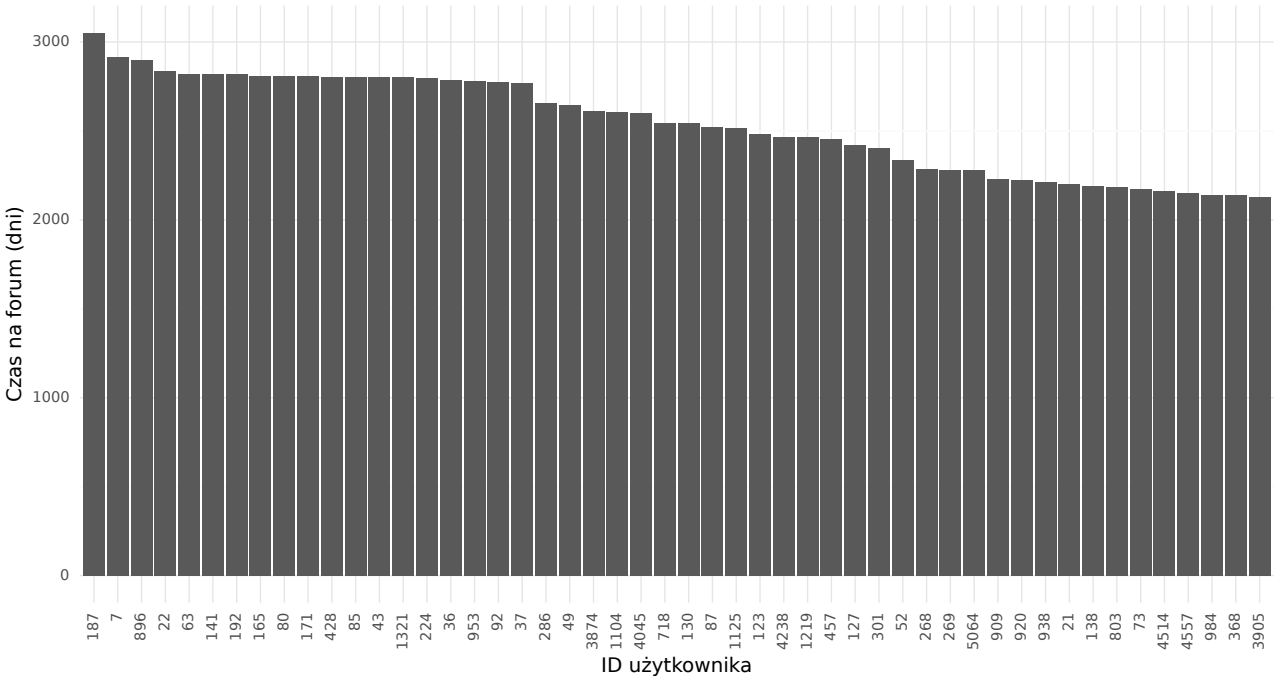
W całej historii forum zarejestrowanych zostało 8948 użytkowników, z czego jedno konto jest kontem bota. Miarą retencji użytkownika na forum określono czas od utworzenia konta to ostaniej zamieszczonej wiadomości. Wśród 50 najdłużej aktywnych kont zidentyfikowano konto z wynikiem aż 3052 dni a konto z ostatnim indeksem w tej grupie miało wynik 2128 dni. Jednakże patrząc na całą populację, wartość środkowa wyniosła 11, a 50% wartości mieści się pomiędzy 0 a 594 dniami. Użytkownicy z wartością 0, są to najprawdopodobniej użytkowicy, którzy zadali jedyne pytanie na forum w dniu założenia konta (410 użytkowników). Czas na forum najdłużej aktywnych użytkowników został zwizualizowany na Rysunek 3.5. Dodatkowo okazało się, że 7691 (86%) kont było biernymi użytkownikami forum i nidgy nie dodało żadnego posta.

3.4 Statystyki najwyżej oraz najniżej ocenianych pytań

Zbadano statystyki zadawanych pytań. Użytkownicy forum mogą oceniać pojawiające się tam pytania czego miarą jest wartość `score`. Sprawdzono, czy długość zadanego pytania ma wpływ na jego ocenę.

Średnia długość pytania wyniosła 415 znaków (+/- 330) a wartość środkowa 331 znaków. Najdłuższe pytanie posiadało 3133 znaki a najkrótsze jedynie 30. Statystyki przedstawiono w Tabela 3.5.

Średnio pytanie było oceniane na wartość 6.28 (+/-5.88) a wartość środkowa wyniosła 5. Najlepiej oceniane pytanie miało ocenę 67 a najgorzej -7. Statystyki zestawiono w Tabela 3.6.



Rysunek 3.5: Czas na forum 50 najdłużej aktywnych kont

Tabela 3.5: Statystyki długości postów

	średnia	odchylenie standardowe	max	min	mediana
0	415.863676	330.836043	3133	30	331

Tabela 3.6: Statystyki ocen użytkowników

	średnia	odchylenie standardowe	max	min	mediana
0	6.278804	5.876114	68	-7	5

Tabela 3.7: Statystyki długości postów z podziałem na podgrupy najlepszych i najgorszych pytań

	type	średnia	odchylenie standardowe	max	min	mediana
0	top	349.08	275.088587	1823	46	261
1	bottom	389.26	424.654836	3133	30	270

Tabela 3.8: Statystyki ocen użytkowników z podziałem na podgrupy najlepszych i najgorszych pytań

	type	średnia	odchylenie standardowe	max	min	mediana
0	top	20.6	8.551685	68	14	18
1	bottom	0.4	1.206045	1	-7	1

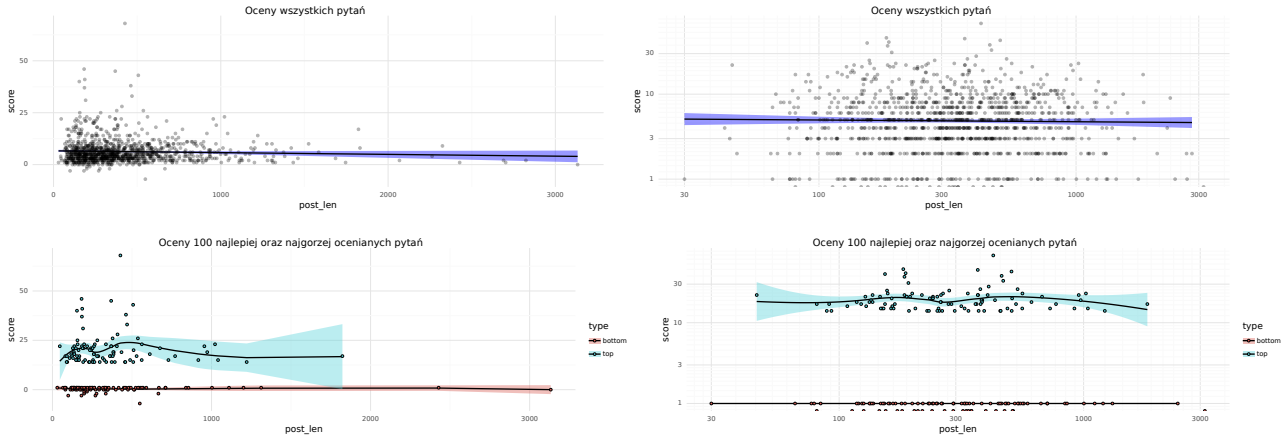
Zależność pomiędzy wartością `score` a długością wiadomości przedstawiono na Rysunek 3.6. Ze względu na obecność dużej ilości wartości odstających obie wartości przedstawiono również w skali logarymicznej.

Nie wykryto korelacji pomiędzy tymi dwoma zmiennymi. Współczynnik korelacji wyniósł -0.048 oraz -0.018 dla wartości zlogarytmowanych.

Postanowiono dodatkowo zbadać dwie podgrupy danych dla pytań najlepiej oraz najgorzej ocenianych. Wyodrębniono po 100 pytań z każdej z grup. Wyniki dla tych grup przestwationo na Rysunek 3.6 (dolny panel) oraz w Tabela 3.7 i Tabela 3.8.

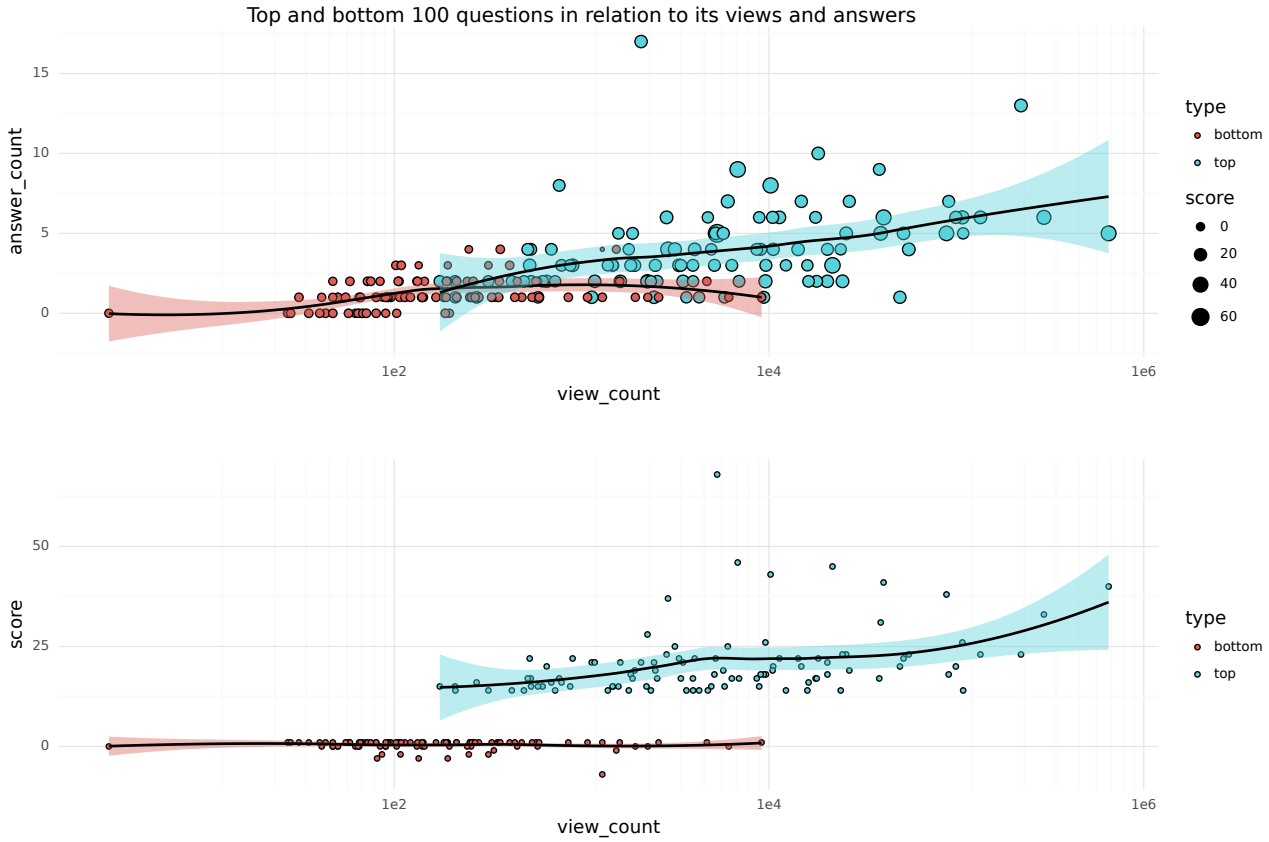
W podgrupie najlepiej ocenianych pytań średnia ocena wyniosła 20.6 +/- 8.5 a w podgrupie najgorzej ocenianych 0.4 +/- 1.2. Rozdzielność tych statystyk dla tych grup była oczekiwanym wynikiem.

Statystyki długości pytań nie odbiegają od siebie w znaczący sposób (średnie 349 +/- 275 najlepiej oceniane oraz 389 +/- 424 najgorzej oceniane) sugerując iż to długość pytania nie ma znaczącego wpływu na jego ocenę.



Rysunek 3.6: Oceny pytań na forum w zależności od długości zadanego pytania

Zauważono również że najlepiej oceniane pytania są również częściej oglądane i mają więcej odpowiedzi niż pytania najgorzej oceniane (Rysunek 3.7). Najlepiej oceniane pytania mają średnio 17 odpowiedzi podczas gdy najgorzej jedynie 4 (Tabela 3.9) oraz odpowiednio średnio 26289 i 475 wyświetleń (Tabela 3.10)



Rysunek 3.7: Oceny pytań na forum w zależności od ilości wyświetleń i odpowiedzi

Tabela 3.9: Statystyki ilości odpowiedzi na pytania z podziałem na podgrupy najlepszych i najgorszych pytań

	type	max(answer_count)	min(answer_count)	mean	std_dev	median
0	top	17	1	3.96	2.581676	3
1	bottom	4	0	1.28	0.964836	1

Tabela 3.10: Statystyki ilości wyświetleń pytań z podziałem na podgrupy najlepszych i najgorszych pytań

	type	max(view_count)	min(view_count)	mean	std_dev	median
0	top	648941	175	26047.65	76276.200191	4715
1	bottom	9124	3	495.56	1220.756292	141

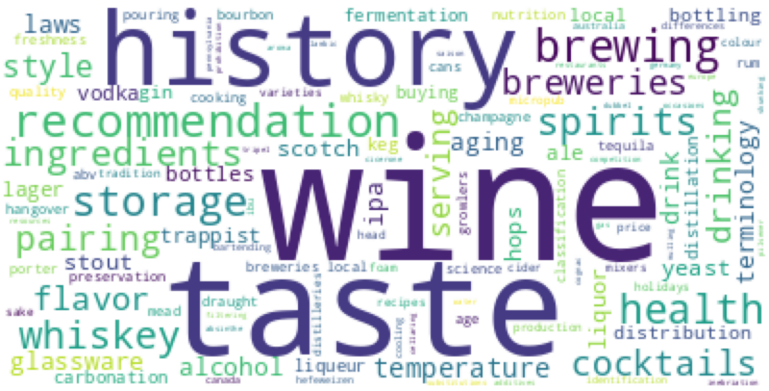
Tabela 3.11: Sumaryczne ilości wyświetleń wiadomości zawierających dany znacznik. Przedstawiono 10 najpopularniejszych znaczników.

	tag	sum_views
0	taste	1330670
1	health	1286001
2	preservation	682216
3	storage	542860
4	whiskey	464756
5	bourbon	330268
6	brewing	307892
7	ipa	291935
8	spirits	255328
9	drinking	225924

3.5 Analiza znaczników (tagów)

W celu identyfikacji tematów zapytań na forum przeanalizowano słowa kluczowe, którymi oznaczano pytania. Wyniki przedstawiono w postaci wykresu typu "Chmura słów" Rysunek 3.8. Okazuje się, że użytkownicy najczęściej pytali o `wina` (134, ang. *wine*), `smak` (120, ang. *taste*), `historię` (85, ang. *history*) i o `rekomendacje` (76, ang. *recommandation*).

Tę samą analizę wykonano na grupie 100 najlepiej i najgorzej ocenianych pytań badanych w Sekcja 3.4. W pierwszej grupie najczęściej używano znaczników `smak` (17) oraz `warzenie` (14, ang. *brewing*), w drugiej natomiast najczęściej pytano o `wina` (20), `rekomendacje` (15) i `zdrowie` (12, ang. *health*).



Rysunek 3.8: Chmura słów przedstawiająca najczęściej używane znaczniki na forum

Następnie sprawdzono, które ze znaczników nie tylko były najczęściej używane, lecz które generowały najwięcej wyświetleń. W tym celu zsumowano liczbę wyświetleń wszystkich zapytań na forum, w którym pojawił się dany znacznik (Tabela 3.11). Znaczniki `taste` oraz `health` generują najwięcej wyświetleń na tym forum w ilości 1349051 oraz 1316064, na dalszych miejscach znajdują się znaczniki `preservation` (686106), `storage` (558949) oraz `whiskey`(470841).

Zbadano również czy wykorzystanie najczęściej używanych znaczników zmieniało się w czasie istnienia forum. Z danych zaprezentowanych na Rysunek 3.9 można zaobserwować że niektóre znaczniki takie jak `spirits`, `bourbon` czy `whiskey` zaczęły być używane dopiero około roku 2016 - około 2 lata po

pierwszej wiadomości na forum. W danych zauważono również, że częstość używania znaczników jest skokowa. Może to być związane z ogólną małą ilością postów zamieszczanych na forum. Wyjątek stanowią najpopularniejsze znaczniki, takie jak `taste` czy `health`, których częstość występowania jest bardziej regularna.

Rysunek 3.9: Liczba postów w czasie dla każdego z najczęściej wyświetlanych znaczników. Wartości zagregowane na poziomie miesięcy. Górny panel przedstawia wartości miesięczne, dolny natomiast miesięczną sumę kumulatywną.

3.6 Analiza tytułów postów

Dodatkową metodą, oprócz analizy znaczkików, którą można wykorzystać w celu zbadania tematów poruszanych na forum może być analiza najczęściej pojawiających się słów. Można do tego wykorzystać treść tytułów wiadomości jak i ich główną treść. W niniejszej pracy skupiono się na analizie treści tytułów wiadomości, jako że bardzo często zawierają one dużo słów kluczowych.

Tytuły wiadomości są nieustrukturyzowanymi ciągami znaków, z tego powodu do ich analizy wykorzystano powszechnie używane narzędzia stosowane przy analizach NLP (ang. ***N**atural **L**anguage **P**rocessing*). Proces zastosowanej tutaj analizy tekstu składał się z 4 kroków:

1. Wyodrębnienia tokenów
2. Usunięcia tokenów o długości znaku 1

Tabela 3.12: Przykładowe rekordy danych wizualizujące proces przetwarzania tekstu. Kolejność chronologiczna kolumn title –> words_token –> words_no_stop –> words_stem

	0
title	What is a citra hop, and how does it differ from other hops?
words_token	[what, is, citra, hop, and, how, does, it, differ, from, other, hops]
words_no_stop	[citra, hop, differ, hops]
words_stem	[citra, hop, differ, hop]

Tabela 3.13: Zliczenia tokenów słów najczęściej pojawiających się w tytułach postów (15 najliczniejszych)

	words	count
0	beer	476
1	wine	147
2	drink	104
3	alcohol	88
4	differ	72
5	bottl	68
6	use	50
7	tast	47
8	brew	43
9	make	41
10	good	33
11	cocktail	29
12	age	27
13	ale	26
14	recommend	26

3. Usunięcia tokenów nie niosących informacji o treści (ang. *stop words*)
4. Ujednolicenia różych form danego wyrazu poprzez zabieg stemming'u (ang. *stemming*), mającego na celu ucinanie końcówek wyrazów (tokenów) i pozostawieniu jedynie jego wartości bazowej.

Przykładowa tabela wizualizująca tokeny surowe w porównaniu do tokenów po odfiltrowaniu słów typu `stop words` oraz tokenów krótszych niż 1 zaprezentowano na Tabela 3.12

Dzięki zabiegowi stemmingu otrzymywany jest bardziej jednorodny zestaw danych. Liczba unikalnych tokenów została zredukowana z 2055 do 1680.

Następnie zliczono, które tokeny pojawiają się w największej ilości tytułów i zaprezentowano w Tabela 3.13. Najczęściej używanymi tokenami były tokeny `beer` (476), `wine` (152) oraz `drink` (105). Wyniki pozostają w zgodzie z wnioskami uzyskanymi poprzez analizę znaczników.

Bibliografia

4 Załączniki

4.1 Polecenia budujące infrastrukturę

4.1.1 EMR

```
aws emr create-cluster --name="MyEMRCluster" \
    --release-label emr-6.8.0 \
    --applications Name=JupyterHub Name=Hadoop Name=Spark \
    --log-uri s3://emr-logs-beer-and-wine/MyJupyterClusterLogs \
    --use-default-roles \
    --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.large InstanceGroupTyp
    --ebs-root-volume-size 32 \
    --configurations file://emr-configurations.json \
    --bootstrap-actions Path=s3://misc-beer-and-wine/install_python_libraries.sh,Name=InstallJupyterL
```

Plik `install-my-jupyter-libraries.sh` dostępny jest pod poniższym adresem

4.1.2 S3

Koszyki S3 zostały utworzone przy pomocy poniższego polecenia:

```
aws s3api create-bucket --acl private --bucket <nazwa koszka>
```

4.2 Definicje funkcji

4.2.1 tags_remove()

```
from pyspark.sql.functions import udf
from bs4 import BeautifulSoup
from html import unescape

def tags_remove(s):
    if s is not None:
        soup = BeautifulSoup(unescape(s), 'lxml')
        return soup.text
    else:
        return None
udf_tags_remove = udf(lambda m: tags_remove(m))
```

4.2.2 regexp_extract_all()

```
from pyspark.sql import DataFrame
from pyspark.sql import functions as f


def regexp_extract_all(
    df: DataFrame,
    regex: str,
    no_of_extracts: int,
    input_column_name: str,
    output_column_name: str = "output",
    empty_array_replace: bool = True,
):
    """Pyspark implementation for extracting all matches of a reg_exp_extract

    Background
    ----------
    The regular implementation of regexp_extract (as part of pyspark.sql.functions module)
    is not capable of returning more than 1 match on a regexp string at a time. This
    function can be used to circumvent this limitation.

    How it works
    ------------
    You can specify a `no_of_extracts` which will essentially run the regexp_extract
    function that number of times on the `input_column` of the `df` (`DataFrame`).
    In between extracts, a set of interim columns are created where every
    intermediate match is stored. A distinct array is created from these matches,
    after which the interim columns are dropped. The resulting array is stored in
    the defined `output_column`. Empty strings/values in the resulting array can
    optionally be dropped or kept depending on how `empty_array_replace` is set
    (default is True).

    Usage example
    -------------
    In the below example, we are extracting all email-addresses from a body of text.
    The returned DataFrame will have a new ArrayType column added named `email_addresses`

    > # Assuming `df` is a valid DataFrame containing a column named `text`
    > email_regex = r"[\w.-]+@[\w.-]+\.[a-zA-Z]{1,}"
    > df = regexp_extract_all(df, email_regex, 6, "text", "email_addresses", True)

    Parameters
    ----------
    df: DataFrame
        Input DataFrame

    regex: str
        Regexp string to extract from input DataFrame
```

```
    no_of_extracts: int
        Max number of occurrences to extract

    input_column_name: str
        Name of the input column

    output_column_name: str
        Name of the output column (default: output)

    empty_array_replace: bool
        If set to True, will replace empty arrays with null values (default: True)
    """
    repeats = range(0, no_of_extracts)

    # A set of interim columns are created that will be dropped afterwards
    match_columns = [f"___{r}___" for r in repeats]

    # Apply regexp_extract an r number of times
    for r in repeats:
        df = df.withColumn(
            match_columns[r],
            f.regexp_extract(
                f.col(input_column_name),
                # the input regex string is amended with ".*?"
                # and repeated an r number of times
                # r needs to be +1 as matching groups are 1-indexed
                "".join([f"{regex}.*?" for i in range(0, r + 1)]),
                r + 1,
            ),
        )

    # Create a distinct array with all empty strings removed
    df = df.withColumn(
        output_column_name,
        f.array_remove(f.array_distinct(f.array(match_columns)), ""),
    )

    # Replace empty string with None if empty_array_replace was set
    if empty_array_replace:
        df = df.withColumn(
            output_column_name,
            f.when(f.size(output_column_name) == 0, f.lit(None)).otherwise(
                f.col(output_column_name)
            ),
        )

    # Drop interim columns
    for c in match_columns:
```

```
        df = df.drop(c)

    return df
```

4.3 Repozytorium kodu wykorzystany w pracy

https://github.com/michkam89/big-data-pw-project

Bibliografia

[1] *Apache Hadoop*. URL: https://hadoop.apache.org/.

[2] Kevin Bartley. "Data Statistics - how much data is there in the world?" W: *Rivery* (list. 2022). URL: https://rivery.io/blog/big-data-statistics-how-much-data-is-there-in-the-world/.

[3] Jeffrey Dean i Sanjay Ghhemawat. "System and method for efficient large-scale data processing". Sty. 2010.