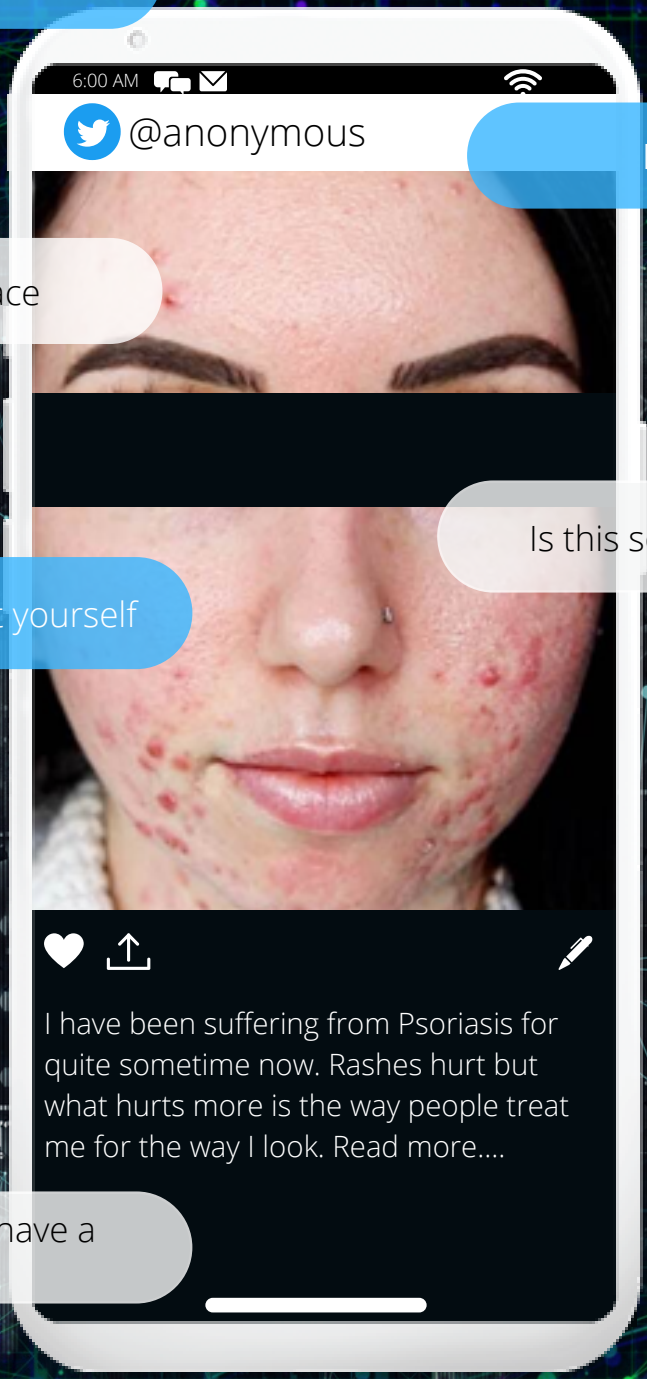
PSORIASIS

A story that grows from the inside

What the f**k is wrong with your face



Do you have AIDS

You should cover your face

Is this some kind of CANCER !!!

How do you even look at yourself

You should be in a circus.

Is this an after effect of COVID -19.

Eat healthy and you might have a better face

Does this spread on contact !!!

Reshma Dua Tanya Ranjan Rishab Sarkar

Psoriasis

A story that grows from the inside.

Abstract

Microblogging has become a useful tool to share everyday life events and news and to express opinions. As Twitter posts are short and frequently generated, they are a great source for providing public sentiments towards a specific topic which can be related to politics, healthcare or technological innovation. The use of Sentiment Analysis in healthcare can assist in understanding how individuals discuss and feel about themselves or people around them. The aim of our project is to analyse and understand the general sentiments of people around Psoriasis, the key topics people discuss positively and negatively about and the most popular Psoriasis drug companies.

Acknowledgement

On the very outset of this report, we would like to express our sincere gratitude to Ms. Jayati Kaushik and Ms. Upasana Parashar for providing us the opportunity to work on "*Analysing the general sentiments of people around Psoriasis*". Thank you for your constant support, guidance and encouragement.

We would also like to thank Dr Kulwant Singh for helping us out in understanding the symptoms, diagnosis and precautions taken by the patients suffering from Psoriasis along with the day-to-day challenges they face. Discussions with him gave us perspective to carry out our analysis.

Chapter 1: Unravelling the truth

"Don't judge a book by its cover" is a common metaphorical phrase that gets tossed around every now and then to avoid forming opinions about people just by the way they look. Is it true, though? Don't we primarily judge people just by their looks? Yes, most of us do. It is a harsh reality, but we still objectify people.

Things become much easier and simpler when we look good. Be it the film industry or a corporate people with good looks get better and frequent opportunities. As per the standards set by our society a person with not so good looks, will have to prove his/her worth. So, the base point is that we get judged everywhere, every time.

We can't change the fact that people judge other people just by their looks. But what we can change is the way we judge them or get judged. The first thing that comes to our mind when we talk about looks is the facial appearance of a person along with their skin colour, features and so on.

Fig 1: Advertisement poster expressing the need to be fair to look handsome

There are several companies making huge profits out of the fact that looks matter and people need beauty products to look better and live up to the standards set by the society. What they don't understand other than making profits is that all this will turn out to be a ripple effect. Portraying fair men as handsome changes the outlook of people and ignites the process of judging people on the basis of their skin colour. This is just one of the many examples out there, there have been cases were celebrities had to go over a complete make over / transformation to get acknowledged. This act did get them recognised for a while but also ended up taking their lives due to the harsh chemicals and painful treatments with side effects.

Brands selling skin care products prove that their product is affective and come up with various ads which in turn affects the way consumers think about themselves and often results in lowering their self-esteem. This creates a mental blockage, pointing to the fact that they are not good enough. Well ads have had their fair share to establish what good looks look like but with the internet and social media being easily accessible to all, the problem has grown exponentially and taken the form of cyber bulling.

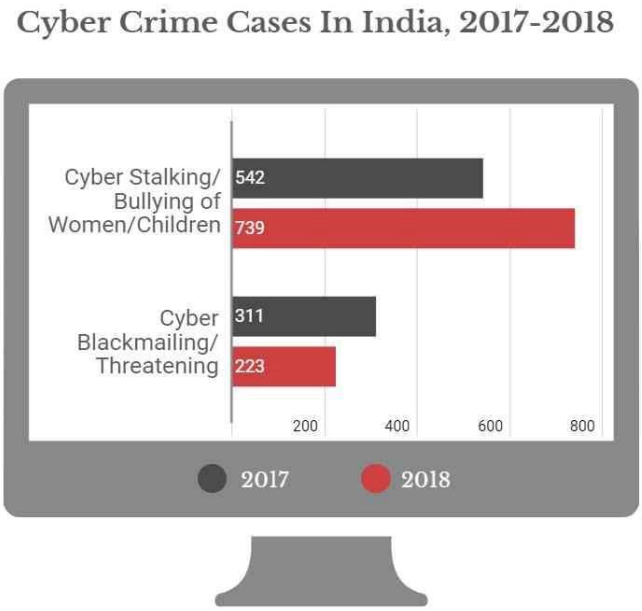


Fig 2: Graphical representation of Cyber Bulling and other Cyber Crimes (2017 and 2018)

People always have had the freedom of speech but there have been instances where opinions have been expressed or comments have been passed in a way we wouldn't expect. The best example of what people say and the impact it creates can be portrayed by the mishap that happened at Microsoft, with their AI Twitter bot. Microsoft's attempt at engaging millennials with artificial intelligence through its bot TAY had backfired hours into its launch, with Twitter users teaching its chatbot how to be racist. Tay was designed to engage and entertain people where they connect with each other online through casual and playful conversation but it appeared that Tay's conversation extended to racist, inflammatory and political statements.

Tay in most cases was only repeating other users' inflammatory statements, but the nature of AI means that it learns from those interactions. It's therefore somewhat surprising that Microsoft didn't factor in the Twitter community's fondness for hijacking brands' well-meaning attempts at engagement when writing Tay.

Fig 3: An Illustration showing a social media user getting trolled for having acne

With the rise of social media came the rise of the social media influencers. Their content on "*how do you look beautiful', "how your skin should look and feel."*, etc. has made the situation worse.

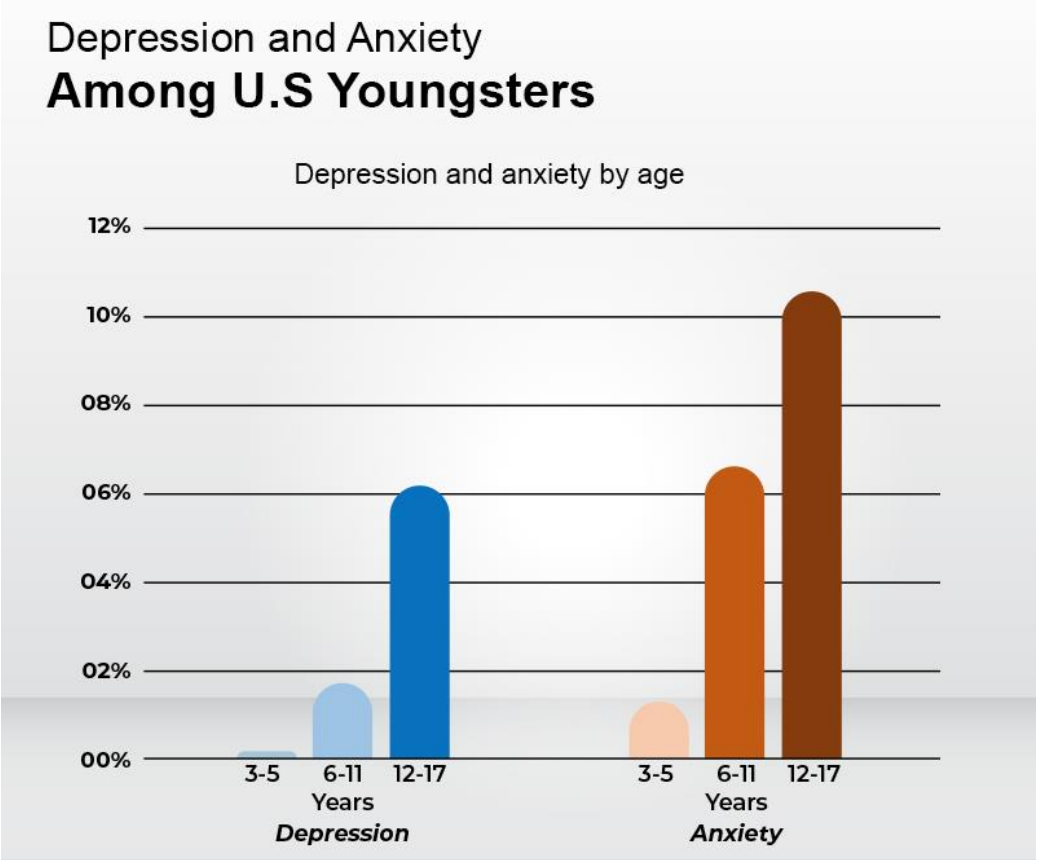


Fig 4: Graphical representation of U.S. youngsters suffering from Depression and Anxiety

The depression and anxiety rate in youngsters have never been this high in years. The main reason behind this is suspected to be cyber bulling and imposter syndrome *(is a psychological pattern in which an individual doubts their skills, talents, or accomplishments and has a persistent internalized fear of being exposed as a "fraud".)*

Chapter 2: Behind the second skin

"The most common fact is the most unknown fact." There are various skin diseases that people have for various reasons. One of the common ones is *'Psoriasis'*, "*Psoriasis is a skin disease that causes red, itchy scaly patches, most commonly on the knees, elbows, trunk and scalp."* According to doctors and experts, it is a common, long-term (chronic) disease with no cure [1]. It tends to go through cycles, flaring for a few weeks or months, then subsiding for a while or going into remission. Treatments are available to help manage symptoms, incorporate lifestyle habits and coping strategies to live better with psoriasis.

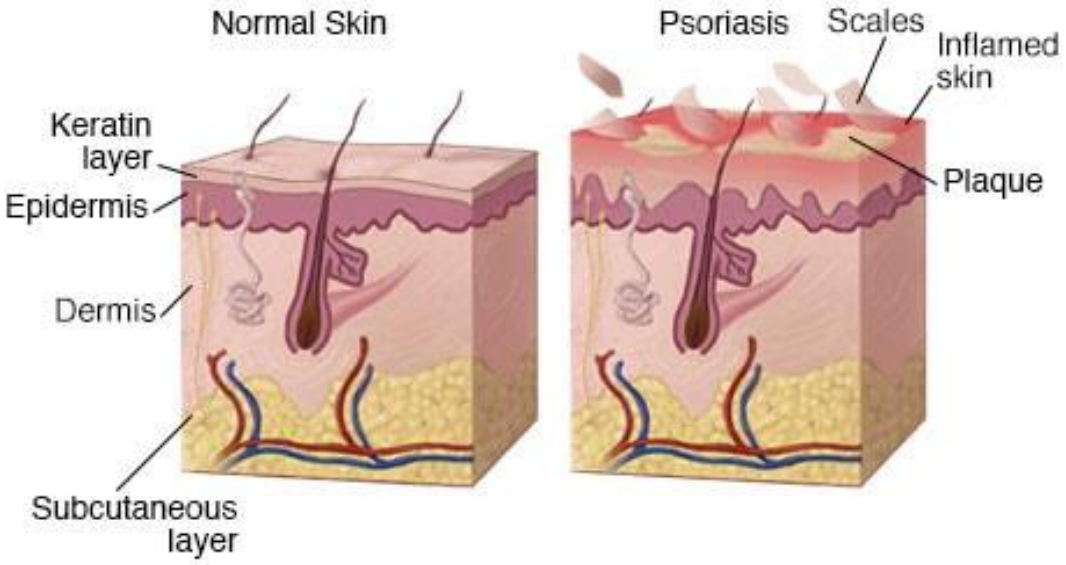


Fig 5: Normal skin Vs Psoriasis under the microscope

Psoriasis signs and symptoms can vary from person to person. Common signs and symptoms include:

- Red patches of skin covered with thick, silvery scales
- Small scaling spots (commonly seen in children)
- Dry, cracked skin that may bleed or itch
- Itching, burning or soreness
- Thickened, pitted or ridged nails
- Swollen and stiff joints

The patches formed through Psoriasis can range from a few spots, like pimples with dandruff like scaling to a major eruption that covers large areas. The most commonly affected areas are- lower back, elbows, knees, legs, soles of the feet, scalp, face and palms. Most types of psoriasis go through cycles, flaring for a few weeks or months, then subsiding for a time or even going into remission.

Fig 6: This picture shows the 2 types of patches as discussed above

There are several types of Psoriasis:

- *Plaque psoriasis*. The most common form, plaque psoriasis causes dry, raised, red skin patches (lesions) covered with silvery scales. The plaques might be itchy or tender, and there may be few or many. They usually appear on elbows, knees, lower back and scalp.
- *Nail psoriasis*. Psoriasis can affect fingernails and toenails, causing pitting, abnormal nail growth and discoloration. Psoriatic nails might loosen and separate from the nail bed (onycholysis). Severe cases may cause the nail to crumble.
- *Guttate psoriasis*. This type primarily affects young adults and children. It's usually triggered by a bacterial infection such as strep throat. It's marked by small, drop-shaped, scaling lesions on the trunk, arms or legs.
- *Inverse psoriasis*. This mainly affects the skin folds of the groin, buttocks and breasts. Inverse psoriasis causes smooth patches of red skin that worsen with friction and sweating. Fungal infections may trigger this type of psoriasis.
- *Pustular psoriasis*. This rare form of psoriasis causes clearly defined pus-filled lesions that occur in widespread patches (generalized pustular psoriasis) or in smaller areas on the palms of the hands or the soles of the feet.
- *Erythrodermic psoriasis*. The least common type of psoriasis, erythrodermic psoriasis can cover your entire body with a red, peeling rash that can itch or burn intensely.
- *Psoriatic arthritis*. Psoriatic arthritis causes swollen, painful joints that are typical of arthritis. Sometimes the joint symptoms are the first or only symptom or sign of psoriasis and at times only nail changes are seen. Symptoms range from mild to severe, and psoriatic arthritis can affect any joint. It can cause stiffness and progressive joint damage that in the most serious cases may lead to permanent joint damage.

People might suffer from one or more of the above-mentioned types. Considering the level of severity, affected people should consult a doctor and get some treatment and/or medication to reduce the effect of the disease. Patients suffering from psoriasis constantly

fear of getting judged or trolled by others for looking the way they look after Psoriasis. There are people who pull their kids away from people who suffer from this disease and they just don't stop there, they go on to add silly comments like *"you shouldn't eat too much of chocolate"* or "*if you don't eat your greens you are going to look like him/ her*."

We all know what is true/right and what is not but to what extent? To understand how true the facts are we can use the same social media data that is generated by people on a regular basis and try and understand the general sentiment of the people for a given topic, in our case '*Psoriasis'*.

Chapter 3: Literature Review

With the advancement of web technology and its growth, there is a huge volume of data present on the web for internet users and a lot of data is generated too. Internet has become a platform for online learning, exchanging ideas and sharing opinions. Social networking sites like Twitter, Facebook, Google+ are rapidly gaining popularity as they allow people to share and express their views about topics, have discussion with different communities, or post messages across the world. There has been a lot of work done in the field of sentiment analysis of twitter data.

The paper on *"Sentiment Analysis of Twitter Data: A Survey of Techniques"* focuses mainly on sentiment analysis of twitter data which is helpful to analyse the information in the tweets where opinions are highly unstructured, heterogeneous and are either positive or negative, or neutral in some cases. A comparative analysis of existing techniques for opinion mining like machine learning and lexicon-based approaches, together with evaluation metrics is done. Using various machine learning algorithms like Naive Bayes, Max Entropy, and Support Vector Machine, research on twitter data streams is provided and also general challenges and applications of Sentiment Analysis on Twitter are discussed in brief [2].

A General model for sentiment analysis is as follows -

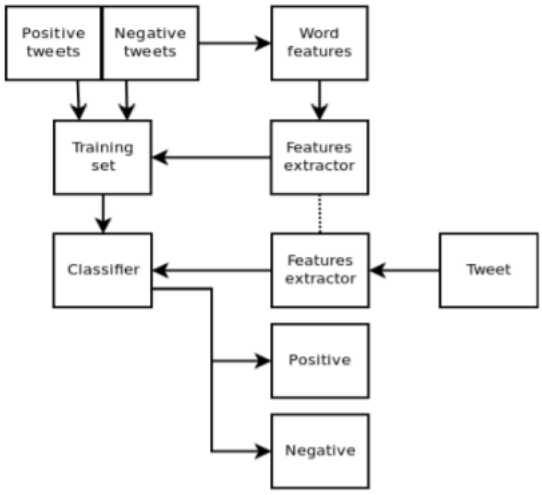


Fig 7: Sentiment Analysis Architecture

Phases required for sentiment analysis of twitter data,

(I) Pre-processing of the datasets

- Remove all URLs (e.g., www.xyz.com), hash tags (e.g., #topic), targets (@username)
- Correct the spellings; sequence of repeated characters is to be handled
- Replace all the emoticons with their sentiment.
- Remove all punctuations, symbols, numbers
- Remove Stop Words
- Expand Acronyms (we can use an acronym dictionary)
- Remove Non-English Tweets

(II) Feature Extraction

(III) Training

(IV) Classification

- Naive Bayes
- Maximum Entropy
- Support Vector Machine

The paper on *"A Review on Sentiment Analysis Approaches"* elaborates on the two approaches widely used for sentiment analysis – *Machine learning based and Lexicon based*.

The machine learning based approach uses various supervised and unsupervised learning algorithms for sentiment classification. While the Lexicon based techniques use a lexicon dictionary with sentiment words related to specific domain for sentiment classification. The dictionary contains the information about the polarity of each word, whether the words are positive or negative. The word in the sentence can be matched with the words in dictionary to determine their polarity [3]. Some researchers combined the machine learning and lexicon-based techniques [4,5]. The various sentiment analysis techniques are categorized as shown below –

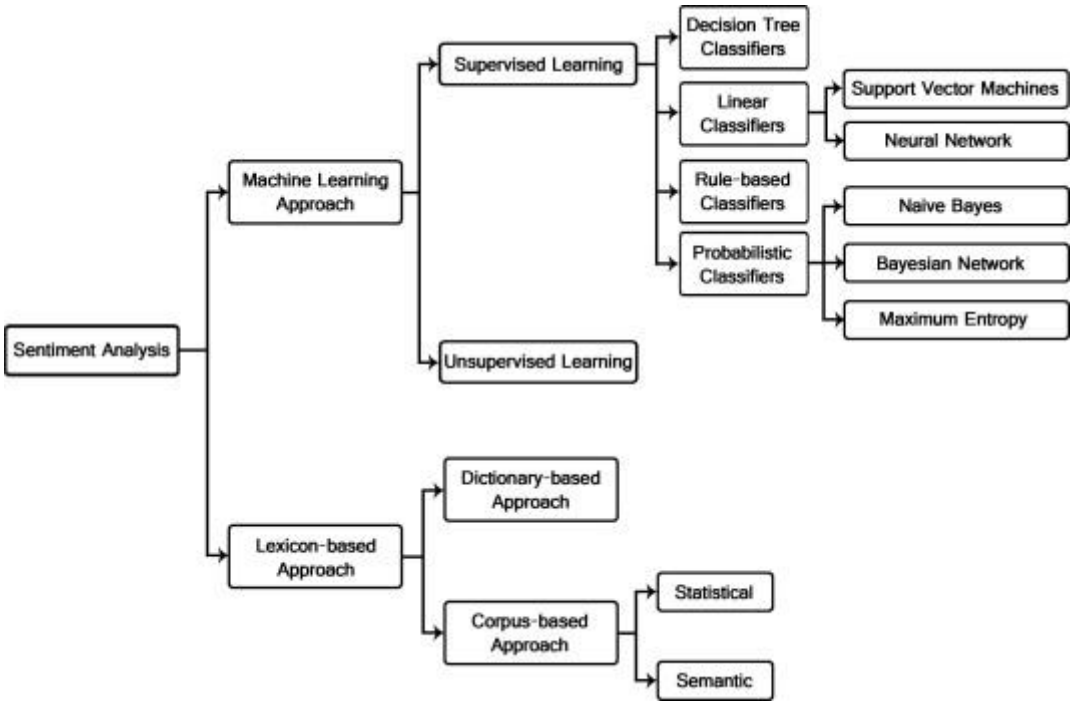


Fig 8. Sentiment Analysis Techniques

Comparison of ML and Lexicon Based Techniques –

	Machine Learning	**Lexicon based**
Approach	Supervised/Unsupervised	Unsupervised
Technique used	Naïve Bayes, Maximum Entropy, Support Vector Machine	PMI Dictionary based Approach
Data Processing	Requires prior training of data	Does not require prior training of data
Factors Affecting Performance	-Selected features -Quality and quantity of training data -Domain of the dataset	-Opinion words included in the dictionary -Polarity of many words in domain context
Accuracy	More	Less

The theoretical study concludes that machine learning (Supervised) techniques have performed better in terms of accuracy than lexicon based (Unsupervised) methods in sentiment analysis. However, the performance of machine learning approaches is heavily dependent on the selected features, quality and quantity of training data and the domain of the dataset. The additional training time required by the machine learning approach is also a prominent issue; as lexicon-based approach do not require time for training. Therefore, the combined approach of machine learning and lexicon-based techniques should be used to avoid the weaknesses of both the techniques and to preserve their strength. The combined approach of machine learning and lexicon-based techniques results in best accuracy of the sentiment classification as compared to the machine learning and lexicon-based approaches. It is inheriting high accuracy from machine learning approach and achieving stability from lexicon-based approach to enhance the performance of sentiment classification.

Our world today is undeniably digital and social media now offers a new data source of user-generated content. Researchers suggest that the social network Twitter may serve as a rich avenue for exploring how patients communicate about their health issues. Psoriasis, as we know is an autoimmune disease posing a significant public health problem and contributing to rising health care costs, affecting people's quality of life and ability to work.

Previous research showed that nontreatment and undertreatment of patients with psoriasis remained a significant problem. Perspectives of patients toward seeking psoriasis treatment are understudied. Hence, *Katja Reuter* and *Delphine Lee leverage* social media to examine perspectives that potentially influence the treatment decision among patients with psoriasis in their paper "*Perspectives Toward Seeking Treatment Among Patients with Psoriasis: Protocol for a Twitter Content Analysis*".

In this study, supported by the "*National Center for Advancing Translational Science"* user-generated Twitter posts that include keywords related to psoriasis are analysed using text classifiers to identify themes related to the research questions. Symplur Signals, a health care social media analytics platform is used to access the Twitter data. Descriptive statistics is used to analyse the data and identify the most prevalent topics in the Twitter content among people with psoriasis [6].

The findings shed light on whether Twitter provides a promising data source for collecting patient perspective data about psoriasis treatment decisions and if the data helps to determine whether Twitter might serve as a potential outreach platform for raising awareness of psoriasis and treatment options among patients and implementing related health interventions.

References

[1] www.mayoclinic.org

[2] arXiv:1601.06971 [cs.CL]

[3] Patil, Ashwini & Gupta, Shiwani. (2021). A Review on Sentiment Analysis Approaches.

[4] Anuj Sharma, Shubhamoy Dey "Artificial Neural Network Based Approach for Sentiment Analysis of Opinionated Text", ACM transaction, 2012

[5] Tsytsarau Mikalai, Palpanas Themis. "Survey on mining subjective data on the web". Data Mining Knowledge Discovery 2012; 24:478–514.

[6] Reuter K, Lee D. Perspectives Toward Seeking Treatment Among Patients with Psoriasis: Protocol for a Twitter Content Analysis. JMIR Res Protoc 2021;10(2):e13731

URL: https://www.researchprotocols.org/2021/2/e13731, DOI: 10.2196/13731

Scrapping Tweets on Psoriasis

1. Installing and loading necessary packages

```
# Installing Packages
!pip install aiohttp==3.7.0
```

```
!git clone --depth=1 https://github.com/twintproject/twint.git
!cd /content/twint && pip3 install . -r requirements.txt
import twint
import nest_asyncio
nest_asyncio.apply()
import time
import pandas as pd
import os
import re
import datetime
```

```
Requirement already satisfied: aiohttp in /usr/local/lib/python3.7/dist-packages
Collecting aiodns
  Downloading aiodns-3.0.0-py3-none-any.whl (5.0 kB)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-pa
Collecting cchardet
  Downloading cchardet-2.1.7-cp37-cp37m-manylinux2010_x86_64.whl (263 kB)
     |████████████████████████████████| 263 kB 2.2 MB/s
Collecting dataclasses
  Downloading dataclasses-0.6-py3-none-any.whl (14 kB)
Collecting elasticsearch
  Downloading elasticsearch-7.16.1-py2.py3-none-any.whl (385 kB)
     |████████████████████████████████| 385 kB 40.7 MB/s
Requirement already satisfied: pysocks in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pandas>=0.23.0 in /usr/local/lib/python3.7/dist-pa
Collecting aiohttp_socks<=0.4.1
  Downloading aiohttp_socks-0.4.1-py3-none-any.whl (17 kB)
Collecting schedule
  Downloading schedule-1.1.0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: geopy in /usr/local/lib/python3.7/dist-packages (f
Collecting fake-useragent
  Downloading fake-useragent-0.1.11.tar.gz (13 kB)
Collecting googletransx
  Downloading googletransx-2.4.2.tar.gz (13 kB)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-pack
```

```
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: attrs>=19.2.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: async-timeout<4.0,>=3.0 in /usr/local/lib/python3.
Requirement already satisfied: chardet<4.0,>=2.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.7/di
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.7/dist-package
```

```
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/lib/python3
Collecting pycares>=4.0.0
  Downloading pycares-4.1.2-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.many
     |████████████████████████████████| 291 kB 39.8 MB/s
Requirement already satisfied: cffi>=1.5.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: urllib3<2,>=1.21.1 in /usr/local/lib/python3.7/dis
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: geographiclib<2,>=1.49 in /usr/local/lib/python3.7
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages
Building wheels for collected packages: twint, fake-useragent, googletransx
  Building wheel for twint (setup.py) ... done
  Created wheel for twint: filename=twint-2.1.21-py3-none-any.whl size=38870 sha2
  Stored in directory: /tmp/pip-ephem-wheel-cache-3js4xnaw/wheels/f7/3e/11/2803f3
  Building wheel for fake-useragent (setup.py) ... done
  Created wheel for fake-useragent: filename=fake_useragent-0.1.11-py3-none-any.w
  Stored in directory: /root/.cache/pip/wheels/ed/f7/62/50ab6c9a0b5567267ab76a9da
  Building wheel for googletransx (setup.py) ... done
  Created wheel for googletransx: filename=googletransx-2.4.2-py3-none-any.whl si
  Stored in directory: /root/.cache/pip/wheels/66/d5/b1/31104b338f7fd45aa8f7d2258
Successfully built twint fake-useragent googletransx
Installing collected packages: pycares, schedule, googletransx, fake-useragent, e
Successfully installed aiodns-3.0.0 aiohttp-socks-0.4.1 cchardet-2.1.7 dataclasse
```

2. Extracting all tweets on Psoriasis for last 4-years(Approx)

```
hashtag = ["Psoriasis"]
Limit = 100000
last_24_date_time = datetime.datetime.now() - datetime.timedelta(days = 1500)
today = last_24_date_time.strftime('%Y-%m-%d')
date = str(today)
c = twint.Config()
c.Lang = "en"
c.Search = hashtag
c.Limit = Limit
c.Since = date
c.Store_csv = True
c.Pandas = True
c.Pandas_clean = True
#c.Min_likes = 20
c.Output = "Psoriasis_3.csv"
twint.run.Search(c)
# data = twint.storage.panda.Tweets_df

#append_data.append(data)

#print('Searching: ' + hashtag)

time.sleep(10)
```

```
1473052396154769409 2021-12-20 22:07:13 +0000 <sirimanthesh> U.S. FDA approves
1473052296535678978 2021-12-20 22:06:50 +0000 <ZyiteGadgets> U.S. FDA approves ex
1473051843282542595 2021-12-20 22:05:02 +0000 <Oukuyk> @Eddie_Rado Is IQVIA that
1473050768903135252 2021-12-20 22:00:45 +0000 <prfinestcanna> CBG and other canna
1473049383487025154 2021-12-20 21:55:15 +0000 <tryingBot05> FDA APPROVES OTEZLA®
```

```
1473047822052122624 2021-12-20 21:49:03 +0000 <CrweWorld> FDA APPROVES OTEZLA® (A
1473047373685444616 2021-12-20 21:47:16 +0000 <cmencke> AwesomeCapital: FDA appro
1473047180009218049 2021-12-20 21:46:30 +0000 <MDMagazine> BREAKING NEWS: #FDA ap
1473045137207005185 2021-12-20 21:38:23 +0000 <newsfilterio> FDA approves expande
1473044904184057864 2021-12-20 21:37:27 +0000 <Benito66123304> Who needs bs from
1473043946582331395 2021-12-20 21:33:39 +0000 <julietaz__> venía muy suertuda has
1473043774272163846 2021-12-20 21:32:58 +0000 <ZonaMediatica> Psoriasis: el probl
1473043447166627844 2021-12-20 21:31:40 +0000 <Amgen> Today, the @US_FDA has appr
1473042783120154628 2021-12-20 21:29:01 +0000 <TWANewsOfficial> $AMGN Amgen annou
1473042639947812875 2021-12-20 21:28:27 +0000 <FlashAlert_me> $AMGN [15s. delayed
1473042317334523905 2021-12-20 21:27:10 +0000 <wwwmoneynet> $AMGN: FDA APPROVES O
1473042207183622145 2021-12-20 21:26:44 +0000 <TheTradeXchange> $AMGN: FDA APPROV
1473042122852950016 2021-12-20 21:26:24 +0000 <StockNewsBot2> $AMGN FDA APPROVES
1473041933891162112 2021-12-20 21:25:39 +0000 <stock_titan> $AMGN FDA APPROVES OT
1473041871417008129 2021-12-20 21:25:24 +0000 <newsfilterio> FDA APPROVES OTEZLA®
1473041859903516673 2021-12-20 21:25:21 +0000 <TWANewsOfficial> $AMGN FDA APPROVE
1473041807130849280 2021-12-20 21:25:09 +0000 <WebChronicleTo1> $AMGN FDA APPROVE
1473039972852662272 2021-12-20 21:17:51 +0000 <Ologies> Heard an ad for a cosmeti
1473037275386855428 2021-12-20 21:07:08 +0000 <CotaSkincare> Deal with the harsh
1473035614429736964 2021-12-20 21:00:32 +0000 <mooicide> @byderekarnell would abs
1473034745739104257 2021-12-20 20:57:05 +0000 <Postematico> Psoriasis: el problem
1473031633334792195 2021-12-20 20:44:43 +0000 <5dragons> Any doctors out there wh
1473027928455880708 2021-12-20 20:30:00 +0000 <AJMC_Journal> With the year coming
1473026839329361927 2021-12-20 20:25:40 +0000 <JoanMatsui> Do you suffer from pso
1473025875062890498 2021-12-20 20:21:50 +0000 <NoAguacatito> Como cuando alguien
1473024139141255171 2021-12-20 20:14:56 +0000 <dayecarter> @lifeinIight If it’s r
1473024089388425229 2021-12-20 20:14:45 +0000 <GiraInformativa> Psoriasis: El sis
1473020789834371082 2021-12-20 20:01:38 +0000 <IpcPs> Find out why over 12,000 #d
1473020342398603274 2021-12-20 19:59:51 +0000 <BiPolarguy1274> 1.)Ok all day yest
1473019614934315012 2021-12-20 19:56:58 +0000 <sadiepuss> @POTUS @anet2111 My son
1473019535611645957 2021-12-20 19:56:39 +0000 <MIRificaremos> 💊PÍLDORA💊 📌Psc
1473017290522247171 2021-12-20 19:47:44 +0000 <PsoriasisRmdy> Cure Psoriasis With
1473017068018622476 2021-12-20 19:46:51 +0000 <FundacionLaudet> https://t.co/uiq
1473016619613962244 2021-12-20 19:45:04 +0000 <PulmAdvisor> For patients with pso
1473016382912712706 2021-12-20 19:44:07 +0000 <ndeajanae> Lmao hoes mad they got
1473016186069733381 2021-12-20 19:43:20 +0000 <APAPPsoriasis> Feliz Navidad y Fel
1473015168569417736 2021-12-20 19:39:18 +0000 <Hiperenlazados> Psoriasis: El sist
1473014160275689472 2021-12-20 19:35:17 +0000 <jurwidyakop3> https://t.co/PnRVf8
1473013236350033926 2021-12-20 19:31:37 +0000 <Blew_Official> 🌸Eczema, 🌸Psoria
1473012908678434817 2021-12-20 19:30:19 +0000 <BeHealthPR> De acuerdo con la doct
1473012839703064580 2021-12-20 19:30:02 +0000 <ophth_advisor> Psoriasis can affec
1473012831800995852 2021-12-20 19:30:01 +0000 <DermTimesNow> A panel of experts r
1473012564208537604 2021-12-20 19:28:57 +0000 <TeActualizas> Psoriasis: el proble
1473012139900166144 2021-12-20 19:27:16 +0000 <tomgoom> @CombatSportPhys I now us
1473011568791150596 2021-12-20 19:24:59 +0000 <SlimPamGrier> Is yk psoriasis dumb
1473011028015390726 2021-12-20 19:22:50 +0000 <amixmus> psoriasis rlly got me on
1473010979478851585 2021-12-20 19:22:39 +0000 <DermaSolutiongh> For the treatment
1473009890041995273 2021-12-20 19:18:19 +0000 <oursoftskin> @niamholearyy I'm so
1473008907928981512 2021-12-20 19:14:25 +0000 <oursoftskin> @SilverbacKRW Congrat
1473008110969176069 2021-12-20 19:11:15 +0000 <MortarPestleUSA> Management of #Ps
1473006813822042112 2021-12-20 19:06:06 +0000 <raquelk00402202> .read ebook [pdf]
1473006704430358540 2021-12-20 19:05:40 +0000 <DundeeDerm> Don't let your #Psoria
```

.

3. Reading the created csv file with tweets

```
pd.read_csv("Psoriasis_3.csv")
```

	username	name	place	tweet	language	mentions
	ariba05723130	ariba	NaN	https://t.co/1C902DTF5Z Natural herbal remedi...	en	[]
	postematico	Postemático	NaN	"🟢 Este es uno de los principales ingredientes...	es	[]
	snarlincarl	Carl Sonnefeld	NaN	If I'm Shy Glizzy I am suing the hell out of S...	en	[]
	debesleerlo	Debes Leerlo	NaN	🟢 Psoriasis: el problema (y la solución) está ...	es	[]
	soyeahbasically	Soyeahbasically	NaN	@WithBiden1 Psoriasis	pt	[]
	...	...	...	...	...	...
	jurwidyakop3	JurnalWidyaKopertis3	NaN	https://t.co/PnRVf86E5P Dameria Sinaga PENGAR...	in	[]
	mardukhai	Scott Adler	NaN	@cosmoskw @RepKatiePorter Most of them seem to...	en	[]
	josh_adhd	Josh ADHD	NaN	If you actually watch commercials anymore, you...	en	[]
	chillipimmelaze	Lord Vader 💉 vaccinated	NaN	3 beliebige Fakten speziell über mich. Wer wil...	de	[]
	jaadjournals	JAAD Journals	NaN	Tonsillectomy and the subsequent risk of psori...	en	[]

Part I – Sentimental Analysis on Psoriasis Tweets

Problem Statement :

1.What are general concerns of people around Psoriasis?
2.What are the ways people discuss psoriasis and treatment?

1. Import necessary libraries

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import wordcloud
import re
import nltk
from nltk.stem.porter import *
stemmer = PorterStemmer()
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC, SVC, NuSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score, confusion_matrix
from sklearn.metrics import classification_report
```

```
from gensim.models.word2vec import Word2Vec, LineSentence, FastText, KeyedVectors
from gensim.models import Phrases, TfidfModel, LdaMulticore
from gensim.corpora.dictionary import Dictionary
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
from gensim.matutils import cossim, sparse2full
from gensim.similarities.docsim import SparseMatrixSimilarity
from gensim.similarities.termsim import WordEmbeddingSimilarityIndex, SparseTermSimilarityMatrix
from gensim.sklearn_api.tfidftransformer import TfidfTransformer
from gensim.sklearn_api.hdbscan import HDBSCAN
```

```
from nltk.tokenize.treebank import TreebankWordTokenizer
from nltk.tokenize.regexp import RegexpTokenizer
from nltk.tokenize.stanford_segmenter import StanfordSegmenter
from nltk.stem.porter import PorterStemmer
from nltk.stem.lancaster import LancasterStemmer
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus.reader.wordnet import WordNetCorpusReader
from nltk.corpus.reader.nombank import NombankCorpusReader
```

```
from nltk import word_tokenize, sent_tokenize, regexp_tokenize, wordpunct_tokenize, blankline_tokenize
from nltk.tokenize.stanford_segmenter import StanfordSegmenter
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus.reader import TaggedCorpusReader
```

```
from nltk.tokenize import PunktSentenceTokenizer, PunktWordTokenizer
from nltk.tokenize.simple import SpaceTokenizer, TabTokenizer, LineTokenizer
from nltk.tokenize.regexp import WordPunctTokenizer
```

```
from nltk import pos_tag, stopwords
from collections import Counter, OrderedDict
import string
import spacy
import gensim
import os
import collections
import smart_open
import h5py
import yellowbrick
import keras
import tensorflow
import scipy
import sklearn
import pickle
import pyLDAvis
import lightgbm
import xgboost
import warnings
warnings.filterwarnings("ignore")
warnings.filterwarnings("ignore", category=DeprecationWarning)
%matplotlib inline
```

2. Load extracted tweets csv

```
# Load extracted tweets csv
psoriasis = pd.read_csv('psoriasis.csv')
```

```
psoriasis.head(5)
```

	id	conversation_id	created_at	date	time	timezone	user_id	username	name	place	tweet	language	mentions	urls	photos	replies_count	nlikes	nretweets	quote_url	search
0	1747833425420345488	1747833425420345488	1642167476000	2022-01-14	17:31:16	+0000	1324497602344091136	psoriasisfree	Psoriasis Free		Can't wait for the #Psoriasis #Psoriasis #Psoriasis

Drug Review Web-Scrapping

Data Extracted from : https://www.drugs.com/condition/psoriasis.html

In [1]:
```
from bs4 import BeautifulSoup as bs
import requests
import re
import pandas as pd
import numpy as np
```

Automating the Scrapping Process

In [13]:
```
def get_review(block_soup):
    review = block_soup.find_all('p')
    for i in range(len(review)):
        temp = review[i].get_text()
        start = temp.index('"')+1
        end = temp.index('"')
        review[i]=temp[start:end]
    return review
```

1. Clobetasol

In [14]:
```
# Clobetasol review collection
reviews = []
med_name = []
count = 0
clobetasol_links = ["https://www.drugs.com/comments/clobetasol-topical/for-psoriasis.html",
                    "https://www.drugs.com/comments/clobetasol-topical/for-psoriasis.html?page=2",
                    "https://www.drugs.com/comments/clobetasol-topical/for-psoriasis.html?page=3"]

for i in range(len(clobetasol_links)):
    page = requests.get(clobetasol_links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)

reviews
med_name.extend(['clobetasol']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 66 and new length of med_name: 66
```

2. Humira

In [15]:
```
# Humira review collection
count = 0
humira_links = ["https://www.drugs.com/comments/adalimumab/humira-for-psoriasis.html",
                "https://www.drugs.com/comments/adalimumab/humira-for-psoriasis.html?page=2",
                "https://www.drugs.com/comments/adalimumab/humira-for-psoriasis.html?page=3",
                "https://www.drugs.com/comments/adalimumab/humira-for-psoriasis.html?page=4",
                "https://www.drugs.com/comments/adalimumab/humira-for-psoriasis.html?page=5"]

for i in range(len(humira_links)):
    page = requests.get(humira_links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['humira']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 178 and new length of med_name: 178
```

3. Stelara

In [16]:
```
# Stelara review collection
count = 0
links = ["https://www.drugs.com/comments/ustekinumab/stelara-for-psoriasis.html",
         "https://www.drugs.com/comments/ustekinumab/stelara-for-psoriasis.html?page=2",
         "https://www.drugs.com/comments/ustekinumab/stelara-for-psoriasis.html?page=3",
         "https://www.drugs.com/comments/ustekinumab/stelara-for-psoriasis.html?page=4",
         "https://www.drugs.com/comments/ustekinumab/stelara-for-psoriasis.html?page=5",
         "https://www.drugs.com/comments/ustekinumab/stelara-for-psoriasis.html?page=6",]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['stelara']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 303 and new length of med_name: 303
```

4. Methotrexate

In [17]:
```
# Methotrexate review collection
count = 0
links = ["https://www.drugs.com/comments/methotrexate/for-psoriasis.html",
         "https://www.drugs.com/comments/methotrexate/for-psoriasis.html?page=2",
         "https://www.drugs.com/comments/methotrexate/for-psoriasis.html?page=3",
         "https://www.drugs.com/comments/methotrexate/for-psoriasis.html?page=4"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['methotrexate']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 389 and new length of med_name: 389
```

5. Triamcinolone

In [18]:
```
# Triamcinolone review collection
count = 0
links = ["https://www.drugs.com/comments/triamcinolone-topical/for-psoriasis.html"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['triamcinolone']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 411 and new length of med_name: 411
```

6. Ustekinumab

In [19]:
```
# Ustekinumab review collection
count = 0
links = ["https://www.drugs.com/comments/ustekinumab/for-psoriasis.html",
         "https://www.drugs.com/comments/ustekinumab/for-psoriasis.html?page=2",
         "https://www.drugs.com/comments/ustekinumab/for-psoriasis.html?page=3",
         "https://www.drugs.com/comments/ustekinumab/for-psoriasis.html?page=4",
         "https://www.drugs.com/comments/ustekinumab/for-psoriasis.html?page=5",
         "https://www.drugs.com/comments/ustekinumab/for-psoriasis.html?page=6"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['ustekinumab']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 536 and new length of med_name: 536
```

7. Adalimumab

In [20]:
```
# Adalimumab review collection
count = 0
links = ["https://www.drugs.com/comments/adalimumab/for-psoriasis.html",
         "https://www.drugs.com/comments/adalimumab/for-psoriasis.html?page=2",
         "https://www.drugs.com/comments/adalimumab/for-psoriasis.html?page=3",
         "https://www.drugs.com/comments/adalimumab/for-psoriasis.html?page=4",
         "https://www.drugs.com/comments/adalimumab/for-psoriasis.html?page=5"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['adalimumab']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 650 and new length of med_name: 650
```

8. Remicade

In [21]:
```
# Remicade review collection
count = 0
links = ["https://www.drugs.com/comments/infliximab/remicade-for-psoriasis.html"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['remicade']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 655 and new length of med_name: 655
```

9. Cosentyx

In [22]:
```
# Cosentyx review collection
count = 0
links = ["https://www.drugs.com/comments/secukinumab/cosentyx-for-psoriasis.html"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['cosentyx']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 664 and new length of med_name: 664
```

10. Skyrizi

In [23]:
```
# Skyrizi review collection
count = 0
links = ["https://www.drugs.com/comments/risankizumab/skyrizi-for-psoriasis.html"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['skyrizi']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 668 and new length of med_name: 668
```

11. Taltz

In [24]:
```
# Taltz review collection
count = 0
links = ["https://www.drugs.com/comments/ixekizumab/taltz-for-psoriasis.html"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['taltz']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 685 and new length of med_name: 685
```

12. Etanercept

In [25]:
```
# Etanercept review collection
count = 0
links = ["https://www.drugs.com/comments/etanercept/for-psoriasis.html",
         "https://www.drugs.com/comments/etanercept/for-psoriasis.html?page=2"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['etanercept']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 714 and new length of med_name: 714
```

13. Guselkumab

In [26]:
```
# Guselkumab review collection
count = 0
links = ["https://www.drugs.com/comments/guselkumab/for-psoriasis.html"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['guselkumab']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 728 and new length of med_name: 728
```

14. Ixekizumab

In [27]:
```
# Ixekizumab review collection
count = 0
links = ["https://www.drugs.com/comments/ixekizumab/for-psoriasis.html"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['ixekizumab']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 746 and new length of med_name: 746
```

15. Tremfya

In [28]:
```
# Tremfya review collection
count = 0
links = ["https://www.drugs.com/comments/guselkumab/tremfya-for-psoriasis.html"]

for i in range(len(links)):
    page = requests.get(links[i])
    soup = bs(page.content, 'html.parser')

    block = soup.find_all('div', class_='ddc-comment ddc-box ddc-mgb-2')
    block_soup = bs(str(block), 'html.parser')
    temp = get_review(block_soup)
    count += len(temp)
    reviews.extend(temp)
reviews
med_name.extend(['tremfya']*(count)
print('new len of reviews: {} and new length of med_name: {}'.format(len(reviews), len(med_name)))
```

```
new len of reviews: 759 and new length of med_name: 759
```

Making the Data Frame:

In [29]:
```
dict_ = {'Drug_Name': med_name,'Review':reviews}
drug_df = pd.DataFrame(dict_)
drug_df
```

Out[29]:

	Drug_Name	Review
0	clobetasol	I have had moderate and sometimes severe plaqu...
1	clobetasol	I would use clobex on my thick plaque psoriasi...
2	clobetasol	CLOBEX IS AMAZING. It doesn't cure psoriasis ...
3	clobetasol	I used this medication as directed and it comp...
4	clobetasol	I have used Clobex for many years. It is the ...
...		...
754	tremfya	Hi, Ive been on steroids few times in past 10 ...
755	tremfya	15 mo on Tremfya for Psoriasis ||Prior, confl...
756	tremfya	I have a severe case of psoriasis only on the ...
757	tremfya	I feel like just so awful! Very sick with pain,...
758	tremfya	I have been on Tremfya for about a year. What ...

759 rows × 2 columns

In [30]:
```
# saving the file in the csv format
drug_df.to_csv('./drug.csv', index=False)
test_df = pd.read_csv('./drug.csv')
test_df.head()
```

Out[30]:

	Drug_Name	Review
0	clobetasol	I have had moderate and sometimes severe plaqu...
1	clobetasol	I would use clobex on my thick plaque psoriasi...
2	clobetasol	CLOBEX IS AMAZING. It doesn't cure psoriasis ...
3	clobetasol	I used this medication as directed and it comp...
4	clobetasol	I have used Clobex for many years. It is the ...

Part III - Sentiment Analysis on Psoriasis Drug Reviews

Problem Statement :

1) What are the most popular Psoriasis drugs/companies based on data?

1. Import necessary libraries

```
import pandas as pd
import numpy as np
import os
import re, nltk, nltk.stopwords
from nltk.stem.porter import PorterStemmer
import string
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import wordcloud, STOPWORDS
import sklearn.metrics as sm
%matplotlib inline
%matplotlib inline
%matplotlib inline
```

```
import nltk
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus.stopwords import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from textblob import TextBlob

import pycountry
import re
import string
import collections
from wordcloud import WordCloud, STOPWORDS
from PIL import Image
from collections import Counter, defaultdict
from plotly.offline.offline import iplot, init_notebook_mode
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.stopwords import ENGLISH_STOPWORDS
from sklearn.feature_extraction.text import CountVectorizer

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpus/stopwords.zip.
[nltk_data] Package stopwords is already up-to-date!
```

2. Load extracted drugs csv

```
df_drug = pd.read_csv('./drug.csv')
```

```
df_drug.head()
```

	Drug_Name	Review
0	Adalimumab	I have had moderate and sometimes severe plaque...
1	Adalimumab	I would give Humira a zero for the plaque psori...
2	Adalimumab	CLEARED IN MONTHS! Erlanda's son psoriasis ...
3	Adalimumab	I used this medication for about 2 years. I ...
4	Adalimumab	I have used Enbrel for many years. It is the ...

```
print(df_drug.iloc[0], sep=',')
```

```
Drug_Name                                                  Adalimumab
Review                  I have had moderate and sometimes severe plaque...
Name: 0, dtype: object
```

```
df_drug.shape
```

```
(750, 2)
```

```
df_drug[pd.isnull(df_drug).any(axis=1)]
```

	Drug_Name	Review
750 rows × 2 columns		

```
df_drug.drug_name.value_counts()
```

```
Adalimumab    80
Etanercept    80
Ustekinumab   80
Secukinumab   80
Ixekizumab    80
Brodalumab    80
Guselkumab    80
Tildrakizumab 80
Risankizumab  80
Apremilast    80
Name: drug_name, dtype: int64
```

```
# Add a word column and sentiment column
df_drug['review_length'] = df_drug['review'].apply(lambda x: len(x.split(' ')))
df_drug['sentiment'] = df_drug['review'].apply(lambda x: TextBlob(x).sentiment.polarity)
df_drug[['review', 'review_length', 'sentiment']].head()
```

	review	review_length	sentiment
0	I have had moderate and sometimes severe plaque...	125	0.151042
1	I would give Humira a zero for the plaque psori...	37	0.000000
2	CLEARED IN MONTHS! Erlanda's son psoriasis ...	35	0.166667
3	I used this medication for about 2 years. I ...	75	0.166667
4	I have used Enbrel for many years. It is the ...	75	0.250000

3. Getting most common words in training dataset

```
all_words = []
for review in df_drug['review']:
    for word in review:
        all_words.append(word)
all_words = nltk.FreqDist(all_words)
print('{0} words.'.format(len(all_words)))
```

```
11.1k words.
```

```
# Frequency of the 20 most common words
for word, frequency in all_words.most_common(20):
    print('{0}: {1}'.format(word, frequency))
```

```
.: 38811
,: 32811
the: 19811
a: 19811
and: 19811
to: 19811
of: 19811
in: 19811
is: 19811
it: 19811
for: 19811
on: 19811
with: 19811
my: 19811
this: 19811
that: 19811
was: 19811
I: 19811
```

```
# Generating Negative, Positive, Neutral and Compound values
# Reference: https://towardsdatascience.com/nlp-sentiment-analysis-3fd6d9f969ad | pd.Series(df[text_feed].sent_count)
for index, row in df_drug['review'].iteritems():
    score = SentimentIntensityAnalyzer().polarity_scores(row)
    neg = score['neg']
    neu = score['neu']
    pos = score['pos']
    comp = score['compound']
    if neg > pos:
        df_drug.loc[index, 'sentiment'] = 'negative'
    elif pos > neg:
        df_drug.loc[index, 'sentiment'] = 'positive'
    else:
        df_drug.loc[index, 'sentiment'] = 'neutral'
    df_drug.loc[index, 'neg'] = neg
    df_drug.loc[index, 'neu'] = neu
    df_drug.loc[index, 'pos'] = pos
    df_drug.loc[index, 'compound'] = comp
df_drug.head()
```

	Drug_Name	Review	review_length	positive	subjectivity	sentiment	neg	neu	pos	compound
0	Adalimumab	I have had moderate and sometimes severe plaque...	125	0.151042	0.583958	negative	0.131	0.811	0.073	-0.8609
1	Adalimumab	I would give Humira a zero for the plaque psori...	37	0.000000	0.000000	neutral	0.000	1.000	0.000	0.0000
2	Adalimumab	CLEARED IN MONTHS! Erlanda's son psoriasis ...	35	0.166667	0.666667	positive	0.000	0.750	0.250	0.7269
3	Adalimumab	I used this medication for about 2 years. I ...	75	0.166667	0.666667	positive	0.000	0.750	0.250	0.7269
4	Adalimumab	I have had moderate and sometimes severe plaque...	75	0.250000	0.625000	negative	0.100	0.750	0.150	0.2732

```
df_drug.to_csv('drug_sentiments.csv')
```

```
# Cleaning the data (clean the all sentiment (positive, negative and neutral)
# Reference: https://towardsdatascience.com/nlp-sentiment-analysis-3fd6d9f969ad
df_drug['review'] = df_drug['review'].str.lower()
review_neutral = df_drug[df_drug['sentiment'].isin(['neutral'])]
```

```
# Function for stop words and tokenization
def count_words_in_sentence(x, keywords):
    '''Helper function to count number of keywords in a given phrase'''
    splitted = re.split('[^a-z]', x.lower())
    intersection = set(splitted).intersection(set(keywords))
    return len(intersection)
# Apply function to create new column
df_drug['stop_words'] = df_drug['review'].apply(lambda x: count_words_in_sentence(x, ENGLISH_STOPWORDS))
# Print out the number of stop words in each review, sorted by descending order
print('Number of stopwords in review:')
df_drug['stop_words'].sort_values(ascending=False).head(10)
```

```
Number of stopwords in review:
```

```
sentiment
positive    75  56.0
neutral     19  14.0
negative    36  30.0
neutral     19   1.0
```

```
# Create and fit the TfidF
def pos_tfidf(df_drug, column):
    tfidf = TfidfVectorizer(sublinear_tfidf=True)
    X = tfidf.fit_transform(df_drug[column])
    return X, tfidf
# Create a column for the length of the post
df_drug['review_length'] = df_drug['review'].apply(lambda x: len(x))
pos_positive, tfidf_positive = pos_tfidf(df_drug[df_drug['sentiment'] == 'positive'], 'review')
print(pos_positive.shape)
print(tfidf_positive.vocabulary_ - tfidf_positive.vocabulary_.values())
```

```
(75, 1)
{'review_length': 0}
```

```
drug_name = df_drug['drug_name'].unique()
```

```
len(drug_name)
```

```
10
```

```
drug_compositions = df_drug['drug_name'].value_counts()
plt.figure(figsize=(10,6))
sns.barplot(drug_compositions[:10], drug_compositions[:10].values, palette='Spectral')
plt.xlabel('Drug Name')
```

Insight:

- The graph shows the comparision of various drug reviews for the entire dataset.
- **Dataset** and **Ustekinumab** have the highest number of reviews.
- **Psoriasis** and **Marvel** have lowest number of reviews.

```
plt.figure(figsize=(10,6))
for drug, color in zip(drug_name):
    temp_df = df_drug[df_drug['drug_name'] == drug.replace(sep=' ')]
    temp = temp_df['sentiment'].value_counts()
    df1 = temp[0 if temp.shape[0] > 0 else 0]
    df2 = temp[1 if temp.shape[0] > 1 else 1]
    df3 = temp[2 if temp.shape[0] > 2 else 2]
    plt.bar(drug, df1, color='r')
    plt.bar(drug, df2, color='g')
    plt.bar(drug, df3, color='b')
plt.xlabel('Drug')
plt.ylabel('Sentiment')
plt.legend(['Positive', 'Neutral', 'Negative'])
```

Insight:

- Ustekinumab, Cosentyx, Remicade: These are the drugs which have a higher positive sentiment attached to them compared to the other drugs, with Remicade having the highest positive sentimental scores (83.67%), leading Ustekinumab and Cosentyx for the second and third place respectively.
- All the other drugs have more or less an equal proportion for both positive and negative sentiments, with positive sentiments having a bigger portion.
- Stelara is the drug that has a higher negative sentiment attached to it compared to the rest of the drugs.

```
pos_figure(figsize=(8,5))
plt.pie(df_drug, 'sn')
```

4. Word Cloud Analysis

```
%matplotlib inline
def wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='black',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1
    ).generate(str(data))
```

```
fig = plt.figure(1, figsize=(15, 15))
plt.axis('off')
if title:
    fig.suptitle(title, fontsize=20)
plt.imshow(wordcloud.to_array())
pos_wordcloud(df_drug)
```

```
# Drawing wordcloud for positive sentiment
pos_wordcloud(df_drug[df_drug['sentiment'] == 'positive'], 'Word Cloud for Positive Reviews')
```

injection
day
Humira
side
effect
psoriasis
skin
Stelara
month
treatment
clear
help
life
work
great
dose
used
thing
Enbrel
cream
little
back
every
taking
week
medicine
shot
doctor
one
first
methotrexate
body
drug
severe
years
time
bad
started
worked
cleared
result
pain
issue
medication
week
year
first
day
helped
time
amazing
shower
hand
feel
skin
scalp
psoriasis
Stelara
Humira
methotrexate
injection
side
effect
severe
month
cleared
worked
started
bad
work
years
pain

Chapter 4: Conclusion

We started the analysis with a preconceived notion that people are mean to other people suffering from Psoriasis. The sentiment analysis we performed on Twitter data has proven to be extremely useful in gaining an overview of the wider public opinion behind Psoriasis. The data that we collected from Twitter.com and Drug.com explains some part of our understanding before analysing it. After analysing the data, we could see a lot of positive comments being shared on the social media, most of them were self-narratives of struggle and fight against the disease. The negative tweets were majorly concentrated towards the pain people suffer, the anxiety of looking good or the medication not working and so on.

The data about the various drugs in use by people in respect to Psoriasis was more or less neutral (from a holistic point of view). In-depth analysis of the same gave us insights that they were neutral as people started off with a positive / negative narrative and then deviated to the opposite sentiment making the entire review classified as neutral. The Pie charts plotted for just the negative and the positive sentiments showed us that most drugs had mixed reaction i.e., had positive and negative reviews in approximately the same proportion.

We observed that **Otezla** and **Amgen** were 2 companies which were positively spoken about in the tweets. **Clobetasol, Cosentyx, Remicade** were the drugs that had greater positive feedback, with Remicade having the highest positive sentimental reviews (83.91%). Making **Johnson and Johnson Inc.** the most trusted drug manufacturing company for Psoriasis.

Even though our model worked well and was able to classify the sentiments correctly for the tweets and the reviews, due to lack of data, our insights are very specific to the dataset and does not capture the general perspective of people. Future work like tokenization of the dataset after capturing the sentiments can now be used to classify the data directly using any machine learning models, without the help of NLTK package.

Our analysis can help companies understand what are the various factors that affect people the most in respect to Psoriasis, so that they can address those concerns in a better way. The drug review analysis can help drug manufacturing companies understand the sentiment of people towards their drug and, can help them improve their drug to have a larger customer base.