**Project Cover Sheet**

---

Student IDs : (We use one table to display these information)
Student Names:
Student Emails :

| Student name | SID | Mail |
|---|---|---|
| CHAN Kiu Nga, Sandy | 56219011 | kiunchan3-c@my.cityu.edu.hk |
| CHAN Tsz Ngai, Matthew | 56213172 | tnchan27-c@my.cityu.edu.hk |
| Ma Wing Kin, Paul | 56243594 | wingkinma2-c@my.cityu.edu.hk |
| NG Sin Yung, Cindy | 56213971 | sinyungng4-c@my.cityu.edu.hk |

---

**Title:**
The sentiment and need analysis of type 1 and type 2 diabetes

**Highlights: (maximum: 5 items)**

- Data extraction
- Data exploration (Basic activity analysis and LDA topic classification also included.)
- Tokenization (Involves parallel computing.)
- Data visualization (TFIDF, Sentiment analysis, and word cloud)
- Comment-based analysis (Examined some comments based on data visualization's result to give insights.)

---

**List of Deliverables**

| File Name | Description |
|---|---|
| Project report group 4.pdf | This is the main project report of our group (CS4480 group 4). |
| CS4480_gp4_codes | This is a folder put in the link (https://drive.google.com/drive/folders/1MIn7S9vYG6M6Wr702CvKkWIuJGrbERPw?usp=share_link) because the size is around 2GB, which is too difficult to directly submit in canvas although all folders inside are zipped already.The csv files (dataset) are also included to ensure the codes are executable, thanks. |
| CS4480_gp4_sentimentAnalysis_commentAnalysis | The comment analysis folders, can be found in the link (https://drive.google.com/drive/folders/1QeovPGJmfJhaFpCHP-xy-vBNT1MxLUMs?usp=share_link ) |

------ END ------

# CS4480 project report
# Project group 4

# The sentiment and need analysis of type 1 and type 2 diabetes


Group members

| Student name | SID | Mail |
|---|---|---|
| CHAN Kiu Nga, Sandy | 56219011 | kiunchan3-c@my.cityu.edu.hk |
| CHAN Tsz Ngai, Matthew | 56213172 | tnchan27-c@my.cityu.edu.hk |
| Ma Wing Kin, Paul | 56243594 | wingkinma2-c@my.cityu.edu.hk |
| NG Sin Yung, Cindy | 56213971 | sinyungng4-c@my.cityu.edu.hk |

Course instructor

Ka Chun WONG

# Abstract

In this report, the needs and sentiment analysis of type1 diabetes and type2 diabetes are discussed using two diabetes discussion forums (Diabetes Daily Forum and Diabetes.co.uk) so that the diabetics understand more about their disease and the public can offer help properly. We concluded that the sentiment of the type2 diabetic patients are relatively more positive than the type1 diabetic patients as they usually suffer less severe diabetes symptoms and emotional disorder, meanwhile, the type1 diabetic patients also need more assistance from their surroundings given the fact that they need more external support. The conclusion is based on the analysis from the result of 3 methods that help explore the textual data, which are TFIDF, wordcloud and sentiment analysis, and also the interpretation of LDA topic classification and sampling statistics.

# Table of content

# *I. Introduction*

Diabetes is a well-known chronic disease in the world, divided into 3 types, type 1, type 2, and type 1.5 (LADA (Latent autoimmune diabetes in adults)). Type 1 is caused by an autoimmune reaction. This reaction destroys the cells in the pancreas that make insulin (Centers for Disease Control and Prevention,2022). Type 1.5 is the type that is between type 1 and type 2. Type 2 is the most common diabetes type. It is caused by insufficient insulin production in the pancreas.

Diabetes patients need to face long-term treatment, such as injecting insulin and taking blood-sugar level-controlling medicine. Also, many of them need to poke their fingers to measure their blood-sugar level every day, maintain their less sugar eating habit, and do sports regularly for self-blood glucose monitoring. The plenty of routines make some diabetes patients feel their life is hard.

However, diabetes has become more common in the world nowadays. According to the World Health Organization, 422 million people have diabetes in the world, and most of them are living in low-income and middle-income countries. Also, there are 1.5 million deaths directly attributed to diabetes every year. The number of diabetes cases and diabetes prevalence has increased steadily over the past few decades. Therefore, it is predicted that more people will suffer from diabetes.

# *II. Motivation*

Though diabetes is a chronic disease, there are some complications (long-term problems) that can be caused by it. Such as eye problems (Diabetic retinopathy, which is a complication that can affect their eyesight), foot problems, and kidney problems that are related to nephropathy (Complications of diabetes, n.d.). These complications enhance the severity of diabetes.

Besides, diabetes is not a disease that is only related to people who are over 50 and above since there are more teens, children, and young adults suffering from diabetes nowadays. According to BBC News (2015), a 3-year-old was diagnosed with type 2 diabetes due to an unhealthy diet and lack of exercise. Moreover, fast-food and other unhealthy foods are too easy to obtain, therefore, it is predicted that the number of diabetes patients (especially type 2) will increase more rapidly.

Due to the increase in the diabetes patient population, maybe one of our friends, families, or relatives can be diagnosed with diabetes. So, if more people understand the diabetes patients' needs and sentiments, we can provide better care and help for them. Not only it helps diabetes patients have a better life, but it also creates more happiness and love in societies, even the world. Hence, the main targets of this research are investigating and analyzing the sentiment and need(s) of diabetes patients.

# *III. Objective*

The two main objectives of this research :
(Mainly focus on diabetes type 1 and diabetes type 2, the reason for the main focus is going to be explained in the **Data exploration ->Activity of each diabetes type** part )

1. Compare the sentiment difference between diabetes types
   By conducting the sentiment analysis and sentiment comparison among each diabetes type, the diabetes group with the worst sentiment is found. Hence, more help and care can be provided to the group with the worst sentiment.

2. Analyze the need of each diabetes type
   The cause of diabetes is different for each type, so analyzing their needs can find what help and care are needed by diabetes patients to enable them to achieve a better life.

Assumption: All comments (texts) from diabetes forums are related to diabetes.

# *IV. Data sources*

Although some diabetes patients make a diary about their blood sugar level, even their conditions every day, these confidential data are not easy to obtain from websites without any permission. Therefore, we chose 2 online diabetes forums (Diabetes Daily Forum and Diabetes.co.uk) to web scrape the information.

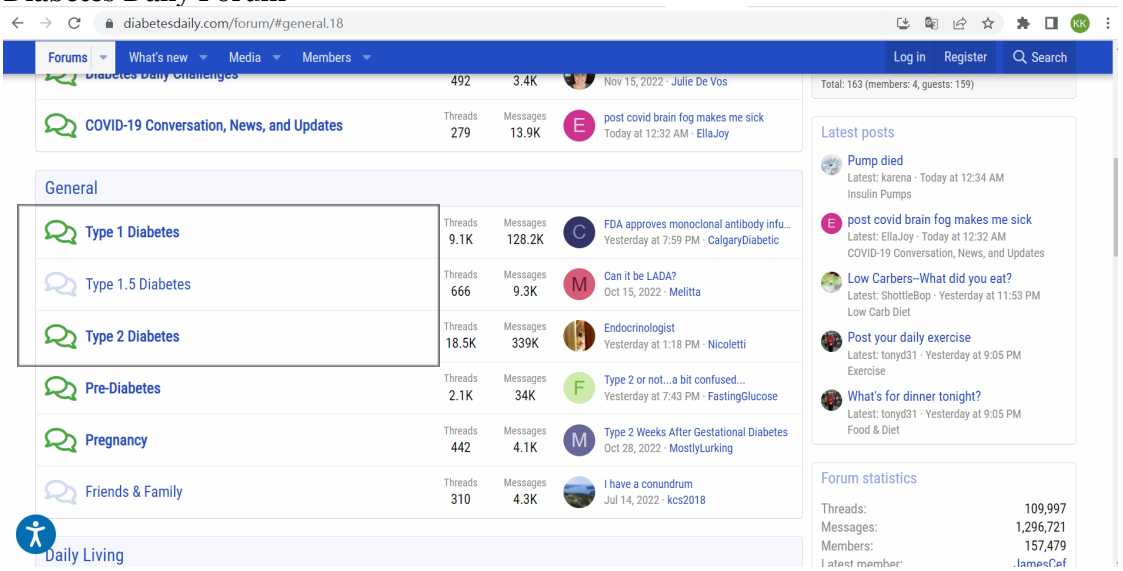
## a. Introduction of the data sources

The advantages of the 2 selected diabetes forums:

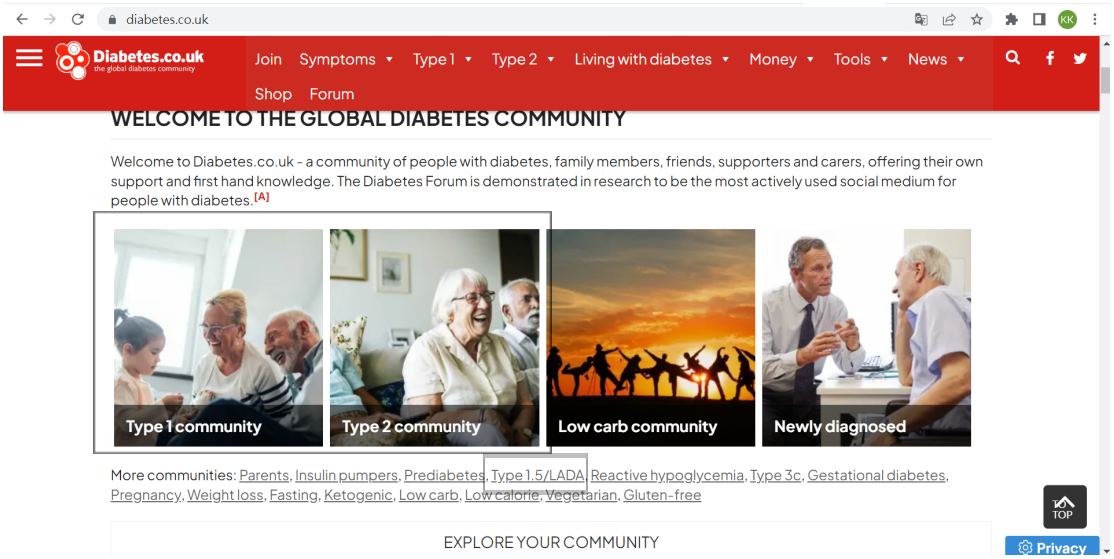
- Accessibility
  No extra permission is required for data extraction since the contents are available to the public.
- Clear discussion groupings
  Each discussion forum is divided into different diabetes types, so it helps the comments' classification.

| Websites | Type 1 forum | Type 1.5 forum | Type 2 forum |
|---|---|---|---|
| Diabetes Daily Forum | ✓ | ✓ | ✓ |
| Diabetes.co.uk | ✓ | ✓ | ✓ |

Diabetes Daily Forum




Diabetes.co.uk




The black-sided rectangles highlighting the diabetes forums can be found on these two websites. They include all general diabetes types' discussion forums, hence, the activity of each diabetes type can be obtained and fairly compared on each website.

## b. Data extraction process

### Data extraction procedure

Here is the two website's data extraction processes' description. This is written together because comparing the two websites, it was found the structure of the two websites is very similar. (This example is taken from Diabetes Daily Forum diabetes type 1).

First, the packages (requests, BeautifulSoup) were imported for data extraction. Pandas and time were used for storing the data in CSV format and calculating the code's running time respectively.

In [3]:
```
#import all the required packages
#web scrapping
import requests
from bs4 import BeautifulSoup as bs

#serve as complement`
import time
import pandas as pd
```

Second, the function, *get_NumberOfPages (url)* was defined for extracting the maximum possible number of pages in a forum or discussion topic.

In [4]:
```
def get_NumberOfPages (url):
    #use requests.get to get the content
    req = requests.get(url)
    #use BeautifulSoup of the text
    soup = bs(req.content, 'html.parser')
    #find the hyperlinks in <ul> ("class" : "pageNav-main" })
    #navs is a list
    navs = soup.find("ul", { "class" : "pageNav-main" })
    #num_of_pages is 1 when navs cannot find anything (no additional pages)
    page_length=1
    if (navs):   #if the navs exist, which means there are one more pages
        navs=navs.find_all("li", recursive=False)
        #navs is a list that show how many page buttons in the same page, since the website will
        #show first several pages , then the number of last page,
        #so if we retrieve the last element in navs and convert it to text then int
        #then we can know how many number of pages need to scrape
        page_length=int(navs[len(navs)-1].text)

    return page_length
```

In [5]:
```
target_forum_url="https://www.diabetesdaily.com/forum/forums/type-1-diabetes.9/"
total_pages=get_NumberOfPages (target_forum_url)
print("Total number of pages in disscussion forum:",total_pages)
```

```
Total number of pages in disscussion forum: 915
```

Third, the function, *getData (page_number)* was defined for extracting the data (each discussion topic, number of texts per topic, and each text), therefore, three empty lists were defined to store the data. (Text refers to comment.)

In [6]:
```
#Build up the lists to hold the data

discussion_topics=[]
NoOfTexts_discussion_topics=[]
Texts=[]
```

## URL modification

There were some discoveries. For example, the page 1 discussion forum's URLs “https://www.diabetesdaily.com/forum/forums/type-1-diabetes.9/” and “https://www.diabetesdaily.com/forum/forums/type-1-diabetes.9/page-1” are on the same page, therefore, for the discussion forum’s page 1, the URL with a page tag and the URL without a page tag have the same meaning.

Similarly, for each discussion topic (Take "Had a Friend with type one" as an example) “https://www.diabetesdaily.com/forum/threads/had-a-friend-with-type-one.136015/” and “https://www.diabetesdaily.com/forum/threads/had-a-friend-with-type-one.136015/page-1 ” are also on the same page. Therefore for the discussion topic’s page 1, the URL with a page tag and the URL without a page tag have the same meaning.

If there is an URL (“https://www.diabetesdaily.com/forum/forums/type-1-diabetes.9/”), we can merge the URL by (suppose page_number is an integer)
url="https://www.diabetesdaily.com/forum/forums/type-1-diabetes.9/"
url=url+"page-"+str(page_number)
then new URL
“https://www.diabetesdaily.com/forum/forums/type-1-diabetes.9/page-1” forms.

Hence, the user-defined function’s parameter refers to each page of the discussion forum. When the forum’s page number is plugged into the function’s parameter, it can first obtain all discussion links on the same page.

## Content extraction and blockquote removal

For each discussion topic’s link, the function will first find the discussion forum title on the first discussion page, it is h1 (header 1) and its class is “p-title-value”. Since only one header is needed for the record, using the find function in BeautifulSoup is good enough.

Then, there can be more than one page for more popular discussion topics, its maximum page number can be obtained by applying *get_NumberOfPages (url)* to get the number of pages included in the discussion topic.

Next, we got the content by plugging the number of pages from the for loop list that starts from 1 and ends at the number of pages obtained from *get_NumberOfPages (url)*. In each for loop, we formed a new link by adding the discussion forum’s url, “page-” and the string of the number of pages to extract all the comments by flipping the web page in for loop, then added the contents into list topic_list_per_page with the BeautifulSoup’s “find_all” function and “.text” function.

While flipping the page in each discussion topic, the number of texts that can be found on the page is also added to number_of_diaglogues_per_topic for further analysis back up. If the variable number_of_diaglogues_per_topic equals 0, we can notice an error during text extraction. As there should be at least one text (the text posted by the poster) in the discussion topic without a response. Also, it could check whether the number of text extracted matches the number of texts displayed on the website, if not, there may be some extraction error, and we can consider skipping the post or extracting again. This case rarely occurred during long-time extraction.

Most contents are found in the “div” block whose class is “bbWrapper”. However, the users would reply to the previous content, and many of them have a forward message that causes dependency. Thus, the codes to solve this issue are necessary.

For each comment found in each discussion topic, if the blockquote, which is in div block and the class is “bbCodeBlock-content” can be found, then find the blockquote and decompose it to make it disappear. If the blockquote is not found, continue and check for the next content.

The codes for dealing with the blockquotes:

Diabetes Daily Forums

```
    for element in topic_list_content_per_page:
        remove=element.find("div",{"class":"bbCodeBlock-content"})
        if (remove!=None):
            element.find("blockquote").decompose()
        else:
            continue
```

Diabetes.co.uk

```
    for element in topic_list_content_per_page:
        for tag in ['blockquote']:
            if (element.find(tag)):
                element.find(tag).decompose()
```

After the blockquote's elimination, we added the text to the topic_list_content_per_page list and extended the text in the topic_list_content_per_page list to topic_content.

Next, these records are included in the three global lists (discussion_topics, NoOfTexts_discussion_topics, Texts) accordingly. Repeat the process in the third stage until all pages' texts have been extracted.

```
#Prevent none type cannot be decomposed bug
#Define a big function to do all the hold the data
def getData (page_number):
    #declare the global lists into function
    global discussion_topics
    global NoOfTexts_discussion_topics
    global Texts

    topic_diag=[] #the number of dialogues of each topic

    forum_url="https://www.diabetesdaily.com/forum/forums/type-1-diabetes.9/page-"+str(page_number)

    #extract all the discussion fourm links from each page
    source_html = requests.get(forum_url)

    topic_pagelinks = [
    f'https://www.diabetesdaily.com{a["href"]}' for a
    in bs(source_html.content, "html.parser").select(".structItem-title a")]
    #for j in topic_pagelinks:print(type(j))
    for link in topic_pagelinks:

        #form the information
        topic_url=link
        topic_req=requests.get(link)
        topic_sp=bs(topic_req.text,'lxml')

        #first we extract the TITLE
        topic_title=topic_sp.find("h1",{"class":"p-title-value"}).text
        discussion_topics.append(topic_title)
        #print(topic_title)

        #In each discussion topic, we may have more than one page

        #so the first step is, we know how many pages are there
        topic_pages_no=get_NumberOfPages(link)
        #print(topic_pages_no)

        topic_content=[] #the contents (text) in each topic
        number_of_diaglogues_per_topic=0
        #use for loop to extract ALL content also the number of dialogues
        for page_num in list(range(1,topic_pages_no+1)):
                topic_page_url=topic_url
                topic_page_url=topic_page_url+"page-"+str(page_num)
                #print(topic_page_url)
                topic_page_r=requests.get(topic_page_url)
                topic_page_sp=bs(topic_page_r.text,'lxml')
                topic_list_content_per_page=topic_page_sp.find_all("div",{"class":"bbWrapper"})
                for element in topic_list_content_per_page:
                    remove=element.find("div",{"class":"bbCodeBlock-content"})
                    if (remove!=None):
                        element.find("blockquote").decompose()
                    else:
                        continue

                topic_list_content_per_page=[i.text for i in topic_list_content_per_page]
                topic_content.extend(topic_list_content_per_page)
        #print(len(topic_content))
        number_of_diaglogues_per_topic+=len(topic_content)


        #after extracted all topic's all content , append it into the global list:
        #why append ? because we can explode the list inside topic's content by each disscussion forum's topic after the dataframe is formed
        Texts.append(topic_content)

        #add the number of dialogues into the global list
        NoOfTexts_discussion_topics.append(number_of_diaglogues_per_topic)
```

Fourth, the for loop and *getData (page_number)* function is used in normal computing extraction. While extracting information, we needed to pay attention to the code when the attribute error happens. This error is related to decompose text error in long-time extraction. (Note: In the photo, it is “normal computing”, not “parallel computing”.)

```
In [10]: start=time.time()
         for i in list(range(1,total_pages+1)):
             getData(i)
         end=time.time()
         exe_time=end-start
         print("Parallel computing extraction total processing time: ",exe_time," s")
```

```
      1 start=time.time()
      2 for i in list(range(1,total_pages+1)):
----> 3     getData(i)
      4 end=time.time()
      5 exe_time=end-start

Input In [7], in getData(page_number)
     25 topic_sp=bs(topic_req.text,'lxml')
     27 #first we extract the TITLE
---> 28 topic_title=topic_sp.find("h1",{"class":"p-title-value"}).text
     29 discussion_topics.append(topic_title)
     30 #print(topic_title)
     31
     32 #In each discussion topic, we may have more than one page
     33
     34 #so the first step is, we know how many pages are there

AttributeError: 'NoneType' object has no attribute 'text'
```

When there is a decompose error, a small function, *getData_topic (specific_url)* that processes each URL manually is created to add the discussion_topic one by one. Therefore,

the discussion forums' topics and text(s) can be added one by one by this function and simple list append technique. After extracting the remaining data for a post with an attribute error, the *getData (page_number)* function can be used to scrape the data for the remaining pages.

```
#stop at p.343 'Humalog v Apidra', so next I would start the loop at p.344
#p.343 remaining topics -> extract MANNUALLY

#I defined a per topic function to add to the global variable if failed
#'Test Results And What I Was Told' #NoOfTexts_discussion_topics need to add ourselves
def getData_topic(specific_url):
        global discussion_topics
        global Texts

        specific_get=requests.get(specific_url)
        specific_sp=bs(specific_get.text,'lxml')
        specific_number_of_texts=0
        specific_number_of_forum_pages=get_NumberOfPages (specific_url)

        #get the title
        specific_topic_title=specific_sp.find("h1",{"class":"p-title-value"}).text
        discussion_topics.append(specific_topic_title)

        specific_topic_content=[]

        for page_num in list(range(1,specific_number_of_forum_pages+1)):
                specific_topic_page_url=specific_url
                specific_topic_page_url=specific_topic_page_url+"page-"+str(page_num)
                #print(topic_page_url)
                specific_topic_page_r=requests.get(specific_topic_page_url)
                specific_topic_page_sp=bs(specific_topic_page_r.text,'lxml')
                specific_topic_list_content_per_page=specific_topic_page_sp.find_all("div",{"class":"bbWrapper"})
                for element in specific_topic_list_content_per_page:
                    specific_remove=element.find("div",{"class":"bbCodeBlock-content"})
                    if (specific_remove!=None):
                        element.find("blockquote").decompose()
                    else :continue
                specific_topic_list_content_per_page=[i.text for i in specific_topic_list_content_per_page]
                specific_topic_content.extend(specific_topic_list_content_per_page)

        Texts.append(specific_topic_content)
```

There are some self-checking codes for monitoring the process. They are extremely useful when an extraction error happens halfway. We can also ensure the order of data is as accurate as possible and the number of records is matched among 3 lists.

**Self-checking for last elements , put it at back**

```
In [73]: #the length of lists
         #don't use [] again! CRASHED MY MIND!!
         print("Length of discussion_topics: ",len(discussion_topics))
         print("------------------------------------------------------------")
         print("Length of NoOfTexts_discussion_topics: ",len(NoOfTexts_discussion_topics))
         print("------------------------------------------------------------")
         print("Length ofTexts: ",len(Texts))
         print("------------------------------------------------------------")

Length of discussion_topics:  9132
------------------------------------------------------------
Length of NoOfTexts_discussion_topics:  9132
------------------------------------------------------------
Length ofTexts:  9132
------------------------------------------------------------
```

```
In [83]: #the last element of list

         print("Latest discussion topics ",(discussion_topics)[-1])
         print("------------------------------------------------------------")
         print("Length of NoOfTexts_discussion_topics: ",(NoOfTexts_discussion_topics)[-1])
         print("------------------------------------------------------------")
         print("Texts in latest topic: ",((Texts)[-1][NoOfTexts_discussion_topics[-1]-1]))
         print("------------------------------------------------------------")
         print("Length of Texts in latest topic: ",len((Texts)[-1]))
         print("------------------------------------------------------------")

Latest discussion topics  Taking Advantage of Diabetics
------------------------------------------------------------
Length of NoOfTexts_discussion_topics:  4
------------------------------------------------------------
Texts in latest topic:  http://msnbc.msn.com/id/10777506/
------------------------------------------------------------
Length of Texts in latest topic:  4
------------------------------------------------------------
```

At last, the result can be added to the data frame. First, the number of texts per topic is aggregated in the first CSV file. Then the second CSV file contains all text and its relevant topic.

For the second CSV file, note that all the text is added per topic, so, the data looks like

| | topic | text |
|---|---|---|
| 0 | topic 1 | [text1,text2,text3] |

But after using pd.explode for the text column, the data frame becomes

| | topic | text |
|---|---|---|
| 0 | topic 1 | text1 |
| 0 | topic 1 | text2 |
| 0 | topic 1 | text3 |

Then each comment can be obtained. Finally, export it to a CSV file.

These are examples of Pandas processing.

In [85]:
```
# group extraction into the dataset
data1=pd.DataFrame(columns=["topic","number of text"])
data1["topic"]=discussion_topics
data1["number of text"]=NoOfTexts_discussion_topics
data1
```

Out[85]:

| | topic | number of text |
|---|---|---|
| 0 | Learning Center - Type 1 Diabetes | 6 |
| 1 | Had a Friend with type one | 65 |
| 2 | Seeking Diary Study Participants (children wit... | 1 |
| 3 | An insulin users joke | 2 |
| 4 | t:connect training quiz | 3 |
| ... | ... | ... |
| 9127 | have you got a spotty head | 1 |
| 9128 | Another Question on diabetes? | 2 |

In [86]:
```
data2=pd.DataFrame(columns=["topic","text"])
data2["topic"]=discussion_topics
data2["text"]=Texts
data2=data2.explode("text") #explode the list in text -> discrete data
data2
```

Out[86]:

| | topic | text |
|---|---|---|
| 0 | Learning Center - Type 1 Diabetes | For up to date information about Type 1 Diabet... |
| 0 | Learning Center - Type 1 Diabetes | I have a question about insulin antibody testi... |
| 0 | Learning Center - Type 1 Diabetes | In general, a reference range for any given te... |
| 0 | Learning Center - Type 1 Diabetes | Thank you for the reply. I think you must be ... |
| 0 | Learning Center - Type 1 Diabetes | Medtronics insulin pump with MiniMed Quick -se... |
| ... | ... | ... |

```
#export both of them to csv
data1.to_csv("DiabetesDailyForum_Type1_CommentCountPerTopic.csv")

data2.to_csv("DiabetesDailyForum_Type1_CommentPerTopic.csv")
```

## Data extraction time

These are the data extraction time approximations. They are references only since the extraction time also depends on the wifi's speed.
(Ignored the time of manually adding the discussion forums.)

| Computer | Wifi | Website | Diabetes type | Number of comments extracted | Approximate time |
|---|---|---|---|---|---|
| Lenovo ideapad Flex 5 | 5G | Diabetes Daily Forum | Type 1 | 128189 | 10.66 hours |
| Lenovo ideapad 330S | 5G | Diabetes Daily Forum | Type 1.5 | 9279 | 0.34 hours |
| Lenovo ideapad 330S | 5G | Diabetes Daily Forum | Type 2 | 339813 | 11 .7622 hours |
| MSI | 5G | Diabetes.co. uk | Type 1 | 222690 | 12 hours |
| Lenovo ideapad Flex 5 | 5G | Diabetes.co. uk | Type 1.5 | 9075 | 0.59 hour |
| Lenovo ideapad Flex 5 | 5G | Diabetes.co. uk | Type 2 | 300226 | 15.18 hours |

*If there is an attribute error, the time of code can not be displayed, so it is written as “approximate time”

## The reason for no parallel computing in the data extraction process

In python, the Joblib package is a popular option for parallel computing. It is because the number of threads (CPUs) executing the task can be adjusted by users. Besides, the code is relatively easy to use and the execution time is usually faster than normal computing. The function, "multiprocessing.cpu_count()" was applied to find how many CPUs can be found in computers to decide the value of n_jobs.

Unluckily, the drawback of Joblib is it is hard to act on the global lists. Thus, specifying “require=”sharedmem”” and map(delayed(getData), list(range(1.4)) (mapping for function and specifying the range meanwhile) is necessary for adding the data to global lists.

These are the codes for parallel computing.

In [33]:
```
#parallel computing
import multiprocessing
from joblib import Parallel , delayed

#for displaying photo
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

In [34]:
```
#check how many CPU in my python
#enviorment : BYO desktop (Lenovo ideapad Flex 5 and 5G wifi)
print("Number of CPUs in my computer: ",multiprocessing.cpu_count())
```

```
Number of CPUs in my computer:  16
```

In [35]:
```
#I use 16 CPUs for parallel computing
#For fair test, I still use first 3 pages (sames as the normal addition one)
discussion_topics=[]
NoOfTexts_discussion_topics=[]
Texts=[]

start=time.time()
#Parallel computing, sharedmem means share memory, so the contents can be added into the global list
Parallel(n_jobs=16, require='sharedmem')(map(delayed(getData), list(range(1,4))))
end=time.time()
exe_time2=end-start
print("Parallel computing extraction total processing time: ",exe_time2," s")
```

```
Parallel computing extraction total processing time:  34.264023780822754  s
```

Here is the comparison between normal computing and parallel computing.
(To ensure a fair comparison, p.1-3 of Diabetes Daily Forum‘s type 1 ‘s discussion forum is chosen.)

Out[92]:

| | Normal computing | Parallel computing (n_jobs=16) |
|---|---|---|
| **Time (second)** | 67.624082 | 34.264024 |
| **Data Accuracy** | Accurate | Non-accurate |

Although parallel computing (n_jobs=16) is much faster than normal computing, it counts only 50.7% of normal computing time, we would not use it because there is a dependency between the three global data-collecting lists, which means we need the topic, the number of texts in each discussion topic, and the topic's text(s) to be matched with each other to put all the data into the data frame. Besides, how the task is distributed among CPUs in parallel computing becomes an uncertainty. It might be a reason why the order of elements shuffles.

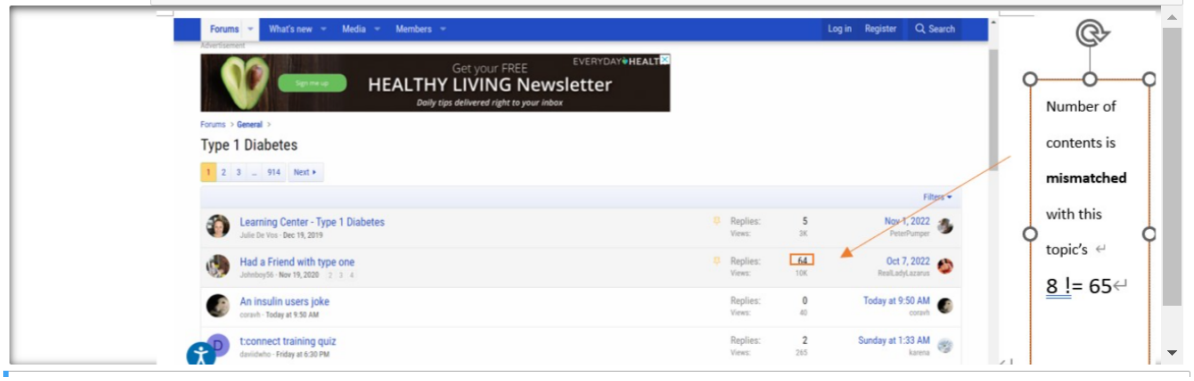
Verification: examining the order of lists in parallel computing (Example: the 6th element in 3 lists)

These are the data found in the 6th element of 3 lists.

```
In [83]: discussion_topics[5]
Out[83]: 'Had a Friend with type one'

In [84]: NoOfTexts_discussion_topics[5]
Out[84]: 8

In [85]: #The first text of 'Had a Friend with type one'
         Texts[5][0]
Out[85]: ''
```

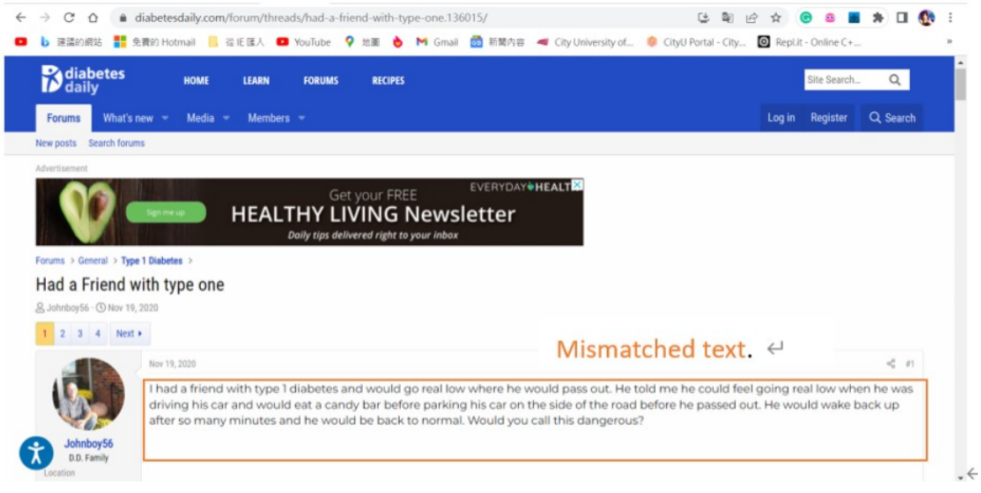
However, the number of texts in this discussion topic must be 65 , not really 8.

```
In [93]: #Check is the result correct
         #By looking this photo, it shows the number of texts is not correct already because the order of elements in list makes up.
         from IPython.display import display, Image
         display(Image(filename='Why_NO_parallel.jpg'))
```




Moreover, the first text has a lot of words, and does not remain empty.

```
First text of 'Had a Friend with type one':

However, the real first text in 'Had a Friend with type one' is...
```




These errors tell the lists' order can be random in parallel computing but not in normal computing. Therefore, parallel computing is not preferred since these computer errors increase the difficulty in analysis and the probability of shuffling the record's order.

# V. *Data processing*

## Data processing introduction

This project uses Spark, a data processing framework that integrates parallel computing. Spark and Hadoop are the two state-of-the art solutions for processing large scale data. This project chooses Spark over Hadoop because Spark provides a more user-friendly programming interface to use compared with the Hadoop Mapreduce. Moreover, spark utilizes in-memory computing, while Hadoop reads and writes from disk. It makes spark a lot faster than Hadoop. Although Hadoop uses less resources than Spark and it suits better in a multi-user environment. Yet the advantages of using Hadoop cannot outweigh the advantages of using Spark.

## Spark environment setup

This project uses PySpark as the spark interface. PySpark allows developers to write Spark programs using Python APIs. PySpark supports most features from Apache , such as Spark Dataframe and Spark Core. For setting up a PySpark environment, this project uses the latest docker image from Jupyter Docker Stacks (jupyter/pyspark-notebook). It provides an Apache Spark environment with Hadoop binaries.

## Docker environment setup

This project uses docker compose for configuring the docker environment, it ensures the Spark environment is set up correctly, and it allows different teammates to work in the same environment, without having to worry about debugging in different environments. Below are the docker-compose configuration file, and the dockerfile for installing extra packages inside the docker container.

```yaml
version: "3.7"

services:
  # jupyterlab with pyspark
  pyspark:
    build:
      context: ./
    environment:
      JUPYTER_ENABLE_LAB: "yes"
      NotebookApp.token: ""
      PYTHONPATH: "/usr/local/spark/python/lib/py4j-0.10.9.5-src.zip:/usr/local/spark/python:"
    ports:
      - "8888:8888"
    volumes:
      - ./data:/home/jovyan/work
    command: "start-notebook.sh --NotebookApp.token='' --NotebookApp.password='' --ServerApp.disable_check_xsrf=True"
```

```dockerfile
FROM jupyter/pyspark-notebook:latest

RUN pip install nltk
```

This project also uses vscode dev containers for development, which allows developers to develop their programs inside docker containers. In our case, we use vscode dev container instead of using the default jupyter notebook web interface, as vscode provides IntelliSense, auto formatting and many useful extensions compared to the jupyter notebook web interface.

# Implementation

This project first applies slicing for the original csv file. There is one new csv file for every 10000 rows. Slicing needs to be applied because the python kernel inside the docker environment may crash if the size of the input file is too large while using PySpark.

```
import glob
from pathlib import Path
import shutil
original_csv = glob.glob("./original_csv/*")


shutil.rmtree(Path('./csv'))
Path('./csv').mkdir(parents=True, exist_ok=True)


for file in original_csv:
    csv = pd.read_csv(file, encoding='utf-8')
    print(os.path.basename(file), len(csv.columns), len(csv), csv.columns)
    length=0
    limit=10000
    nth=0
    valid = lambda s: all(i in '0123456789' for i in str(s[csv.columns[0]]))
    while length<len(csv):
        nth+=1
        slice_csv = csv.loc[length:length+limit-1]
        slice_csv = slice_csv[slice_csv.apply(valid, axis=1)]
        slice_csv = slice_csv.dropna(thresh=3)
        slice_csv.to_csv(os.path.join("./csv", os.path.basename(file)[:-4]+'_'+str(nth)+'.csv'), index=False)
        length+=limit
```

After that, several text processing techniques are applied. First, punctuations and extra characters like the newline character are replaced with a white space using regular expression. Next, consecutive white spaces are reduced to one white space with regular expression. Then, official implementations of tokenizer and stop words remover from PySpark are applied. Finally, this project uses the lemmatizer from NLTK to transform the token into its basic form.

```
def transformation(filepath):
    dataset = pd.read_csv(filepath, encoding='utf-8')
    df = spark.createDataFrame(dataset, schema=inputSchema)

    print(os.path.basename(filepath), df.count(), len(dataset))

    punc=string.punctuation+"\n"+"\t"+"\r"

    df = df.withColumn("clearText", regexp_replace(
    "text",
    punc,
    " "
    )).withColumn("clearTextNoExtraSpace", regexp_replace(
    "clearText",
    "\\s+",
    " "
    ))

    tokenizer = Tokenizer(inputCol="clearTextNoExtraSpace", outputCol="words")

    tokenized = tokenizer.transform(df)

    remover = StopWordsRemover(inputCol="words", outputCol="filteredWords")

    removed = remover.transform(tokenized)

    wnl = WordNetLemmatizer()

    nltk.download('omw-1.4', quiet=True)
    nltk.download('words', quiet=True)
    nltk.download('wordnet', quiet=True)

    word_set = set(nltk.corpus.words.words())

    @F.udf('array<string>')
    def remove_words(words):
        res=[]
        for word in words:
            new_word=wnl.lemmatize(word)
            if new_word in word_set:
                res.append(new_word)
        return res

    final = removed.withColumn('finalWords', remove_words('filteredWords'))

    drop_list = set(('clearText', 'clearTextNoExtraSpace', 'words', 'filteredWords'))
    final = final.select([column for column in final.columns if column not in drop_list])

    targetPath = os.path.join(output_folder, "output_"+os.path.basename(filepath))
    final.toPandas().to_csv(targetPath, index=False)

    print('Done')
    return targetPath
```

## Outcome and usage

At the end, the Spark dataframe with a list of tokens is exported as a csv file. Although csv does not support the array data type, we can use python functions like literal_eval to parse it back from a string to an array, so the list of tokens can be used in upcoming processes such as TF-IDF and word cloud. Below picture shows the result.

| Topic Numbe | Topic | Text | Final Words |
|---|---|---|---|
| 0 | Managing exercise an | A number of members have posted recently abou<br>Personally I do it to stay well and fit as well as to | ['number', 'member', 'posted', 'recently', 'sport', 'exercise', 'way', 'manage', 'blood', 'glucose', 'I |
| 0 | Managing exercise an | I train and compete in powerlifting, which is a stre<br>My typical training week currently centres around<br>Competition days can be long, as I compete and a<br>My best tip for diabetes and exercise - test freque | ['train', 'compete', 'strength', 'sport', 'based', 'around', 'squat', 'bench', 'typical', 'training', 'wee |
| 0 | Managing exercise an | I used to play tennis, but had some issues with m | ['used', 'play', 'tennis', 'issue', 'coach', 'also', 'didnt', 'time', 'anyways', 'go', 'everywhere', 'bicyc |
| 0 | Managing exercise an | I never called myself a runner when I could just ru<br>I have tried to go running, I have not been able to<br>If I can only run if I do a bunch of preparation and<br>To end on something useful if not positive, anyon | ['never', 'runner', 'run', 'whenever', 'someone', 'went', 'running', 'however', 'saying', 'runner', 'qu |
| 0 | Managing exercise an | Hi @NoKindOfSusie I would get into see your GP<br>You have a goal to look forward to and you will fin | ['hi', 'get', 'see', 'get', 'checked', 'take', 'week', 'longer', 'recover', 'fully', 'best', 'run', 'symptom', |
| 0 | Managing exercise an | I'm checking my levels lots and lots, definitely bet<br>Running does make my blood sugar drop incredib<br>If you work it out there's about 5 grams of actual | ['level', 'lot', 'lot', 'definitely', 'go', 'running', 'afterward', 'yes', 'start', 'feeling', 'low', 'always', 'lo |
| 0 | Managing exercise an | Hi @Juicyj . . .<br>I recently read the book "Bright Spots and Landm<br>https://diatribe.org/bright-spots-and-landmines-r<br>Interesting to read that you exercise to prove that<br>Regards<br>Antony | ['hi', 'recently', 'read', 'book', 'bright', 'spot', 'brown', 'topic', 'insulin', 'dos', 'manage', 'exercise', |
| 0 | Managing exercise an | Hi @NoKindOfSusie - Do you have any quick actir<br>It would be great to see you get to a place where | ['hi', 'quick', 'acting', 'insulin', 'board', 'running', 'tried', 'eating', 'something', 'like', 'small', 'bana |
| 0 | Managing exercise an | I've only been diagnosed for about 8 weeks now a | ['week', 'found', 'exercise', 'thing', 'sort', 'even', 'big', 'exercise', 'like', 'running', 'something', 're |

# *VI. Data exploration*

## a. Activity of each diabetes type

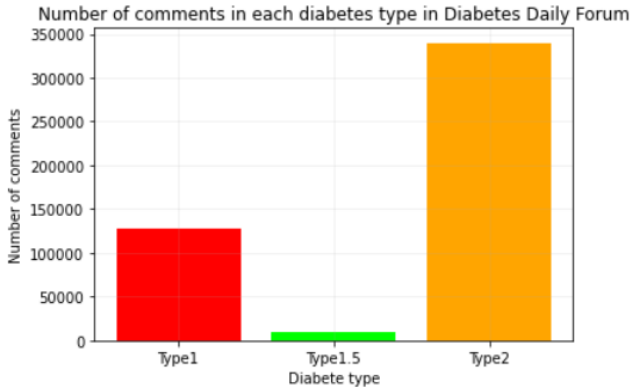
This table presents how many comments were extracted.

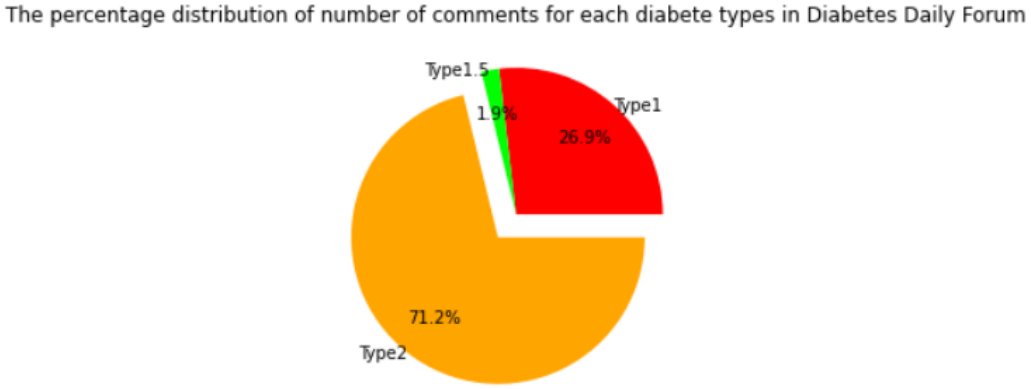
Out[16]:

| | Diabetes Daily Forum | Diabetes.co.uk | Total |
|---|---|---|---|
| **Type1** | 128189 | 222690 | 350879 |
| **Type1.5** | 9279 | 9075 | 18354 |
| **Type2** | 339813 | 300226 | 640039 |

There are too few samples (18,354 samples) in diabetes type 1.5, which is much lower than 100,000 samples. Inadequate samples reduce the power of our research. As a result, we ignore diabetes type 1.5. Next, there are the visualizations of each diabetes type's activity.

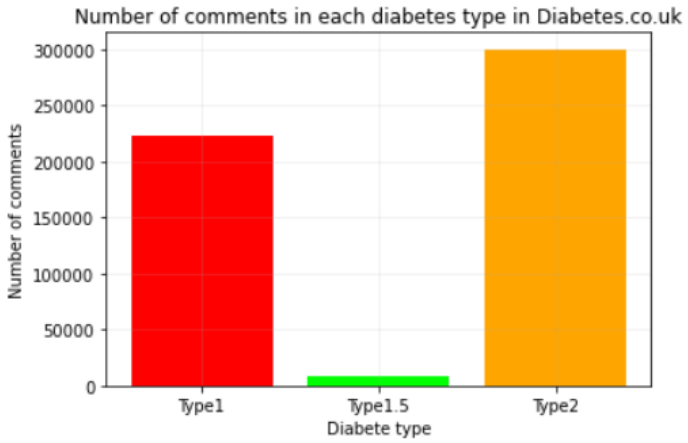
Diabetes Daily Forum

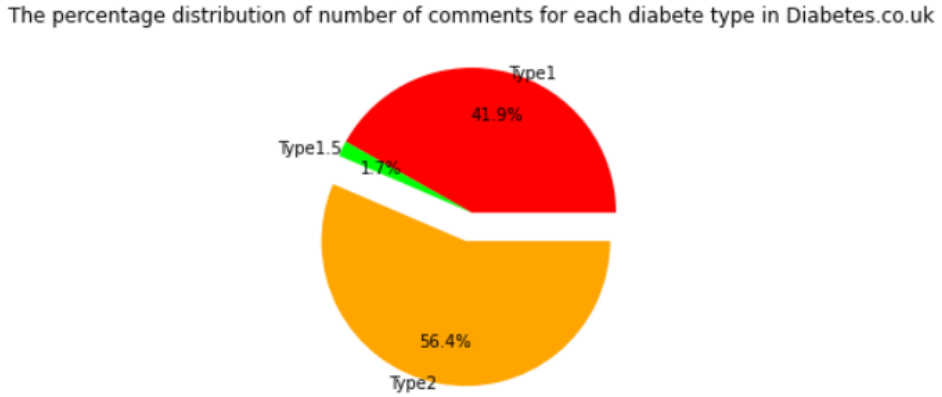



In the Diabetes Daily Forum, type 2 occupies more than two-thirds of the comments (71.2%). Type 1 occupies more than ⅓ (26.9%) of comments. Type 1.5 only occupies a tiny part of the comments. So the users in the diabetes type 2 forum have the highest activity.

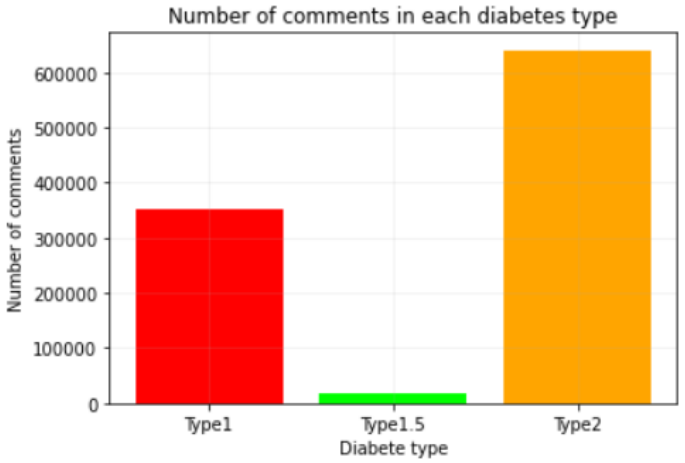
Diabetes.co.uk

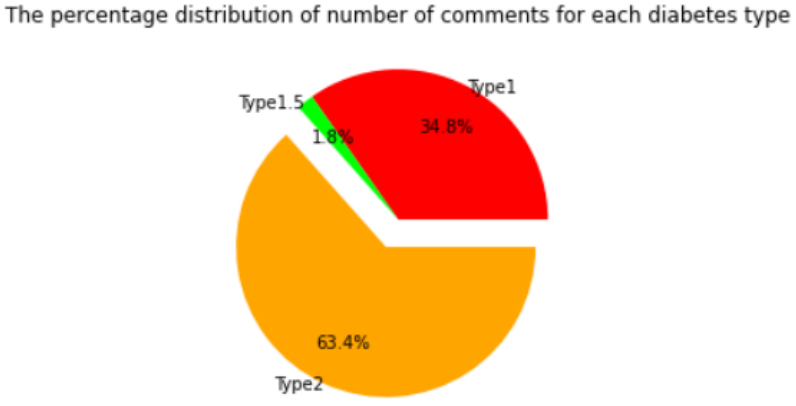





In Diabetes.co.uk, diabetes type 2 still occupies the most comments (56.4%). However, the difference (14.5%) between the number of comments in type 1 and type 2 is much smaller than in Diabetes Daily Forum (44.3%). Therefore, though in terms of diabetes patients in the world, the population of diabetes type 2 is much higher than type 1's, the comment amount's difference is not that much. One of the possible reasons is there are so many active users in the type 1 diabetes forum, they drop the comments frequently. Another possible reason is plenty of diabetes type 1 patient use this platform to find or seek the information they want.

Combining two websites




It is found that type 2 still occupies most of the comments (63.4%). It is not surprising since most diabetes patients are type 2. According to the Centers for Disease Control and

Prevention (2022), 90-95% of the population are type 2 diabetes patients among US diabetes patients. Therefore it is reasonable that most of the forum users are type 2 diabetes. Besides, diabetes type 1 occupies around ⅓ of the comments, perhaps, it is because diabetes type 1 patients love to interact with each other to seek answers/solutions to their daily life diabetes-related problems, or some of them are highly active users.

## b. LDA for topic classification

### LDA introduction

Topic modeling finds patterns of topics within a corpus. Then, group documents into different sets of computed topics. LDA (Latent Dirichlet Allocation) is a topic modeling model. LDA requires a user-defined corpus. Then it generates the result based on parameters such as the number of targeted topics. Below are the topics that are generated by LDA.

```
(0, '0.046*"test" + 0.038*"insulin" + 0.034*"type" + 0.026*"gad" + 0.023*"antibody" + 0.017*"peptide" + 0.014*"cell" + 0.014*"level" +
0.013*"diagnosis" + 0.012*"low"')
(1, '0.016*"meal" + 0.016*"unit" + 0.014*"day" + 0.013*"level" + 0.013*"hour" + 0.013*"low" + 0.012*"insulin" + 0.011*"time" + 0.010*"eat" +
0.009*"hypo"')
(2, '0.017*"fat" + 0.017*"protein" + 0.016*"low" + 0.013*"get" + 0.012*"see" + 0.009*"blood" + 0.009*"one" + 0.009*"thing" + 0.008*"sugar" +
0.008*"also"')
(3, '0.021*"insulin" + 0.016*"get" + 0.011*"see" + 0.011*"type" + 0.011*"good" + 0.010*"know" + 0.009*"much" + 0.009*"think" + 0.009*"need" +
0.008*"control"')
(4, '0.014*"think" + 0.013*"know" + 0.012*"one" + 0.010*"well" + 0.009*"done" + 0.009*"oil" + 0.009*"diagnosis" + 0.008*"thanks" + 0.008*"test" +
0.008*"really"')
(5, '0.059*"thanks" + 0.034*"lucy" + 0.022*"thank" + 0.017*"great" + 0.016*"yes" + 0.016*"reply" + 0.013*"phoenix" + 0.011*"helpful" + 0.011*"know" +
0.010*"cheer"')
(6, '0.040*"insulin" + 0.021*"diabetes" + 0.017*"type" + 0.016*"people" + 0.014*"blood" + 0.012*"level" + 0.010*"glucose" + 0.010*"diabetic" +
0.009*"sugar" + 0.009*"low"')
(7, '0.020*"insulin" + 0.017*"day" + 0.012*"weight" + 0.011*"week" + 0.010*"time" + 0.009*"good" + 0.009*"month" + 0.008*"still" + 0.008*"feel" +
0.008*"year"')
(8, '0.048*"diabetes" + 0.032*"type" + 0.017*"forum" + 0.013*"know" + 0.012*"one" + 0.009*"read" + 0.009*"sent" + 0.009*"year" + 0.008*"age" +
0.008*"hospital"')
(9, '0.040*"insulin" + 0.037*"basal" + 0.015*"bolus" + 0.014*"pen" + 0.013*"acting" + 0.012*"time" + 0.012*"need" + 0.012*"injection" + 0.011*"use" +
0.009*"re"')
```

### LDA implementation

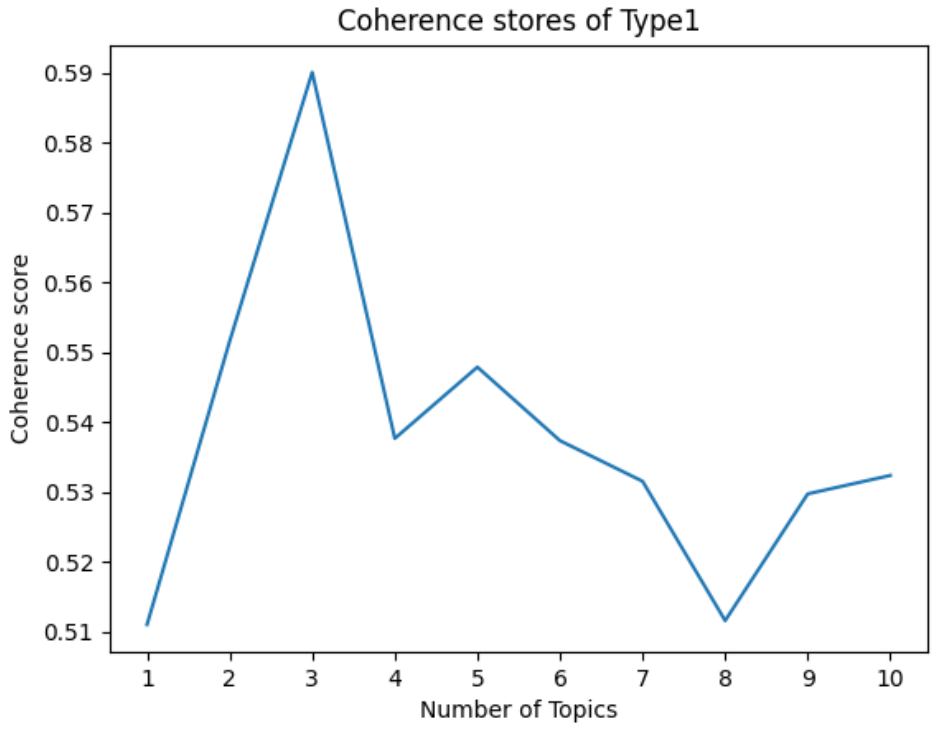
This project uses the gensim library for the implementation of LDA, and pyLDAvis for visualizing the result of LDA. First, the output directory which contains the csv with a list of processed tokens is read. Then, for each file in the output directory, if it contains a keyword such as '_Type1_', it would be parsed, and the result is appended to the text_data list, a list that contains all tokens from different csv files.

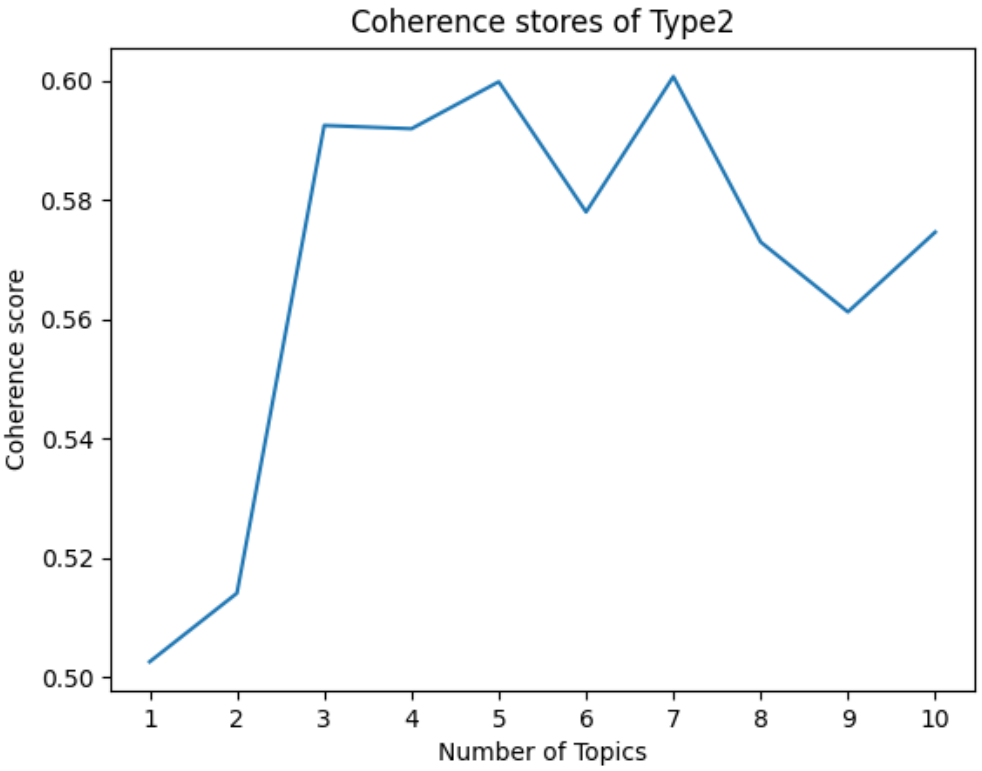
```
import glob
import ast

def get_text_data(keyword):

    text_data=[]
    # Load the data
    for file in glob.glob("./output/*"):
        if keyword in file:
            df=pd.read_csv(file)
            df['finalWords']=df['finalWords'].apply(lambda x: ast.literal_eval(x))
            text_data.extend(df['finalWords'].tolist())

    text_data=[i for i in text_data if len(i)>0]
    print('Total number of documents: ',len(text_data))
    return text_data

keywords=["_Type1_", "_Type2_"]
table={}
for keyword in keywords:
    table[keyword.strip("_")]=get_text_data(keyword)

print(table.keys())
```

After that, this project runs the provided LDA model from gensim. This project executes the LDA model with topic number ranges from 1 to 10 to find out the model with the highest topic coherence score. Topic coherence is the word similarities of top words in a given topic. A high topic coherence score indicates the model performs well. We use c_v measure to calculate the topic coherence score. Below are the implementation and topic coherence scores for different numbers of topics.

```python
import pyLDAvis.gensim_models
import time

def prepare_model(text_data, key):
    pyLDAvis.enable_notebook()
    id2word = Dictionary(text_data)

    corpus = [id2word.doc2bow(text) for text in text_data]

    print(id2word)
    print([(id2word[i], j) for i,j in corpus[:1][0]])
    scores=[]
    score=float('-inf')
    best_model=None
    for i in range(1,11):
        lda_model = LdaModel(corpus=corpus,
                    id2word=id2word,
                    num_topics=i,
                    random_state=0,
                    alpha='auto',
                    per_word_topics=True)
        if i>1:
            vis = pyLDAvis.gensim_models.prepare(lda_model, corpus, id2word)
            pyLDAvis.save_html(vis, f"{key}_{i}.html")
        coherence_model_lda = CoherenceModel(model=lda_model, texts=text_data, dictionary=id2word, coherence='c_v')
        coherence_lda = coherence_model_lda.get_coherence()
        scores.append(coherence_lda)
        if coherence_lda>score:
            score=coherence_lda
            best_model=lda_model

    print('\nCoherence Score: ', score)
    return best_model, corpus, id2word, scores

score_table={}
for key, value in table.items():
    model, corpus, id2word, scores=prepare_model(value, key)
    vis = pyLDAvis.gensim_models.prepare(model, corpus, id2word)
    pyLDAvis.save_html(vis, f"{key}_highest.html")
    score_table[key]=scores
```

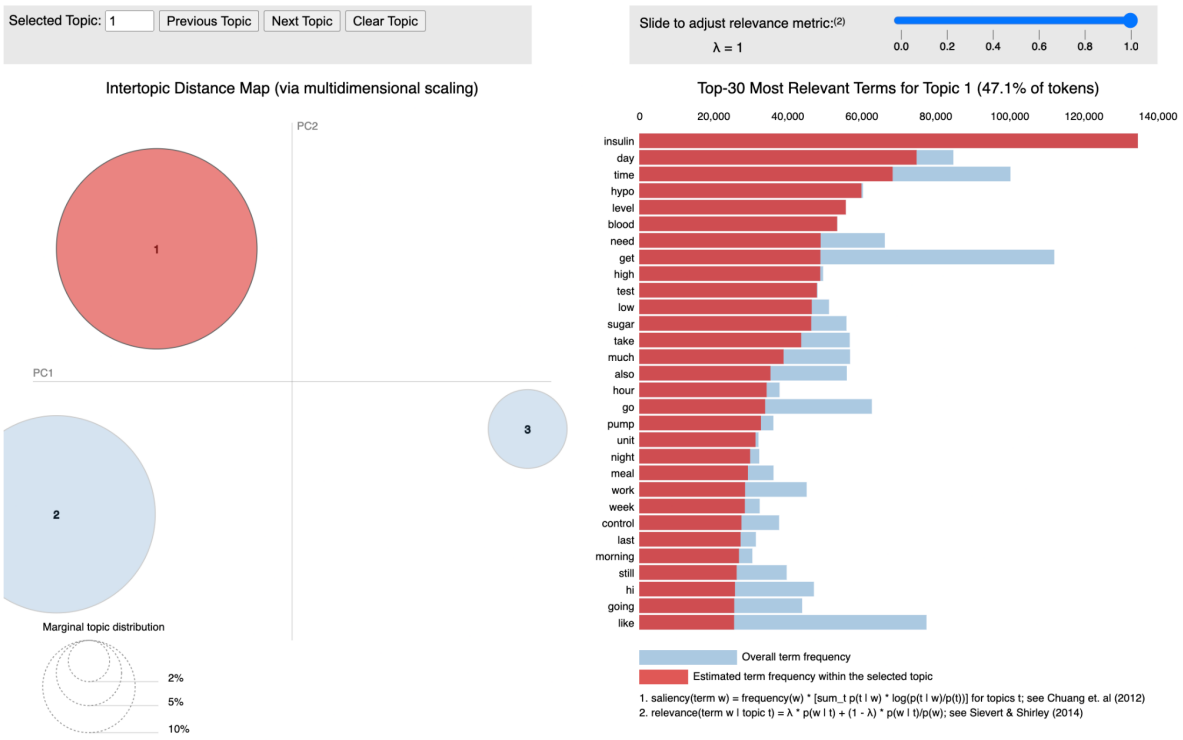
pyLDAvp

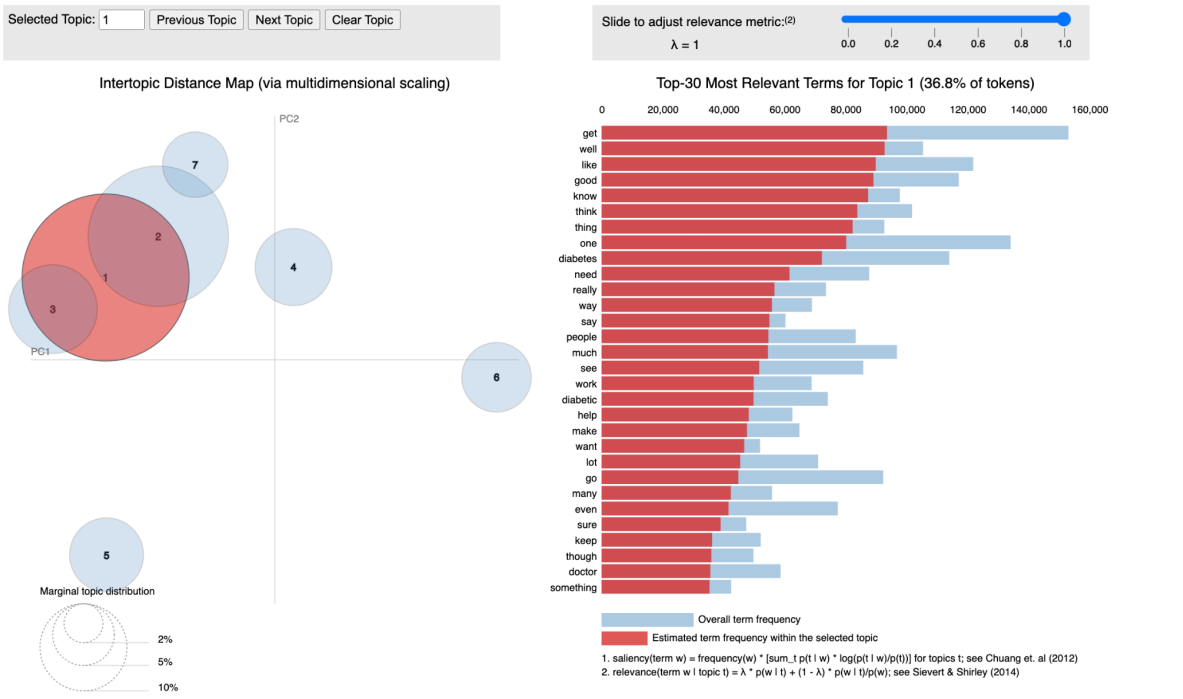





## LDA result

For type 1 diabetes, the optimal number of topics is 3, which has a topic coherence score of 0.590107437087784. For type 2 diabetes, the optimal number of topics is 7, which has a topic coherence score of 0.6006481267679115. We use pyLDAvis to visualize the result. This project saves the visualization result for every computed LDA model, and saves the model with the highest topic coherence score separately. Below are the saved visualizations. The first picture belongs to type 1 diabetes and the second picture belongs to type 2 diabetes.


Selected Topic: 1
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4 0.6 0.8 1.0
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
Marginal topic distribution
2%
5%
10%
Top-30 Most Relevant Terms for Topic 1 (47.1% of tokens)
0 20,000 40,000 60,000 80,000 100,000 120,000 140,000
insulin
day
time
hypo
level
blood
need
get
high
test
low
sugar
take
much
also
hour
go
pump
unit
night
meal
work
week
control
last
morning
still
hi
going
like
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)


Selected Topic: 1
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4 0.6 0.8 1.0
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
6
7
Marginal topic distribution
2%
5%
10%
Top-30 Most Relevant Terms for Topic 1 (36.8% of tokens)
0 20,000 40,000 60,000 80,000 100,000 120,000 140,000 160,000
get
well
like
good
know
think
thing
one
diabetes
need
really
way
say
people
much
see
work
diabetic
help
make
want
lot
go
many
even
sure
keep
though
doctor
something
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

# VII. Methodology

## a. TFIDF

TfidfVectorizer in sklearn is used for calculating the Term Frequency - Inverse Document Frequency (TFIDF) scores. TFIDF reflects the importance of the specific term to a document in the whole corpus. By analyzing and comparing the TFIDF of Type 1 and Type 2 diabetes, we may possibly figure out some patterns and differences in the needs of patients.

Implementation steps:

1. The first step is to group the comments by topics. Simultaneously, loop through the "finalWords" column to obtain a list of unique words from all tokenized words. There are two ipynb files for each type since one is for outputting the final TFIDF scores and the other one is for filtering and generating a list of unique words and visualizing its term frequency. TfidfVectorizer requires lists when "dictionary fromkeys" function for finding unique words require Strings as input.

| Type 1.ipynb | unique words - Type 1.ipynb - Type 1 |
|---|---|
| `tempcommentlist = eval(df.finalWords[i])`<br>`for j in tempcommentlist:`<br>`  j = j.replace(" ",'')`<br>`  temp.append(j)`<br>`  if not(j in alluniquedata):`<br>`  alluniquedata.append(j)`<br>`alldata.append(temp)` | `listlen = len(df.finalWords)`<br>`df.finalWords = df.finalWords.str.strip('[]').str.split(',')` |

2. In unique words - Type 1.ipynb, counting frequency of unique words and filter the words appearing less than 200 times in total documents for reducing redundant calculations. Letters and words containing only two letters will be removed as well.

```
#Counting frequency of unique words
alldict = []
count = []
fcount = []
w = []
c = []
wordnum = dict.fromkeys(alluniquedata,0)
for j in databytopic:
    for i in j:
        wordnum[i]+=1
for i in wordnum:
    tmp = i.replace("'",'')
    count.append([tmp,wordnum[i]])

count.sort(key = lambda x: x[1],reverse=True)
for i in range(len(count)):
    if (len(count[i][0])>2) & (count[i][1]>200):
        fcount.append(count[i])
        w.append(count[i][0])
        c.append(count[i][1])
```

3. Generating TF graphs and save the word list

4. Going back to the Type 1.ipynb, filter the words that are not within the unique words list and combine the tokenized words according to topics.

5. Run the tfidf_vectorizer for the topic list, then sum up and sort the order of tfidf for each terms.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer


df = pd.DataFrame({"texts": filtertestdataalltopic})
tfidf_vectorizer = TfidfVectorizer(ngram_range=[1, 1])
tfidf_separate = tfidf_vectorizer.fit_transform(df["texts"])

df_tfidf = pd.DataFrame(
    tfidf_separate.toarray(), columns=tfidf_vectorizer.get_feature_names(), index=df.index
)
```

Similar process will be done to Type 2 csv files.

## b. Word cloud

To visualize data, we may usually use charts or graphs to present our data. However, these methods may not fit our project as our data are textual data. WordCloud is one of the data visualization tools to highlight important textual data. If a specific word frequently appears in a collection of textual data, then the word will appear in the word cloud with a bolder and larger font, showing that the word is important in the data collection, and vice versa. Wordcloud is usually applied to convey important information, which fits our project objectives, to convey the needs and sentiment analysis to the diabetic patients and the public.

The implementation of WordCloud is not complicated. First, the installation of wordcloud can be executed by the following command.

```
In [80]: !pip install WordCloud
```

Once the installation is completed, tokenized words can be inputted for wordcloud generation. As we have created different csv files to store the tokenized words of each discussion topic, we can now simply extract the column that contains the tokenized word of each csv file, grouped by each diabetes category and the respective discussion forums (couk_type1, couk_type2, dailyforum_type1, dailyforum_type2), and store it into a list.

```
#READ COUK_Type2
from pathlib import Path

#get COUK_Type2 csv
path=r'D:/Matthew/University/Sem7/CS4480/project/output/'

fileName=["output_Diabetescouk_Type2_CommentPerTopic_[0-9].csv",
          "output_Diabetescouk_Type2_CommentPerTopic_[0-9][0-9].csv"]

COUK_TYPE2_dfs=[]

for group in fileName:
    for file in Path(path).glob(group):
        COUK_TYPE2_dfs.append(pd.read_csv(file))
COUK_TYPE2_DF=pd.concat(COUK_TYPE2_dfs)
print(COUK_TYPE2_DF)
```

The extraction of Diabetes.co.uk (type2) is taken as an example. Using the path library allows us to render all the csv file directories that meet the condition we have set. In our data, 31 csv files are generated for Diabetes.co.uk (type2). First, the directory is constructed by the base path and combined with the directory with wildcards so that only the csv files of Diabetes.co.uk (type2) are selected. Then, the pandas library can start reading the selected files under the loop and store the dataframes to the dataframes list. Finally, we can concatenate all the dataframes into one.

```
COUK_TYPE2_DATA_LIST=[]
for row in COUK_TYPE2_DF['finalWords']:
    data=eval(row)
    COUK_TYPE2_DATA_LIST.extend(data)
len(COUK_TYPE2_DATA_LIST)
```

Next, we extract the 'finalWords' column of the dataframe that contains all the dataframes of each csv file. Then, all the tokenized words are extracted and stored into a data list.

```
from wordcloud import WordCloud
import numpy as np
from PIL import Image

mask = np.array(Image.open("cloud.png"))
wordcloud2=WordCloud(width=3000, height=2000, max_words=50, background_color="white"
,mask=mask).generate(' '.join(COUK_TYPE2_DATA_LIST))

wordcloud2.to_file("couk_type2.png")
```

Lastly, using the wordcloud library can generate word cloud of the data list we have extracted. The wordcloud parameters are set to have a maximum of 50 important words with specific height, width and also shape using the PIL and numpy library to shape the wordcloud. The wordcloud is finally saved as png. The wordcloud generation process is similar for other types of diabetes and discussion forums.

## c. Sentiment analysis

VADER is the package used in sentiment analysis. According to VaderSentiment 3.3.1 documentation, it is a package that is a pre-trained package for a sentence's sentiment analysis. The sentiment analysis can work with vader_lexicon, a VADER self-designed lexicon that works by labeling the words, Western-style emoticons (e.g. ":-)"), sentiment-related acronyms and initialisms (such as" LOL"), and commonly used slang with sentiment value (such as meh). VADER also works with the bag-of-words approach at the same time. No training is needed, it is convenient and easy to use, hence, it is a red-hot tool for business sentiment analysis in social media comments.

Using VADER is simple, first, import the package (vaderSentiment and nltk).
*pip install vaderSentiment*
*pip install nltk*

Then, download 'vader_laxicon' (the dictionary used in VADER from nltk) and import SentimentIntensityAnalyzer from nltk.sentiment.vader.

```
import nltk
nltk.download('vader_lexicon')

from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

Next, initialize the sentiment analysis and use a variable (sentiment) to hold it.

```
sentiment=SentimentIntensityAnalyzer() #initialize
```

After that, plug the sentence string into the sentiment variable.

Finally, obtain the sentiment score from the key ' compound'. VaderSentiment 3.3.1 documentation states it indicates whether the sentiment is positive, negative, or neutral. The 'compound' is normalized between -1 to +1.
(NOTE: Since Joblib is problemed with slicing, we did not use it.

| Indication | Compound score |
|---|---|
| Positive | compound score >= 0.5 |
| Neutral | (compound score > -0.5) and (compound score < 0.5) |
| Negative | compound score <= -0.5 |

In [7]:
```
#Examples
list_=["This phone is extremely bad.","This phone is not that good.","This phone is very good."]
sentiment_list=[]

print("Example 1 Only using vader\n")
for i in list_:
     print(i,': ',sentiment.polarity_scores(i))
print("\n--------------------------------------------------\n")

print("Example 2 Only using vader 's compound (Sentiment score)\n")
for i in list_:
     print(i,': ',sentiment.polarity_scores(i)['compound'])
print("\n--------------------------------------------------\n")

print("Example 3 Only using vader 's compound (Sentiment score) and append\n")
for i in list_:
     sentiment_list.append(sentiment.polarity_scores(i)['compound'])

for i in range(len(list_)):
     print(i,': ',sentiment_list[i])
print("\n--------------------------------------------------\n")
```

Output:

```
Example 1 Only using vader

This phone is extremely bad. :  {'neg': 0.487, 'neu': 0.513, 'pos': 0.0, 'compound': -0.5849}
This phone is not that good. :  {'neg': 0.325, 'neu': 0.675, 'pos': 0.0, 'compound': -0.3412}
This phone is very good. :  {'neg': 0.0, 'neu': 0.556, 'pos': 0.444, 'compound': 0.4927}

--------------------------------------------------

Example 2 Only using vader 's compound (Sentiment score)

This phone is extremely bad. :  -0.5849
This phone is not that good. :  -0.3412
This phone is very good. :  0.4927

--------------------------------------------------

Example 3 Only using vader 's compound (Sentiment score) and append

0 :  -0.5849
1 :  -0.3412
2 :  0.4927

--------------------------------------------------
```

Therefore, our workflow is:

Target: 1. Understand which diabetes type needs more help by finding which type has the more negative sentiment.

Target 2: Understand what help is needed from the texts in most negative 20 comments.

- Take diabetes type 1 as code example

1. Read the CSV file.

In [9]:
```
#read csv file
t1=pd.read_csv("Type1.csv",index_col=0)
```

2. Insert the sentiment column to store the sentiment

In [11]:
```
#Make a new column for sentiment
#I just make some values to sentiment's column first as initialization
t1.insert(2,'sentiment',list(range(1,len(t1)+1)))
```

3. Some content is empty. So it remains “nan”, which is a float data type, not compatible with VADER. So we need to drop “nan” values in the data.

In [12]:
```
#check how many comments in float type since they oppose our sentiment analysis
float_num=0
for i in list(range(1,len(t1)+1)):
    if (type(t1["text"][i])==float):
        print(i,"Content:",t1["text"][i])
        print("---------------------------------")
        float_num+=1
print("Number of float-type comments:",float_num)
```

```
73 Content: nan
---------------------------------
109 Content: nan
---------------------------------
215 Content: nan
---------------------------------
256 Content: nan
---------------------------------
```

In [13]:
```
#drop all nan
t1=t1.dropna()
```

4. Adjust the index

In [14]:
```
#index rearrangement
t1.index=list(range(1,len(t1)+1))
```

5. Define a function, which bases on the procedure in the example for adding the sentiment score of each text.

In [16]:
```
def add_sentiment(j):
    t1["sentiment"][j]=sentiment.polarity_scores(t1["text"][j])['compound']
```

6. Adding the sentiment score by the for loop and defined function.

In [27]:
```
#Now try the whole stuff
start=time.time()
for i in list(range(1,len(t1)+1)):
    add_sentiment(i)
end=time.time()
normal_exe_time=end-start
print("The time of normal computing is:",normal_exe_time," s")
```

```
C:\Users\kiusandy\AppData\Local\Temp\ipykernel_33456\3648476584.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  t1["sentiment"][j]=sentiment.polarity_scores(t1["text"][j])['compound']
```

```
The time of normal computing is: 7587.370795249939  s
```

7. Conduct a box plot, summary of statistics, and pie chart. (The visualizations can be found in **Result and analysis -> Sentiment analysis**)
Statistics and boxplot

In [41]:
```
print("Statistics of sentiments")
print("Maximum: ",t1["sentiment"].max())
print("Upper quantile: ",t1["sentiment"].quantile(.75))
print("Mean: ",t1["sentiment"].mean())
print("Lower quantile ",t1["sentiment"].quantile(.25))
print("Minimum: ",t1["sentiment"].min())
```

```
Statistics of sentiments
Maximum:  0.9998
Upper quantile:  0.7783
Mean:  0.2658757233638426
Lower quantile  -0.1025
Minimum:  -0.9997
```

In [50]:
```
#box plot
import matplotlib.pyplot as plt
plt.boxplot(t1["sentiment"])
plt.xlabel("Comment")
plt.ylabel("Sentiment")
plt.title("Type 1's comment's sentiment 's distribution")
plt.show()
```

Pie chart

In [19]:
```
t1_pos=0 #type 1 's positive comment
t1_neu=0 #type 1 's neutral comment
t1_neg=0 #type 1 's negative comment

for i in t1["sentiment"]:
    if i>=0.5:
        t1_pos+=1
    elif i>-0.5 and i<0.5:
        t1_neu+=1
    elif i<=-0.5:
        t1_neg+=1

print("Number of positive comments: ",t1_pos)
print("Number of neutral comments: ",t1_neu)
print("Number of negative comments: ",t1_neg)
```

```
Number of positive comments:  151364
Number of neutral comments:  149521
Number of negative comments:  47833
```

```
In [21]: sentiment_list=["Positive","Neutral","Negative"]
         data_list_type1=[t1_pos,t1_neu,t1_neg]
         color_list=["red","cyan","grey"]

In [22]: #pie chart
         import matplotlib.pyplot as plt
         plt.pie(data_list_type1,
                 labels=sentiment_list,
                 colors=color_list,
                 labeldistance=1.1,
                 autopct="%2.1f%%")
         plt.title("Sentiment distribution of type 1")
         plt.show()
```

8. Drop the text column (easier for analysis).

```
In [80]: t1_group=t1.copy()

In [81]: #Drop the text column for groupby
         t1_group=t1_group.drop('text',axis=1)
```

9. Store the dataset with sentiment score in CSV.

```
In [28]: t1
```

Out[28]:

| | topic | text | sentiment |
|---|---|---|---|
| 1 | Learning Center - Type 1 Diabetes | For up to date information about Type 1 Diabet... | 0.5661 |
| 2 | Learning Center - Type 1 Diabetes | I have a question about insulin antibody testi... | -0.7096 |
| 3 | Learning Center - Type 1 Diabetes | In general, a reference range for any given te... | 0.6335 |
| 4 | Learning Center - Type 1 Diabetes | Thank you for the reply. I think you must be ... | -0.5267 |
| 5 | Learning Center - Type 1 Diabetes | Medtronics insulin pump with MiniMed Quick -se... | -0.7644 |
| ... | ... | ... | ... |
| 348714 | Test strips | I test regularly, about 4 times a day, so in a... | 0.0000 |
| 348715 | Test strips | I test on adverage just over 50 strips a week,... | -0.9775 |
| 348716 | Test strips | Well now you say it, yes and not very prett !... | 0.6114 |
| 348717 | Welcome to the Type 1 Diabetes forum | Hope this is useful to our visitors...\n\nDan | 0.7003 |
| 348718 | Welcome to the Type 1 Diabetes forum | Random new forum outta nowhere! Let's have a... | 0.0000 |

348718 rows × 3 columns

```
In [29]: t1.to_csv("Type1_withSentiment.csv")
```

10. Group the text by its topic to compute each topic's comment average sentiment.

11. Find out the most negative 20 topics. Then visit these topics in the discussion forum or open the dataset in excel to see the whole comments to find out what help(s) they need. (The results and discovery of comments can be found in **Result and analysis -> Sentiment analysis**)

12.

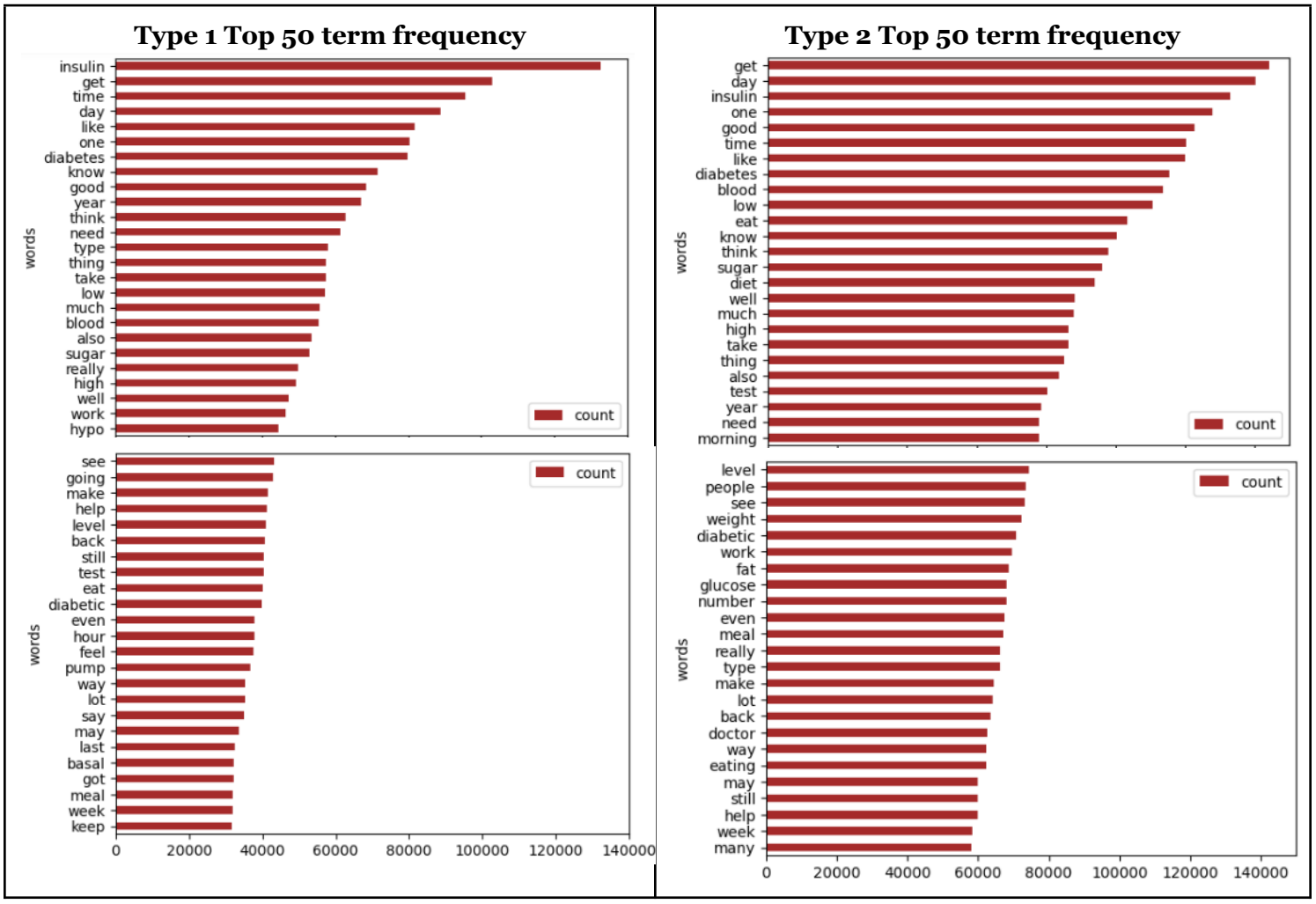
```
In [97]: #The 20 most negative sentiment post (find by average)
         group.sort_values(ascending=False)[-20::]
```

Extraction time of diabetes type 1 and diabetes type 2

| Diabetes type | Time | Number of texts (After drop nan values) |
|---|---|---|
| Type 1 | 7587.37 s | 348718 |
| Type 2 | 9154.32 s | 636515 |

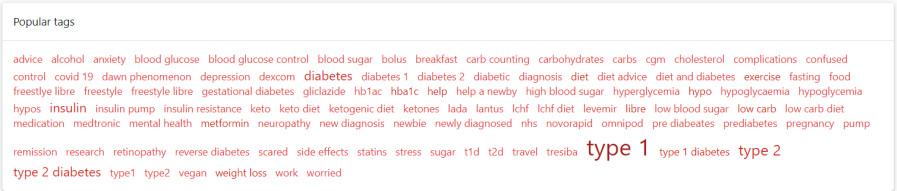
# *VIII. Result and analysis*

## a. TFIDF




We can observe that the terms “insulin”, “blood”, “sugar”, “level”, “low” have high frequencies in both types. This is because the patients need to check their blood sugar level using the insulin pump. On the other hand, the terms with lowest frequency are more likely to be adjectives or descriptive terms which are less or even not relative to diabetes nor patients’ feelings.

In comparison to the popular tags on the website of diabetes.co.uk, we can also find that patients usually search for the phrases related to blood sugar and low carbon diets. By searching the keywords “insulin”, “blood”, “sugar”, “level” in the forum, it is found that patients are required to do the HbA1c test that reflects average blood sugar(glucose) level of the past 2-3 months of a person.



We can also compare the TF ranking between type 1 and type 2. It is found that words such as eat, diet, weight and fat are mentioned more often in type 2.

**Type 1 TFIDF (Top 100 TF)**

| | | | |
|---|---|---|---|
| insulin | 3500.619530 | lot | 1099.199760 |
| get | 2584.948962 | say | 1094.933600 |
| diabetes | 2448.483390 | last | 1088.988240 |
| time | 2343.125814 | night | 1088.138767 |
| day | 2258.450528 | got | 1087.670577 |
| one | 2086.353623 | problem | 1076.153358 |
| like | 2084.480493 | way | 1066.097288 |
| year | 1995.542273 | forum | 1065.087602 |
| know | 1943.654684 | control | 1057.013845 |
| type | 1869.839029 | better | 1056.299580 |
| good | 1848.226671 | doctor | 1039.332578 |
| blood | 1774.408307 | sure | 1038.063872 |
| sugar | 1768.370527 | month | 1024.782366 |
| low | 1716.907939 | want | 1024.321546 |
| take | 1696.227530 | new | 1021.334667 |
| need | 1692.469586 | try | 1020.277484 |
| think | 1671.429009 | long | 1017.299463 |
| high | 1581.964483 | never | 1016.660404 |
| thing | 1579.856286 | find | 1010.596055 |
| hypo | 1547.970411 | food | 1009.427290 |
| also | 1542.453514 | many | 1000.742568 |
| much | 1526.815764 | morning | 997.430631 |
| really | 1454.825968 | always | 984.207849 |
| level | 1439.504744 | said | 980.639387 |
| pump | 1421.007218 | bolus | 972.110353 |
| help | 1416.926252 | glucose | 971.644318 |
| well | 1411.306506 | first | 957.764831 |
| work | 1405.431980 | bit | 951.487025 |
| test | 1402.073364 | hope | 943.386082 |
| diabetic | 1372.238011 | something | 939.645043 |
| hour | 1348.081967 | right | 920.782102 |
| see | 1321.683266 | dose | 908.646813 |
| eat | 1305.560426 | life | 892.767943 |
| basal | 1291.662994 | great | 885.771861 |
| back | 1283.569305 | reading | 884.269793 |
| people | 1271.320061 | around | 879.053018 |
| feel | 1256.382392 | since | 870.963607 |
| going | 1255.878332 | give | 868.255576 |
| make | 1241.111571 | getting | 864.755505 |
| use | 1239.454481 | change | 864.095971 |
| still | 1210.272982 | eating | 863.940915 |
| meal | 1204.066092 | might | 854.547402 |
| unit | 1184.022367 | start | 842.087121 |
| thanks | 1165.834230 | used | 841.093141 |
| week | 1164.194804 | best | 840.140197 |
| may | 1134.757457 | little | 831.846549 |
| even | 1126.731779 | normal | 830.935436 |
| keep | 1102.299884 | though | 823.578572 |
| | | different | 804.486043 |
| | | every | 802.791028 |
| | | look | 790.898758 |
| | | body | 786.769743 |

**Type 2 TFIDF (Top 100 TF)**

| | | | |
|---|---|---|---|
| get | 3092.932419 | way | 1481.112966 |
| insulin | 3085.496245 | still | 1471.838786 |
| day | 3000.792316 | fat | 1467.374365 |
| diabetes | 2840.888309 | going | 1449.153685 |
| blood | 2729.992387 | many | 1449.117457 |
| one | 2710.562656 | thanks | 1402.102901 |
| good | 2687.511531 | say | 1401.799894 |
| low | 2651.334123 | last | 1401.459831 |
| time | 2639.459599 | better | 1393.209282 |
| like | 2580.242698 | morning | 1359.666655 |
| sugar | 2539.371175 | problem | 1348.241509 |
| eat | 2440.037354 | use | 1345.109423 |
| know | 2337.467770 | exercise | 1344.905478 |
| test | 2308.932982 | result | 1342.592630 |
| take | 2306.822647 | taking | 1335.546414 |
| diet | 2193.714799 | great | 1332.567709 |
| well | 2131.822856 | want | 1320.188337 |
| think | 2124.432816 | done | 1294.873697 |
| high | 2075.674957 | got | 1286.107568 |
| year | 2065.873787 | said | 1271.447495 |
| level | 2051.680385 | normal | 1266.549123 |
| also | 2032.826472 | first | 1265.816097 |
| thing | 1972.709363 | body | 1256.687352 |
| number | 1965.404329 | since | 1245.785756 |
| doctor | 1946.283214 | find | 1241.637690 |
| much | 1936.511501 | control | 1240.214043 |
| food | 1914.580276 | try | 1234.271719 |
| weight | 1896.360041 | effect | 1210.716147 |
| meal | 1876.332157 | meter | 1205.969373 |
| diabetic | 1872.048453 | never | 1202.447274 |
| need | 1857.582863 | sure | 1201.838769 |
| see | 1821.017216 | something | 1172.111226 |
| type | 1811.658814 | forum | 1162.350128 |
| work | 1809.917333 | fasting | 1141.505106 |
| help | 1769.017834 | long | 1131.951278 |
| week | 1686.430679 | right | 1131.021565 |
| hour | 1672.213241 | new | 1103.971804 |
| back | 1671.956406 | might | 1094.546132 |
| glucose | 1666.295207 | two | 1092.540135 |
| really | 1664.217078 | lower | 1067.442725 |
| people | 1641.587366 | bit | 1061.989198 |
| make | 1606.216614 | always | 1050.234391 |
| reading | 1578.131661 | give | 1044.336077 |
| eating | 1577.647424 | little | 1043.524241 |
| may | 1562.750685 | around | 1026.790849 |
| lot | 1559.363883 | getting | 1006.999481 |
| month | 1552.343546 | different | 1000.047238 |
| even | 1523.117086 | though | 996.618153 |
| keep | 1509.922535 | maybe | 974.824329 |
| feel | 1486.748494 | today | 864.733387 |

Furthermore, we can compare the ranking of head and tail TFIDF results of both types. We can find that the term time, hypo, pump are much more important keywords to type 1 than type 2. This is because type 1 patients are required to be very strict about the time of having meals to control their blood sugar.

After we go through the related comments, it is found that since type 2 is an acquired disease, when type 1 is congenital. A major reason that type 2 patients result in diabetes is they absorb too much fats and sugar from food. Thus, to alleviate the symptoms, type 2 usually starts with dieting. On the other hand, since type 1 is congenital, even if they have normal diets, they still suffer from diabetes.

Besides , we can find terms such as “eat”, “low” and “meal” having high TFIDF scores in type 1. Below are a few examples of the comments of the topic related to the keywords “insulin”, “blood”, “sugar”, “level” and “type 1”.

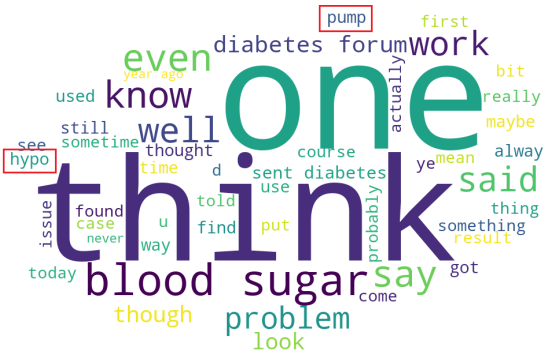
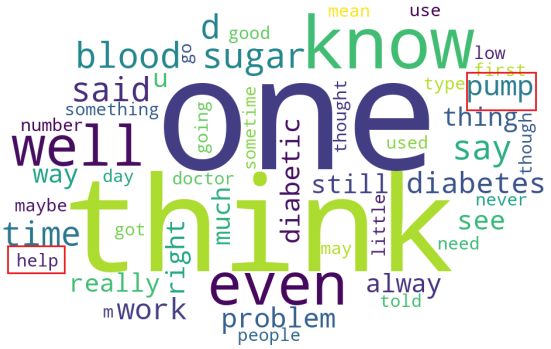
| | |
|---|---|
| Comment 1 (Type 1) | I'm lowish carb (less than 100g a day) because I have issues with weight gain and insulin resistance and low carb stops me from gaining any more or needing super large quantities of insulin.<br><br>But I inject insulin if I really want to eat a high carb meal, I just accept that I might get the dose wrong. |
| Comment 2 (Type 1) | I follow a low carb/keto diet based on Bernstein. The flip side of the coin is to have normal blood sugars which is freeing. I'm living a normal life without the stress of following that rollercoaster. |

The above results and examples show the importance of controlling intake of carbohydrates for type 1 patients, since carbohydrates will break down into sugar within the blood. When the amount of carbohydrates exceeds the insulin inside the body that can cope with, the blood sugar will rise. Thus, we can find that the Type 1 patients do not focus on the amount of food intake, but the choice of food that contains lower carbohydrates.

In conclusion, the major need of type 1 patients is to monitor their blood sugar regularly, while for type 2 patients is dieting to lose weight.

## b. Word cloud

Type1 word cloud analysis:

| Diabetes.co.uk (Type1): | Daily forum (Type1): |
|---|---|
|  |  |

For the word cloud analysis of type1 diabetes, we can see that most of the generated words are generic. Some important words related to the topic of diabetes such as pump (insulin injection) and hypo (fainting due to low blood sugar) are specific to the type1 but not appearing in type2 wordcloud, which means the type1 patients may suffer from these problems. The sentiments of type1 word cloud is relatively neutral, but they tend to need more assistance compared with type2 diabetes. For example, we can see the word help is present in daily forum (type1) wordcloud.

Type2 word cloud analysis:

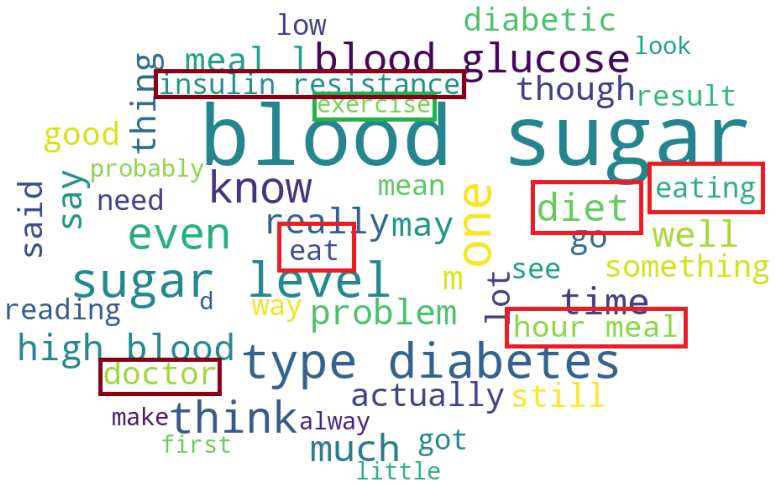
| Diabetes.co.uk (Type2): | Daily forum (Type2): |
|---|---|
|  |  |

For the word cloud analysis of type2 diabetes, we can see that the most common and frequent word is blood sugar and the result is relatively more related to the topic of diabetes, and we may use the term blood sugar for further investigation to see why this word frequently appeared in type2 diabetes. For the sentiments of type2 wordcloud, it is relatively more positive, given the words well done, believe, right are present in the word clouds.

To investigate why the term 'blood sugar' is frequently appearing in both type2 forums, we make a nested wordcloud by filtering the data in csv files.

```
#type2 concat
type2_dfs=COUK_TYPE2_DF.append(DAILY_TYPE2_DF)

#blood sugar word cloud
bs_filter = type2_dfs['text'].str.contains("blood sugar", na=False)

type2_bs=[]
for row in type2_dfs[bs_filter]['finalWords']:
    data=eval(row)
    type2_bs.extend(data)
```

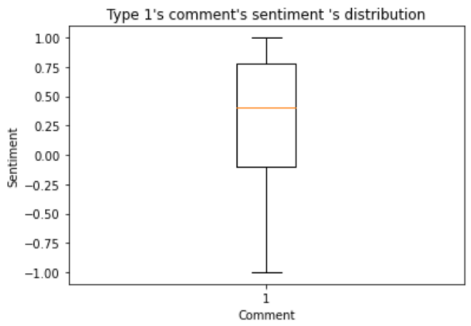
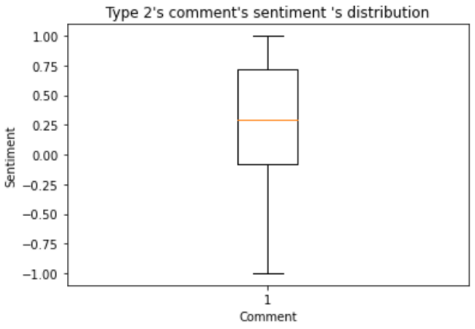
Executing the above code can filter out the data in csv files. First, we append both type2 forums into one dataframe. Then, we can check the text column (original text of forum posts) of the csv files to check whether the post text contains the term 'blood sugar', if so, we select those columns. Finally, the tokenized words of the filtered rows are extracted for further analysis.




Another word cloud is generated based on the term 'blood sugar'. From the wordcloud, there are many terms related to the eating habits, for example the words eat, eating, hour meal, diet. This is possibly the main cause for type2 diabetes. Another factor could be insulin resistance. The needs for these types of people can also be concluded from the word cloud. First of all, they need to transform their eating habits, words such as doctor and exercise are also present and these are something they have to improve as exercise and eating habits are the essential elements of the management of type 2 diabetes, according to the experiment conducted by Allem (2016).

## c. Sentiment analysis

The box plot

| Diabetes type 1 | Diabetes type 2 |
| --- | --- |
|  |  |





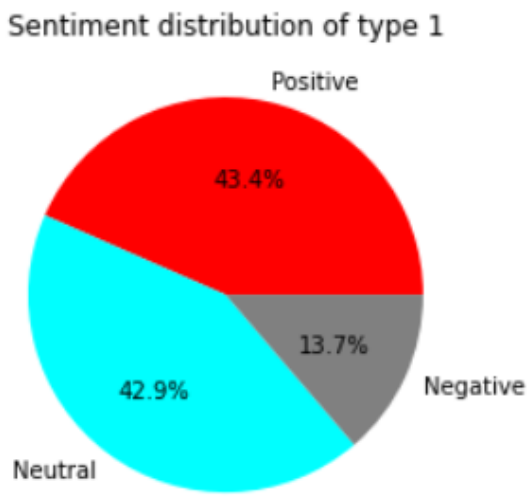
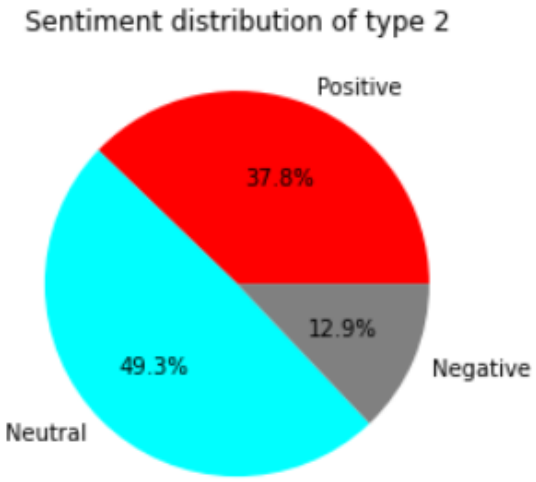
The statistics

| Diabetes type 1 | Diabetes type 2 |
| --- | --- |
| Statistics of sentiments<br>Maximum: 0.9998<br>Upper quantile: 0.7783<br>Mean: 0.2658757233638426<br>Lower quantile -0.1025<br>Minimum: -0.9997 | Statistics of sentiments<br>Maximum: 0.9998<br>Upper quantile: 0.71715<br>Mean: 0.22697844072806617<br>Lower quantile -0.07515<br>Minimum: -0.9996 |

The box plots do not present a huge difference in both types' sentiments. The maximum is the same. The minimum is very similar. The upper quartile of type 1 (0.7783) is a bit higher than type 2's (0.71715), and both are positive. The mean of type 1 (0.2659) is a bit higher than type 2's (0.2270), and both are neutral. The lower quartile is type 2 (-0.07515) is a bit higher than type 1's (-0.1025). Therefore, by viewing the boxplot and statistics, no significant difference between the two types' sentiments can be found.

Therefore, the pie charts are created to find out the difference in their sentiment.

The pie charts

| Diabetes type 1 | Diabetes type 2 |
|---|---|
|  |  |





According to the pie chart, the percentage of negative comments in type 1 (13.7%) is higher than in type 2's (12.9%). Therefore, type 1 patients have more negative sentiments than type 2 patients.

After averaging the sentiment score, 'compound' in VADER per topic, the top 20 negative topics of each type are obtained:

Type 1

| | Topic | Sentiment |
|---|---|---|
| 1 | "More Ways to Cope With Type 1 Diabetes" | -0.98710 |
| 2 | the revolution starts now | -0.98480 |
| 3 | Rapid D sets-just be aware of this | -0.98060 |
| 4 | Addison's * Diabetes Questions? | -0.97890 |
| 5 | Help With Finding Alternate Diabetes Supplier | -0.97880 |
| 6 | LADA | -0.97120 |
| 7 | how does stress affect your bg? | -0.97060 |
| 8 | you will just have another honeymoon perio d before your immune system destroys that pancreas too | -0.96360 |
| 9 | Neuropathic Pain | -0.96330 |
| 10 | Been There, Done.....it all?? | -0.96330 |
| 11 | Kidney/Stomach/Gland Issues | -0.95650 |
| 12 | Gastroparesis bolusing | -0.95450 |
| 13 | Floaters after Haemorrhage | -0.95240 |
| 14 | emotionally drained | -0.94410 |
| 15 | Now I want a suggestion | -0.94045 |
| 16 | What's so special about " Serums "? | -0.93860 |
| 17 | GP telling me nothing is wrong, leading to bad infection | -0.92270 |
| 18 | Is it because i had a hypo????? anyone hel p?? | -0.92240 |
| 19 | Type 1 and glandular fever | -0.91910 |
| 20 | is Apologies for rant! | -0.91905 |

Type 2

| topic | Sentiment score |
|---|---|
| How dos one cope with fake lows | -0.91985 |
| Little bit goes a long way. | -0.92870 |
| Finally able to check bg after regular Saturday dinner - surprised | -0.92910 |
| Diabetics face risk on drug choices | -0.93490 |
| Diabetes and Gallstones | -0.93530 |
| The Predators on the Prowl of our Beta Cells (hunting license not required) | -0.94290 |
| feeling generally unwell | -0.94345 |
| Anxiety, Neuropathy, Back Pain, Unsteady BG Readings @ Hunger | -0.94580 |
| Annual checkups | -0.95430 |
| Trulicity Advise | -0.95640 |
| Getting back on track again after my fall injury | -0.96400 |
| Gastric Flu | -0.96450 |
| Feeling Drained Again | -0.96490 |
| Drink up girls | -0.97070 |
| DAPAGLIFLOZIN/METFORMIN | -0.97140 |
| Bad nights make me feel deathly | -0.97540 |
| high bg | -0.97840 |
| Ever have hours or days where you just don't care? | -0.98430 |
| Breathing troubles | -0.99260 |
| Alcohol and hypo..hypo and depression | -0.99560 |

Name: sentiment, dtype: float64

The need analysis

In type 1, we selected some comments that were out of the top 20 negative topics but still related to the needs. It is because of a higher percentage of type 1 patients with negative sentiment. Mayo Clinic (2022) suggests that diabetes type 1 patients face symptoms that tend to come on quickly and be more severe. It can be a reason why diabetes type 1 patients have worse sentiments.

Type 1 - top 20 negative topics

Mental problems

| Problem | Possible solution |
|---|---|
| Hypo causes irrational anger | - Patients' families or acquaintances may need more tolerance and comfort for their anger because it is uncontrollable sometimes. |
| The treatment and symptoms (complications) cause depression and worries. | - The patients can talk to the people who are possibly able to help them mitigate the difficulties or relieve their stress (e.g. family, friends, acquaintances, and doctors), meanwhile, these people can give some useful suggestions and encouragement to them. |

Symptoms / Complications

| Problem | Possible solution |
|---|---|
| Glandular fever causes difficulty swallowing, headaches, fever, and fatigue. | - Patients need to monitor their situation regularly.<br>- If the patients need help, the people near them can take care of them, even help them call an ambulance if they feel extremely unwell. |
| A pregnant woman had epilepsy.<br><br>(Dafoulas, Toulis, Mccorry, Kumarendran, Thomas, Willis, Gokhale, Gkoutos, Narendran, Nirantharakumar (2016) state patients with type 1 diabetes are at approximately three times greater risk of developing epilepsy compared with matched controls without type 1 diabetes.) | - If people see a patient with epilepsy, they can provide instant help (e.g. support the patient with his/her arm and help them get a seat first.)<br>- Patients' family/friends can connect with them regularly to know their latest condition, it might be extremely helpful for patients who live alone. |
| Some patients suffer from floaters, a complication of diabetes.<br>(Floaters: The abnormal blood vessels associated with diabetic retinopathy stimulate the growth of scar tissue, which can pull the retina away from the back of the eye.) | - The patients need to self-monitor their body status regularly. Also, when they discover that they have floaters, they must seek help as soon as possible because floaters can be a quickly deteriorated complication. |
| Diabetes creates a higher chance of Gastroparesis.<br>(Gastroparesis: the stomach does not work properly to digest food.) | - The diabetes patients need to find a doctor and discuss if there is any adjustment that can be made in their diet to deal with the Gastroparesis issue. |
| Diabetes gets bruised easily. | - Treat the affected area with ice.<br>- Do not inject insulin within an inch or two of the belly button.<br>Fallabel, C. (2022)<br>Their family can remind them to do this. |
| Patients find it difficult to keep enough weight on their bones due to autoimmune conditions (white cells destroy the pancreas's insulin-producing cells. This is a symptom of osteoporosis. | - Regular whole body checking can be done (e.g. Do body checking every year).<br>- Their families or friends may encourage them to live a healthy lifestyle (e.g. exercise regularly to strengthen their bones, don't |

| | |
|---|---|
| | smoke… etc.) |

Problems in treatment

| Problem | Possible solution |
|---|---|
| Diabetes doctors are not helpful.<br>(e.g. the patient is suffering from a water infection but the doctor just told him to drink more water to control his blood sugar level.) | - Society may provide some emergency information and reliable resources to diabetes patients to enable them to realize who / where they can get help from. |
| Insulin injection causes a high level of serum (injection). Therefore, tiredness occurs. | - If patients' families discover it, they can help the patient to find the doctor as soon as possible to make changes in the patient's treatment. |
| Misdiagnosed as type 2 caused wrong treatment. The medicine, Lantus, caused neuropathy pain in a patient's fingers and feet. | - The patients may need to seek treatment from one more doctor to ensure that the doctor's judgment is accurate. |
| Immunosuppressants, a medicine for type 1 diabetes increase the risk of infection. | - The patients can see the doctor regularly to decide whether they need to change anything in treatment. |
| Some diabetes patients need machines to help them maintain insulin levels in the body, so it is problematic if the machine does not work. | - The diabetes patient's family might buy one more machine as a backup if the financial status is allowed. It can ensure if the current insulin-supplying machine is shut down unluckily, one more machine still can be used. |

Other

| Problem | Possible solution |
|---|---|
| Type 1 patients need injections of insulin, and a meter, so they suffer from high living costs. Many expensive holiday activities cannot be afforded, and these problems lower life quality. | - These patients' families and friends may try to select low-cost activities to enable them to join.<br>- The society can offer financial help (e.g. enable the patients to purchase insulin injection-related equipment or blood sugar meters at a lower price.) |

Out of the top 20 negative topics

| Problem | Possible solution |
|---|---|

| | |
|---|---|
| The patients are not clear about the details of insulin injection.<br>(e.g. What time is most suitable for injection) | - Their doctors need to communicate with them regularly to find out what insulin injection time and which insulin injection brand is most suitable for them. |
| Heat causes unstable sugar levels. | - Patients can drink water regularly.<br>- The people who accompany the patients can remind them to drink water in hot/dry environments. |

Type 2 - top 20 negative topics

Symptoms/complications

| Problem | Possible solution |
|---|---|
| Diabetes type 2 may increase the risk of gallstones, it makes patients suffer. | - If the patients' friends/families find them suffering from gallstones, the fastest way is to do a heated compress for him/her. (Anthony, K.,2019) If the friends/families find it does not work, then they need to take the patients to the hospital.<br><br>Heated compress: use a towel with warm water and apply it to the affected area for 10 to 15 minutes. |
| Some patients suffered from vertigo, or tiredness so they felt dizzy.<br>(Type 2 diabetes can cause low or high blood sugar and dehydration (Sissons, B.,2021)) | - Getting plenty of rest may help. So if the diabetes patient's family has good financial status, they can let the patient not need to do too long-working hours to earn money. |
| High blood glucose, which is called hyperglycemia, or too little glucose, which is called hypoglycemia affects lung function, so it is hard to breathe (P. Assid,2022). | - Diabetes patients can open windows or get outside to breathe the fresh air to increase blood oxygen levels and prevent difficulty in breathing. The families or people who work with diabetes can provide a good ventilation environment for them. |

Mental problems

| Problem | Possible solution |
|---|---|
| Hard to lose body weight quickly, so some people with diabetes feel very stressed.<br>( This is a difficult task for many people who are initially diagnosed with type 2 diabetes.) | - Family and friends can encourage and support the patients to maintain a healthy lifestyle while they are losing weight.<br>- Diabetes patients can change their diet to a low-carb diet gradually instead of instantly to adapt to the |

| | |
|---|---|
| | new change in lifestyle. After the lifestyle is changed, it is important to keep it since diabetes is a chronic disease. |
| Some diabetes patients feel depressed due to diabetic issues and other unlucky issues in daily life. | - Patients' friends might need to give more encouragement, comfort, and meaningful suggestions to them to help them overcome hard times. |

The side effect of medicines

| Problem | Possible solution |
|---|---|
| There are worrying side effects of medicines.<br><br>The drug, Avandia (a drug for diabetes type 2 adults) is still used. It lowers blood sugar but causes a 30% higher risk of heart attack. Hence, it becomes a worrying issue in their treatment.<br><br>Metformin is a medicine that lowers blood sugar levels. However, NHS website (2022) suggests the common side effects are feeling sick (nausea), being sick (vomiting), diarrhea, stomach ache, loss of appetite, and a metallic taste in the mouth.<br><br>Dapagliflozin is a medicine that works with kidneys to prevent the absorption of blood sugar. (Maya-clinic). However, its side effect is thrush, back pain, peeing more than normal, feeling dizzy, and mild skin rash. | - The doctors must inform the patients of the function and side effects of medicine clearly.<br>- The doctors must communicate with their patients well to ensure that the patients are taking the treatment and medicine that they really want.<br>- If the patients feel unwell after taking medicines, they must seek help or advice immediately.<br>- The patients must make sure they take the right amount of medicine each time.<br>- The side effects may affect the working efficiency of the patients. Hence, they need more tolerance from others in their daily life. |

Other

| Problem | Possible solution |
|---|---|
| The patients do not know how to keep a healthy diet and way to regular exercise. | - The family and friends may provide some ideas to them if they are more familiar with diabetes issues.<br>- Society can provide more free healthcare consultations to them. It ensures that diabetes can seek help from reliable professionals. |

# *IX. Discussion*

## a. Limitation

- Many different kinds of data are difficult to access.
  The patient's blood sugar levels, and hand-written notes of their conditions are very difficult to gain. If we have these data, perhaps the problems of diabetes can be reflected more comprehensively.

## b. Challenge in project

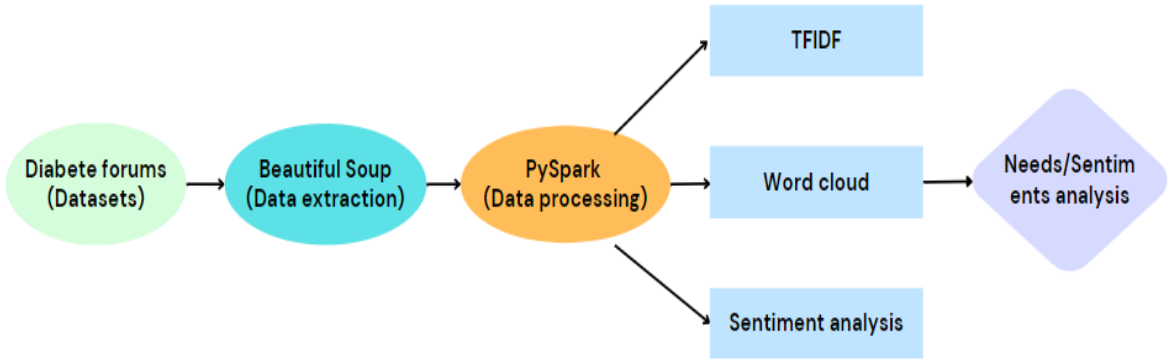
- RAM issue
  There is too much data, so if the RAM in python runs too long, the RAM and kernel shut off suddenly. The RAM ran out 2 times in the extraction.
  Mitigation: divide one extraction into several extractions. (To release the memory in RAM after the first extraction was finished.)

## c. Further suggestion

- Extend the data sources
  In this report, we just used two diabetes discussion forums as our dataset. The analysis may not be accurate as the data samples are relatively small. Besides, the needs and sentiment analysis can also be concluded from different data sources such as surveys, medical reports, journal articles. Extending the data sources could be a huge improvement in understanding their needs and sentiments.

- Using text classifiers
  If using more data sources for analysis, the increase in scalability may require classifiers or learning models. The model can extract the feature and train with the data samples to get the prediction on topics. Therefore, we can quickly understand what are the most frequent topics and focus on these topics for further analysis.

# X. ***Conclusion***




To conclude, we have conducted our project in the following approach. First, we have selected 2 diabete forums as our datasets. Then, we use different libraries such as request and beautiful soup to get the access and extract different elements from the forums such as page title, page content, number of pages to compile the csv files. PySpark is also applied to process data and standardize the csv files with a maximum of 10000 rows in parallel. The data sources are finally formatted from unstructured textual data to structured ones. Next, we use 3 methods to analyze the textual data, which includes TFIDF, wordcloud, and sentiment analysis. Finally, we can conduct the needs and sentiments analysis based on the results of the methods mentioned. Below gives a detailed description of our findings.

Result analysis of type1 diabetes:

| Needs | Sentiments |
|---|---|
| • insulin and hypo are their major concerns<br>• need both emotional and physical support from friends and family members | • major sentiments are relatively neutral to negative |

We analyzed that the major concerns of type1 diabetic patients are insulin and hypo. From TFIDF analysis, insulin ranks the first in term frequency, and in the wordcloud and sentiment analysis, the word pump (insulin injection) and hypo appears in both type1 wordclouds and the topics of sentiment analysis respectively. They also need more assistance from the surroundings, given the word 'help' appears in the type1 wordcloud and high medication cost analyzed in the sentiment analysis.

The sentiment of type1 diabetic patients are mostly neutral, but it is relatively worse than type2. In sentiment analysis, although the proportion of positive sentiment in type1 is slightly higher than type2, the proportion of negative sentiment is also higher. Besides, they

tend to suffer more severe symptoms and emotional disorder from the result of sentiment analysis such as anger, depression and worries.

Result analysis of type2 diabetes:

| Needs | Sentiments |
|---|---|
| • blood sugar, weight and diet are their major concerns<br>• mainly needs to be self-motivated, and also the guidance from doctor and dietitian | • major sentiments are relatively neutral to positive |

For the needs analysis of type2 diabetes, we analyzed that the major concerns of type2 diabetic patients are blood sugar, weight and diet. From the wordcloud analysis, the frequent words appearing in type2 wordclouds are related to blood sugar and eating habits. However, unlike type1 diabetes that requires external support and their remedy could be life-long, type2 diabetic patients only need to be self-motivated for regular exercise and healthy diet, and seek the help from doctors and dietitian, given the word 'doctor' appears in type2 wordcloud and also their questions in maintaining healthy diet and regular exercise analyzed in sentiment analysis.

The sentiments are mostly neutral, but it is more positive than type1 diabetes. In wordcloud, there are more positive words appearing in the type2 wordclouds such as well done and believe, which means they are less worried about their symptoms and they tend to encourage each other. Besides, they don't suffer much mental problems and their diabetes symptoms are less severe than type1. This could be the reason why the sentiment of type2 diabetic patients are relatively more positive.

Anyway, we found no matter whether diabetes type 1 patients or diabetes type 2 patients, they face some diabetes-related problems in real life. However, type 2 can be preventable by a healthy diet, regular exercise, and good living habits. Therefore, prevention is always better than curation. Let's try to maintain a healthier lifestyle and give more care, love, and tolerance to diabetes patients.

# *XI. Job Distribution*

| Student name | Job |
|---|---|
| CHAN Kiu Nga, Sandy | Code:<br>Data extraction, sentiment analysis, data visualization |

| | |
|---|---|
| | Powerpoint<br><br>Report:<br>Introduction, motivation, objective, data sources, sentiment analysis in methodology and, result and analysis, limitation and challenge in project in discussion. |
| CHAN Tsz Ngai, Matthew | Code:<br>Code debug, wordcloud, data visualization<br><br>Powerpoint<br><br>Report:<br>Abstract, Wordcloud in methodology and result and analysis, Discussion, Conclusion |
| Ma Wing Kin, Paul | Report:<br>Data processing (Spark/environment/methods/result), LDA (methods/visualization/result)<br><br>Powerpoint<br><br>Code:<br>LDA, data processing, apply Spark for parallel computing, data visualization |
| NG Sin Yung, Cindy | Code:<br>Data extraction, TFIDF, data visualization<br><br>Powerpoint<br><br>Report:<br>TFIDF in methodology and, result and analysis. |

# *XII. Reference (A-Z)*


Allem, R. & Laissaoui, A. (2016). The eating habits of type 2 diabetics in the region of Ain-Defla (Algeria). *Pakistan Journal of Medical Sciences*, *32*(2). https://doi.org/10.12669/pjms.322.9266

Anthony, K. (2019, October 9). Relieving Gallbladder Pain Naturally. Healthline. https://www.healthline.com/health/gallbladder-pain-relief

BBC News. (2015, September 17). "Youngest" toddler with type 2 diabetes raises concern. https://www.bbc.com/news/health-34259221

Complications of diabetes. (n.d.). Diabetes UK. Retrieved November 22, 2022, from https://www.diabetes.org.uk/guide-to-diabetes/complications

Dapagliflozin (Oral Route) Side Effects - Mayo Clinic. (n.d.). https://www.mayoclinic.org/drugs-supplements/dapagliflozin-oral-route/side-effects/drg-20095101?p=1

D. Brennan. (2021, April 22). How to Increase Your Blood Oxygen Level. WebMD. https://www.webmd.com/fitness-exercise/how-to-increase-blood-oxygen-level

Diabetes - Symptoms and causes. (2022, October 25). Mayo Clinic. https://www.mayoclinic.org/diseases-conditions/diabetes/symptoms-causes/syc-20371444

Fallabel, C. (2022, August 7). How to Avoid Injection Site Bruising. Diabetes Strong. https://diabetesstrong.com/how-to-avoid-injection-site-bruising/

George E. Dafoulas, Konstantinos A. Toulis, Dougall Mccorry, Balachadran Kumarendran, G. Neil Thomas, Brian H. Willis, Krishna Gokhale, George Gkoutos, Parth Narendran, Krishnarajah Nirantharakumar.Type 1 diabetes mellitus and risk of incident epilepsy: a population-based, open-cohort study. (2016, October 31). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6518067/

Diabetes.co.uk (n.d.) Retrieved November 1, 2022 https://www.diabetes.co.uk/

Diabetes Daily - Diabetes Daily Forums (n.d.) . Retrieved November 1, 2022 https://www.diabetesdaily.com/forum/

P. Assid. (2022, August 23). Diabetes and Shortness of Breath: What's the Connection? Verywell Health. https://www.verywellhealth.com/diabetes-and-shortness-of-breath-5114863

NHS website. (2022, July 22). Side effects of metformin. nhs.uk. https://www.nhs.uk/medicines/metformin/side-effects-of-metformin/

Sissons, B. (2021, October 11). Type 2 diabetes and dizziness. https://www.medicalnewstoday.com/articles/type-2-diabetes-dizziness

Type 2 Diabetes. (2022, March 2). Centers for Disease Control and Prevention. https://www.cdc.gov/diabetes/basics/type2.html

Welcome to VaderSentiment's documentation! — VaderSentiment 3.3.1 documentation. (n.d.). https://vadersentiment.readthedocs.io/en/latest/index.html

What Is Type 1 Diabetes? (2022, March 11). Centers for Disease Control and Prevention. https://www.cdc.gov/diabetes/basics/what-is-type-1-diabetes.html

What People With Diabetes Need To Know About Osteoporosis | NIH Osteoporosis and Related Bone Diseases National Resource Center. (2018, November 1). https://www.bones.nih.gov/health-info/bone/osteoporosis/conditions-behaviors/diabetes

World Health Organization. (2022, September 16). *Diabetes (Newsroom)*. Retrieved November 1, 2022, from https://www.who.int/news-room/fact-sheets/detail/diabetes