Compiler design - CS6109

Project submission

Sentiment analysis of tweets using lexical & semantic concepts

Identifying the response of the public for an entity

Presented by,

S Somaya Sree - 2018103598

Aparna S S - 2018103008

Juanita J - 2018103544

Introduction

Twitter is one of the micro-blogging sites that have scored its place in the ever increasing social networking epoch. The present scenario of the microblogging sites have made the opinion forming and advancement very efficient. The public opinion formed using tweets is direct. People use emoticons, short hands etc to write the tweets. People tweets can be used to analyze thoroughly and effectively to shape up the opinions, thus the opinion mining. At present, sentiment analysis is one of the most important field in the development of the organization and thus many technologies have been applied to automate the work as in Natural Language Processing(NLP) and Machine learning algorithms(ML).

Problem Statement

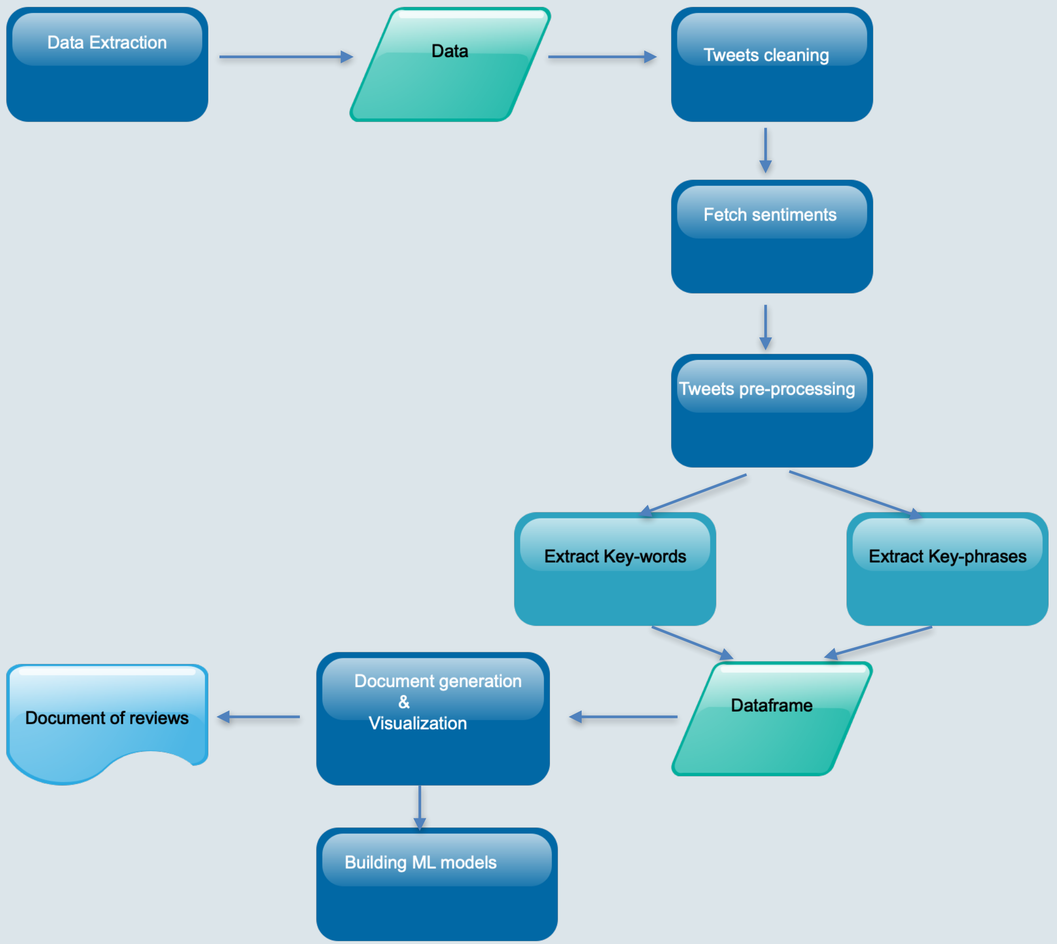
Until now the work of the sentimental analysis is largely limited review of movies but twitter data analysis is gaining much recognition in the present scenario may be because it provides the latest and the most instant opinion of the public on almost all the latest fields of interest. Twitter sentiment analysis technology provides the methods to survey public emotion about the events or products related to them. Sentiment analysis over Twitter offer organizations a fast and effective way to monitor the publics' feelings towards their brand, business, directors, etc. The project focuses on deriving the sentiments related to a specific entity which implies how the entity has been received by the public.

Literature survey

Title of the paper	Author	Methodologies	Limitations
Alleviating data sparsity for twitter sentiment analysis Published - 2012	Saif Hassan, He, Yulan and Alani	Entity extraction Incoporating semantic concepts	Static dataset Language limitation Absence of future predictions
Real time sentiment analysis of political twitter data using machine learning approach Published - 2019	Joylin Riya Pinto, Visaya Muradi	Analysis of public's receival for an entity Sentiment classification Predictive models	Static dataset Language limitation

Architecture

1. **Data Extraction** - Extraction of tweets using tweepy
2. **Tweets Cleaning** - Removal of unwanted details
3. **Fetch sentiment** - Deriving positive and negative sentiments
4. **Tweets pre-processing** - Extraction of Keywords & Keyphrases
5. **Document generation & visualization** -Visualizing positive & negative reviews , storing them in csv file
6. **Build ML models** - Construct predictive models



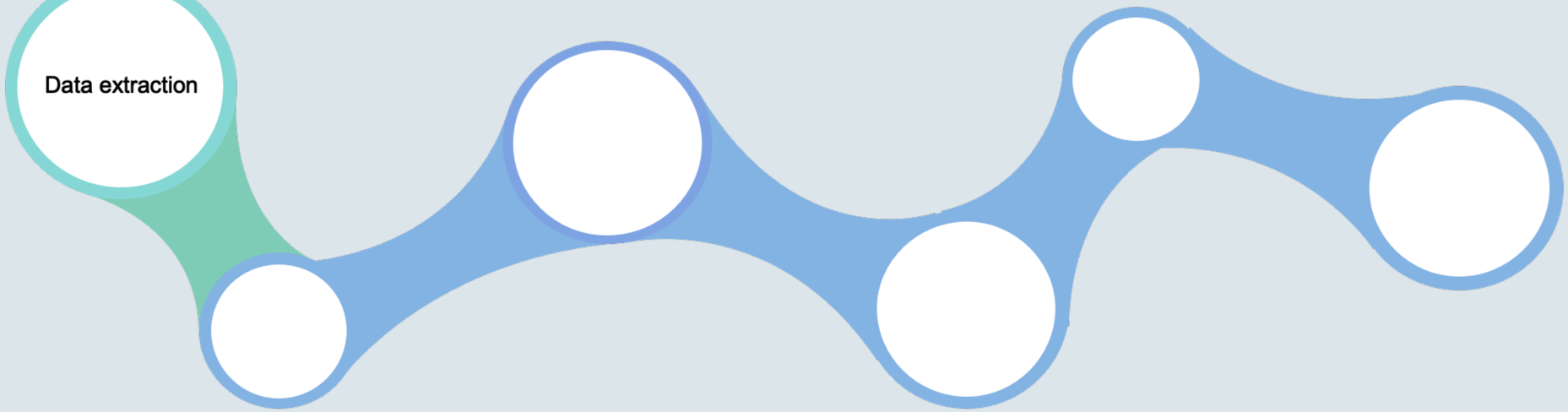
Entity chosen

iOS 14

iOS 14 is the fourteenth and current major release of the iOS mobile operating system developed by Apple Inc. for their iPhone and iPod Touch lines. It was released on September 16, 2020. This is the entity whose reach to the public is going to be analyzed via sentiment analysis of tweets.

Module 1

The real time tweets regarding a specific entity is fetched via Tweepy. These tweets are then used to analyze how the public has received the entity



Tweets extraction using Tweepy

Tweepy is an open source Python package that gives you a very convenient way to access the Twitter API with Python. It does so by encapsulating much of the Twitter API's complexity and adding a model layer and other useful functionalities on top of it.

The Tweepy functionality groups used here are
OAuth & *API*("Search" method)

OAuth - Tweepy takes care of all the details for using OAuth required by the Twitter API to authenticate each request. It provides an OAuthHandler class that is used to set the credentials to be used in all API calls.

API - The API class has many methods that provide access to Twitter API endpoints. Using these methods, the Twitter API's functionality can be accessed. The Search method is used to extract the most recent tweets for a given query.

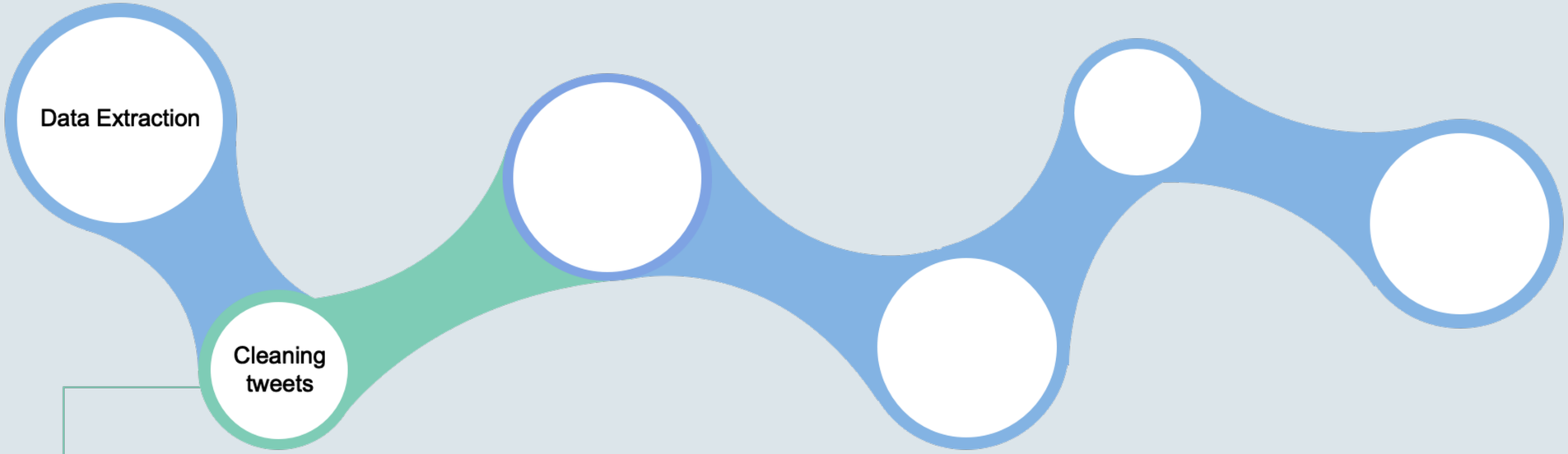
```
twitter_client = TwitterClient()

# calling function to get tweets
tweets_df = twitter_client.get_tweets
('iOS 14', maxTweets=20000)
```

```
Downloaded 98 tweets
Downloaded 198 tweets
Downloaded 298 tweets
Downloaded 398 tweets
Downloaded 498 tweets
Downloaded 598 tweets
Downloaded 698 tweets
Downloaded 798 tweets
Downloaded 898 tweets
Downloaded 998 tweets
Downloaded 1098 tweets
Downloaded 1198 tweets
Downloaded 1298 tweets
Downloaded 1398 tweets
Downloaded 1498 tweets
Downloaded 1598 tweets
Downloaded 1698 tweets
Downloaded 1798 tweets
Downloaded 1898 tweets
```

The entity chosen **iOS 14** and the maximum number of tweets to be fetched are passed as parameters to the **get_tweets()** function(coded).

Module 2



The tweets extracted via the API needs some cleansing and processing activities to make them fit to fetch sentiments from them. Also the tweets are translated to get accurate sentiments.

Operations on specific lexicons

```
['Hai kak👨🏻\u200d♀ Aku jual Spotify Premium cuma 9K ! Netflix cuma 30k! Apple Music bisa IOS 14 ! App premium lainny
a bis… https://t.co/xImTPOCip1',
 'RT @APride0fficial: iBypasser v2.4 💯\n✅Untethered bypass with all icloud services work\nFree Re-Bypass\n✅FMI OFF
for iOS 13-14 (Paid 2$)\n✅Free…',
 'RT @kirarafantasia: 【お知らせ】\n「iOS 14」「iPadOS 14」「Android 11」につきまして、『きららファンタジア』の動作確認が完了いたし
ました。詳細につきましてはおしらせをご確認ください。\n#きららファンタジア\nhttps://t.co/G…',
 'Pinterest launched a new Pinterest widget for iOS 14, which enables to feature boards on the home screen. This wil
l… https://t.co/IfLl7vV8r5',
 '@HooverUSA the app no longer works on iOS 14, it keeps asking for local network permissions (which I allowed in Se
t… https://t.co/MnwzGH1TJE',
 'iOS 14はアプリの削除方法が変わったので要注意（アスキー） https://t.co/RGIGPnNnMb',
 'בדיקה של iOS 14 של אפל מגלה פיתוח חדש של האייפון או האייפד בגרסאות\nhttps://t.co/YtUpU9SIwf',
 'あいぽんにパスモを入れたかったのと、背面タップがやってみたくて、iOSを14にアプデしました🐸',
```

It's clearly visible that the live tweets extracted needs to be refined to extract the sentiment associated with them

Mentioned below are some of the steps carried to clean the tweets.

1. ***Removing '@names'*** - The mentions are removed since the focus is on the sentiment of the tweet rather than the entity which every tweet points to
2. ***Removing links(http|https)*** - The links contained in the tweets are avoidable for sentiment analysis
3. ***Removing tweets with empty text*** - The empty tweets are of no use
4. ***Dropping duplicate rows*** - The duplicate tweets are dropped since the redundant tweets takes unnecessary space
5. ***Removing punctuations, numbers & special characters*** - These characters are noisy from the viewpoint of sentiment analysis

Language translation

Necessity to translate tweets

```
tweet = tweets_df['tidy_tweets'][46]
print(tweet)
sid = SentimentIntensityAnalyzer()
polarity_scores = sid.polarity_scores(tweet)
print(polarity_scores)
```

```
iOS 14の「メモ」アプリで4つの新機能を使いこなそう #it #feedly
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
```

```
tweet = tweets_df['tidy_tweets'][46]
print(tweet)
translation = translator.translate(tweet)
tweet = translation.text
print(tweet)
sid = SentimentIntensityAnalyzer()
polarity_scores = sid.polarity_scores(tweet)
print(polarity_scores)
```

```
iOS 14の「メモ」アプリで4つの新機能を使いこなそう #it #feedly
Take full advantage of four new features in the iOS 14 Memo app #it #feedly
{'neg': 0.0, 'neu': 0.875, 'pos': 0.125, 'compound': 0.25}
```

The sentiment extracted from the translated tweet is more specific(the positivity of the new features is identified and is reflected in the positive score) unlike the sentiment from the tweet in other language which generalizes the tweet as neutral

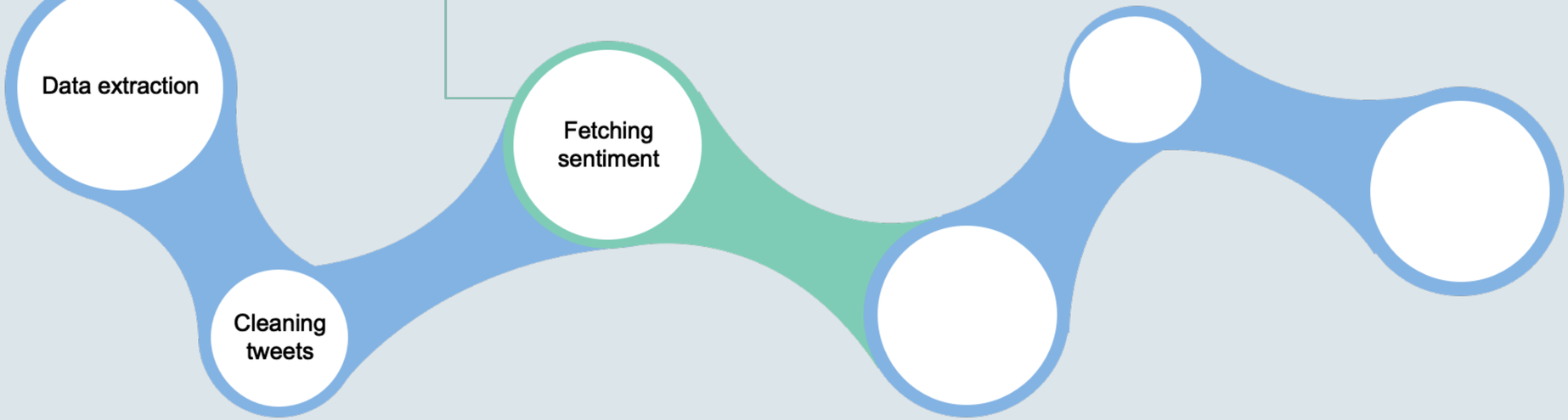
The tweepy extracts the tweets for the specified conditions. No language specification was provided since the tweets from all languages are needed to analyze the response from the public of all nationalities. But it's equally important to translate the tweets to 'english' since the sentiments are extracted much better from English compared to other languages.

The Google translate API

Google Translate API is a simple API that allows you to translate an arbitrary string of text from one language to the other. The API supports two endpoints, "detect" and "translate". As the name suggests, one is for detecting the language, and the other is for translating from one language to another.

Module 3

Having cleaned and translated the tweets, the sentiments of the tweets are fetched for two reasons,
1. To analyze the overall sentiment regarding an entity from the public
2. To get the targets to pass them to the machine learning models



Extraction of sentiments

Vader's SentimentIntensityAnalyzer (SIA)

VADER produces four sentiment metrics from the word ratings. The first three, positive, neutral and negative, represent the proportion of the that falls into those categories.The final metric, the compound score, is the sum of all of the lexicon ratings which have been standardised to range between -1 and 1.

```
tweet = tweets_df['tweets_en'][47]
print(tweet)
sid = SentimentIntensityAnalyzer()
polarity_scores = sid.polarity_scores(tweet)
print(polarity_scores)
```

```
Interesting UI tearing bug in Twitter for iOS 14.2
{'neg': 0.0, 'neu': 0.748, 'pos': 0.252, 'compound': 0.4019}
```

The sentiment scores of a sample tweet by SIA

Textblob

The sentiment object of textblob returns a tuple that contains polarity and subjectivity of the review. Polarity is considered. The value of polarity can be between -1 and 1 where the reviews with negative polarities have negative sentiments while the reviews with positive polarities have positive sentiments.

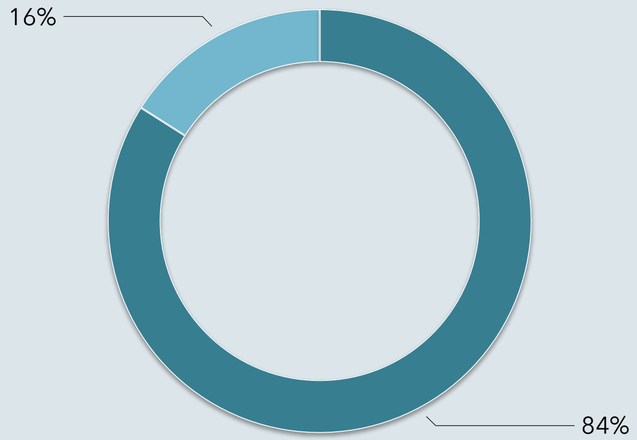
```
tweet = tweets_df['tidy_tweets'][47]
print(tweet)
analysis = TextBlob(tweet)
print(analysis.sentiment.polarity)
```

```
Interesting UI tearing bug in Twitter for iOS 14.2
0.5
```

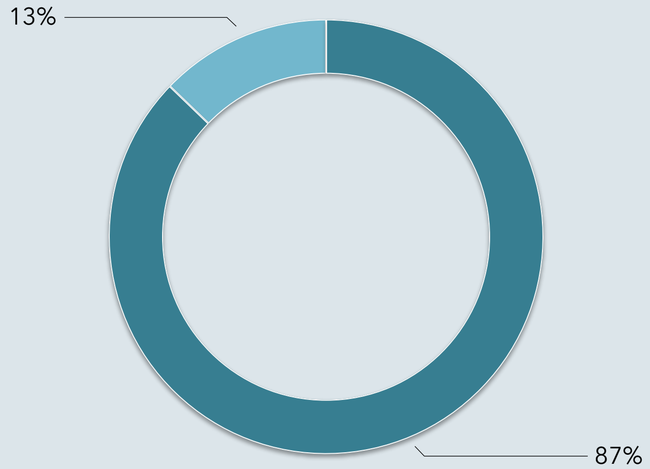
The sentiment score of a sample tweet by Textblob

Choice of sentiment analyzer

SIA

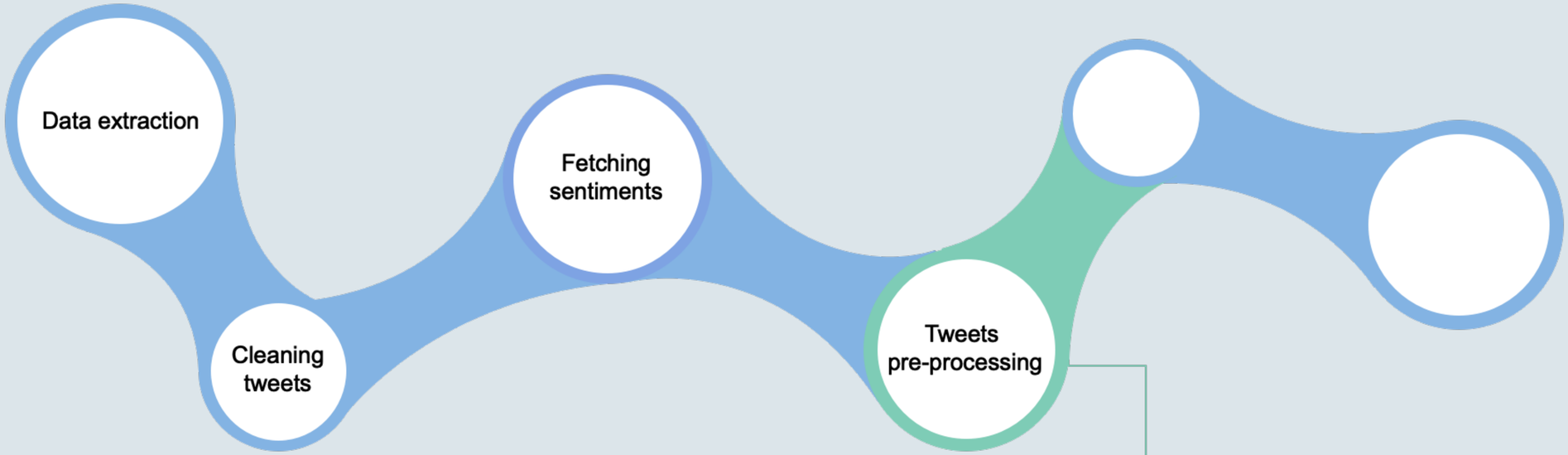


Textblob



While the SIA from Vader tool is universal and can be used everywhere to analyze the text (i.e., for general use), Textblob focuses on specific domain by making use of handpicked adjectives. The SIA gives labels more tweets as negative when compared to textblob. Here SIA is considered because the classes are better balanced. Class balance is given higher preference here due to the fact that both the sentiment analyzers are proven to produce nearly same results and the results would be used for the classification tasks if the sentiments are going to be predicted by Machine Learning models

Module 4



The translated tweets has to be processed further to extract the key-words and key-phrases for the upcoming classification tasks. Its based on these keywords and key-phrases that the model will learn the patterns of positive and negative tweets.

Extraction of Key-words

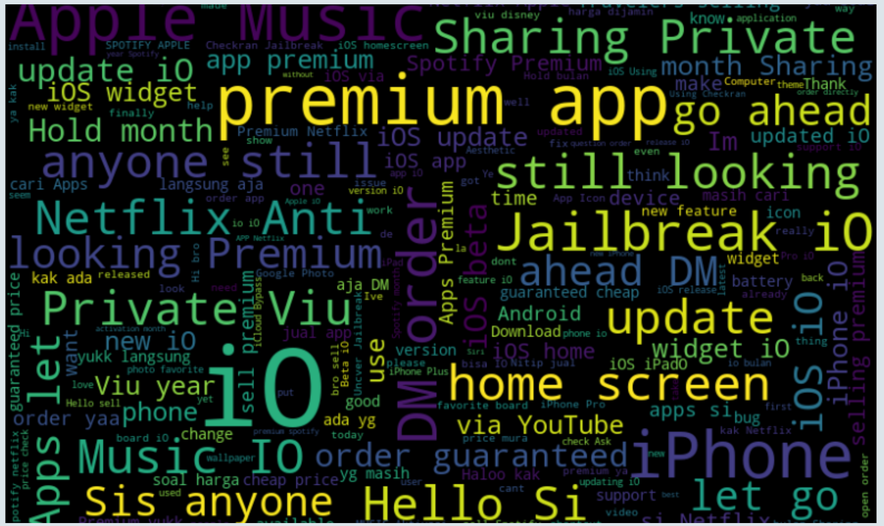
The absolute tidy tweets

1. **Removal of stop words -** The stop words are removed from the tweets using the '**Stopwords'** library available in the NLTK toolkit. It has a list of stopwords stored in 16 different languages.

2. **Tokenize 'absolute tidy tweets'** - The tweets free from stop words are tokenized by a simple python code.

3. **Converting tokens to lemma -** The tokens are lemmatized using the the '**Wordlemmatizer'** library from the '**Stem'** package available in the NLTK toolkit.

4. **Joining all the tokens into sentences -** The lemmatized words are then joined back to form sentences and this is the set of updated 'absolute tidy tweets'.



The keywords from the positive tweets

The keywords from the negative tweets

Extraction of Key-phrases

The helper class

1. **__init__ -** Initializes the the lemmatizer and stemmer from the NLTK. PorterStemmer class chops off the 'es' from the word. On the other hand, WordNetLemmatizer class finds a valid word.

2. **Leaves -** Finds NP (nounphrase) leaf nodes of a chunk tree.

3. **Normalise -** Normalises words to lowercase and stems and lemmatizes it.

4. **Acceptable word -** Checks conditions for acceptable word: length, stopword. The length of the word can be increased when considering larger phrases

5. **Get terms -** Makes use of the above functions to yield terms

```
class PhraseExtractHelper(object):
    def __init__(self):
        self.lemmatizer = nltk.WordNetLemmatizer()
        self.stemmer = nltk.stem.porter.PorterStemmer()

    def leaves(self, tree):
        for subtree in tree.subtrees
        (filter = lambda t: t.label()=='NP'):
            yield subtree.leaves()

    def normalise(self, word):
        word = word.lower()
        word = self.lemmatizer.lemmatize(word)
        return word

    def acceptable_word(self, word):
        accepted = bool(3 <= len(word) <= 40
            and word.lower() not in stopwords
            and 'https' not in word.lower()
            and 'http' not in word.lower()
            and '#' not in word.lower()
            )
        return accepted

    def get_terms(self, tree):
        for leaf in self.leaves(tree):
            term = [ self.normalise(w)
                    for w,t in leaf if self.acceptable_word(w) ]
            yield term
```

Grammar definition for parse tree

```
sentence_re = r'(?:(?:[A-Z])(?:.[A-Z])+.?)|(?:\w+(?:-\w+)*)|(?:\$?\d+(?:.\d+)?%?)|(?:...|)(?:[][.,;"\'?():-_`])'
grammar = r"""
    NBAR:
        {<NN.*|JJ>*<NN.*>}  # Nouns and Adjectives, terminated with Nouns

    NP:
        {<NBAR>}
        {<NBAR><IN><NBAR>}  # Above, connected with in/of/etc...
"""
chunker = nltk.RegexpParser(grammar)
```

1. **Building Tokenizer -** Instead of simply tokenizing, a pattern has been pre-defined by means of a regular expression. The strings are tokenized in such a way that they match these patterns by the **'regexp_tokenize'** function from NLTK

2. **Obtaining the POS -** The **'pos_tag'** function in the **'tag'** library of NLTK returns the tokens with their POS(Parts Of Speech) as pairs.

3. **Fitting tokens in RegexpParser -** The parser is defined by a grammar into which the <token, POS> pairs are fitted

4. **Obtaining key-phrases -** The parser is passed into the helper class to get the phrases

```
key_phrases = []
phrase_extract_helper = PhraseExtractHelper()

for index, row in tweets_df.iterrows():
    toks = nltk.regexp_tokenize(row.tweets_en, sentence_re)
    postoks = nltk.tag.pos_tag(toks)
    tree = chunker.parse(postoks)
    tree.draw()
    terms = phrase_extract_helper.get_terms(tree)
    tweet_phrases = []

    for term in terms:
        if len(term):
            tweet_phrases.append(' '.join(term))

    key_phrases.append(tweet_phrases)

#key_phrases[:10]
```

The code for extracting key phrases

Process of deriving key-phrases

Tweets

↓

Regular expression based tokenization

↓

Generation of <token, POS> pairs for each token

↓

Construction of parse tree with the defined grammar(with <token, POS> fitting)

↓

Helper class operations

↓

Refined Key-phrases

'Eve's light strip is first to support HomeKit's color-changing adaptive lighting'

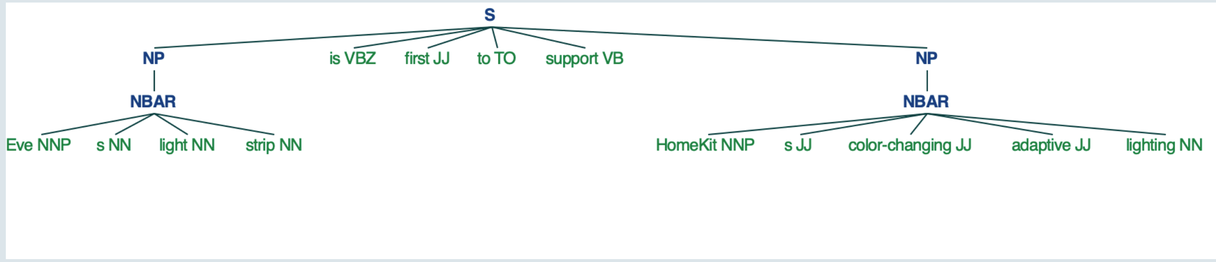
↓

['Eve', 's', 'light', 'strip', 'is', 'first', 'to', 'support', 'HomeKit', 's', 'color-changing', 'adaptive', 'lighting']

↓

[('Eve', 'NNP'), ('s', 'NN'), ('light', 'NN'), ('strip', 'NN'), ('is', 'VBZ'), ('first', 'JJ'), ('to', 'TO'), ('support', 'VB'), ('HomeKit', 'NNP'), ('s', 'JJ'), ('color-changing', 'JJ'), ('adaptive', 'JJ'), ('lighting', 'NN')]

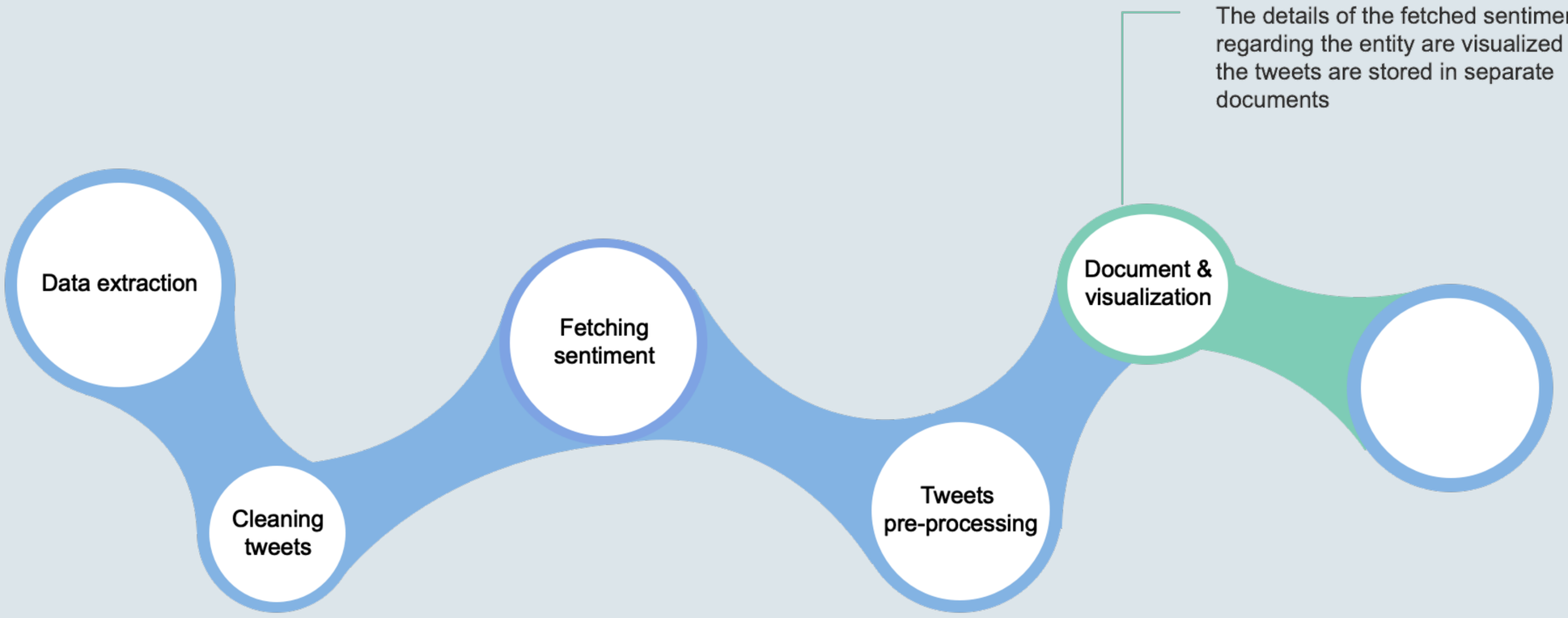
↓



↓

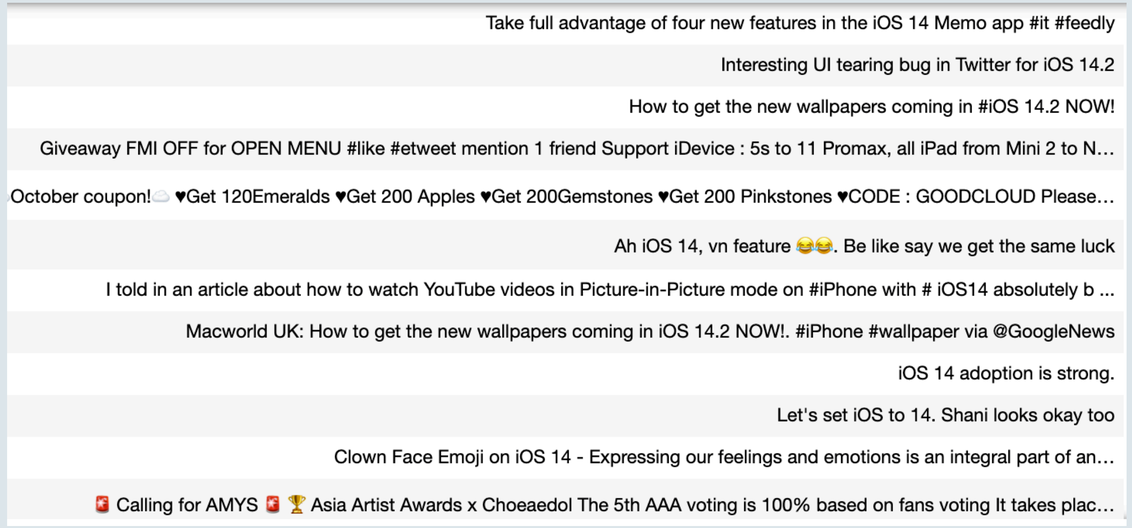
```
['eve '',
 'light strip',
 'support homekit '',
 'color-changing adaptive lighting']
```

Module 5


The details of the fetched sentiments regarding the entity are visualized and the tweets are stored in separate documents
Data extraction
Cleaning tweets
Fetching sentiment
Tweets pre-processing
Document & visualization

Document generation

Having analyzed the sentiments of tweets, the tweets with positive and negative reviews are stored in separate documents for the entity's reference purposes.



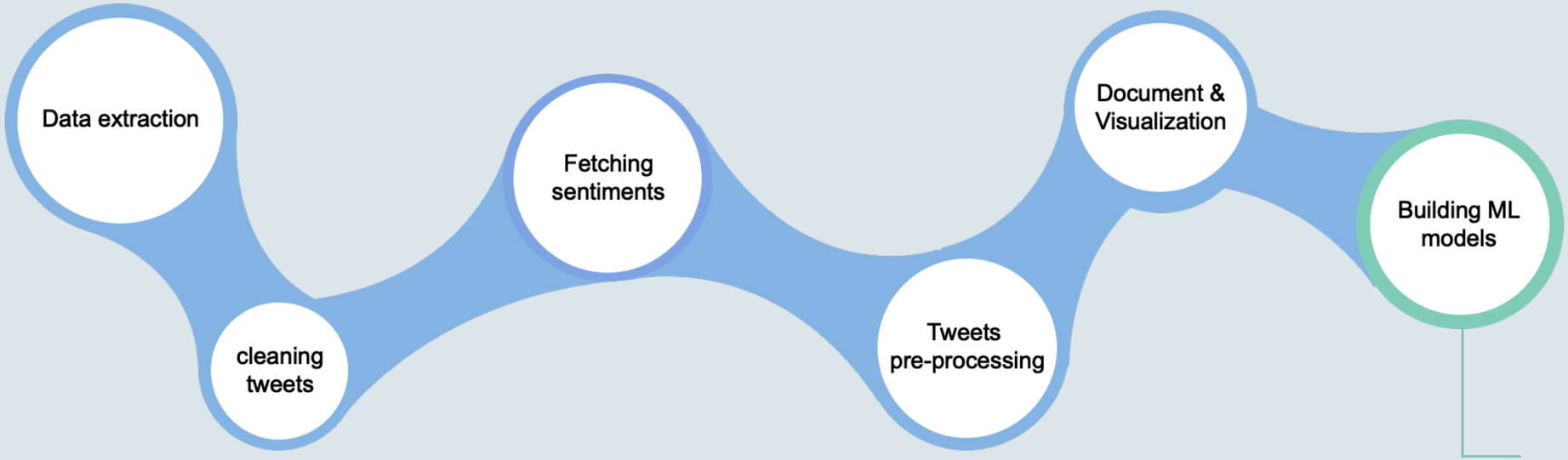
Take full advantage of four new features in the iOS 14 Memo app #it #feedly
Interesting UI tearing bug in Twitter for iOS 14.2
How to get the new wallpapers coming in #iOS 14.2 NOW!
Giveaway FMI OFF for OPEN MENU #like #etweet mention 1 friend Support iDevice : 5s to 11 Promax, all iPad from Mini 2 to N…
October coupon!☁️ ♥Get 120Emeralds ♥Get 200 Apples ♥Get 200Gemstones ♥Get 200 Pinkstones ♥CODE : GOODCLOUD Please…
Ah iOS 14, vn feature 😂😂. Be like say we get the same luck
I told in an article about how to watch YouTube videos in Picture-in-Picture mode on #iPhone with # iOS14 absolutely b ...
Macworld UK: How to get the new wallpapers coming in iOS 14.2 NOW!. #iPhone #wallpaper via @GoogleNews
iOS 14 adoption is strong.
Let's set iOS to 14. Shani looks okay too
Clown Face Emoji on iOS 14 - Expressing our feelings and emotions is an integral part of an…
🧧 Calling for AMYS 🧧 🏆 Asia Artist Awards x Choeaedol The 5th AAA voting is 100% based on fans voting It takes plac…

positive_reviews.csv

Not that I can see the outfits anymore anyways. The iOS 14 update kicked my old iPhones ass and can't take anything…
can ios 14 not turn down the volume when i'm in my car?! obviously playing it louder while driving. moronic update.
may disadvantage talaga ios 14 lakas sa data
Shortcuts für mehr Produktivität: Die Macht der iOS-14-Kurzbefehle
Big Sur beta, widgets iOS 14, Epic Battle … Semaine du podcast Infinite Loop
I forgot to put off the wifi, i come plug my phone join, iOS don update itself overnight to 14.1.... my data i cannur cry 😿
ios 14 dog nagging
A skeleton sketch for commission quests that is now unfortunate. The shape creation function of iOS 14 memo has a round head.
no cap i got rid of my ios 14 layout after 2 weeks i couldn't stand that it went to shortcuts before it opened the app.
I downgraded from 14.2. That's probably the problem since I experienced similar problems when I dow…
Apple has stopped downgrading to iOS 14.0.1 after updating to iOS 14.1.
iOS 14.0.1 (18A393) is no longer being signed. #iOS #iOS1401

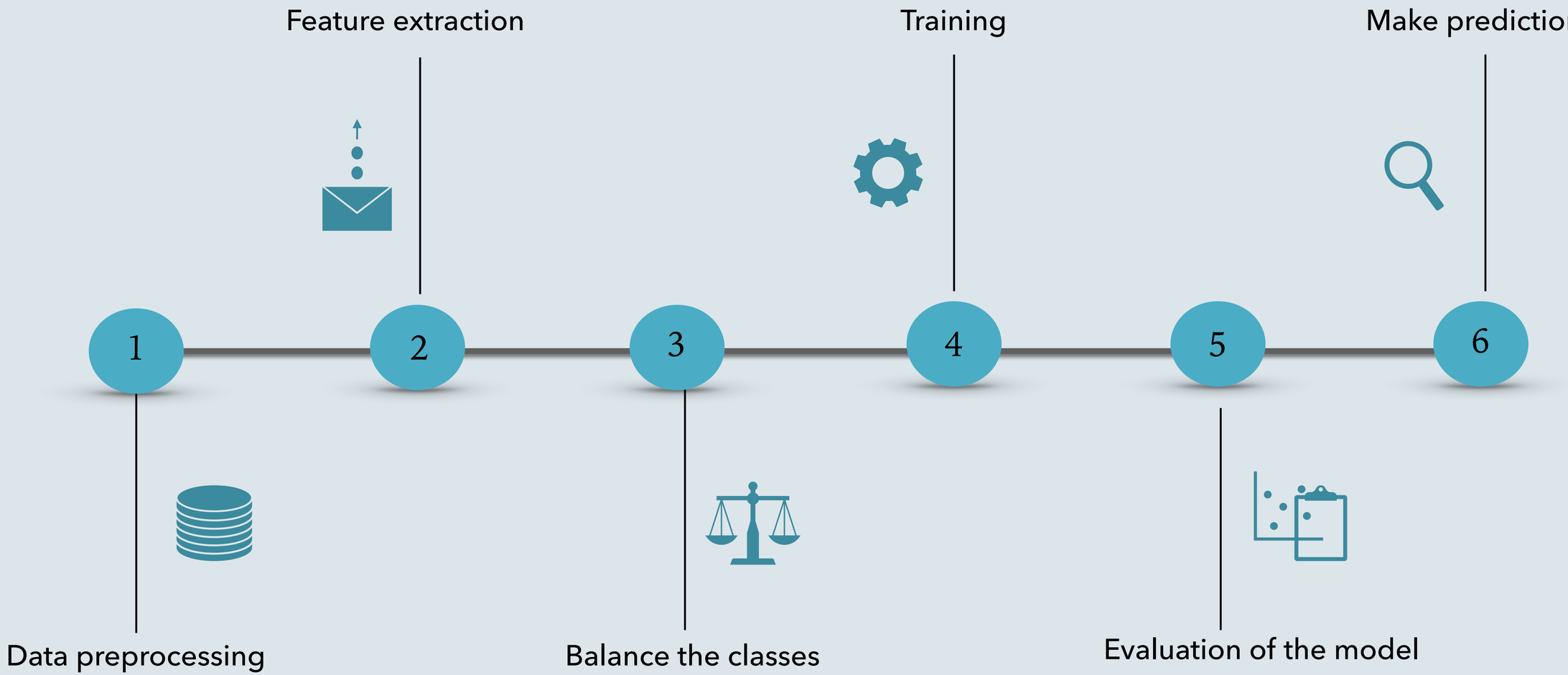
negative_reviews.csv

Module 6



With the completely tidy and processed tweets along with the sentiments, machine learning models are used to predict the sentiment for a dynamically given tweet.

The work flow of classifier model



Data pre-processing

Involves retaining the necessary details in the data and converting the target labels into numerical values for the ease of computation in fitting in the models.

Key-words vs Key-phrases

The Key-words and Key-phrases are extracted from tweets which are then used to train the ML models. Mentioned below are the steps followed for both the inputs fed into the models.

Feature extraction

The two methods of feature extraction used are as follows. Both the approaches vectorize the results in texual formats to numerical values by vectorization.

Bag Of Words (BOW)

It's a collection of words to represent a sentence with word count and mostly disregarding the order in which they appear.

TF-IDF (Term Frequency - Inverse Document Frequency)

A technique to quantify a word in documents, the weight of each word is computed which signifies the importance of the word in the document

The word feature vector by BOW & TF-IDF

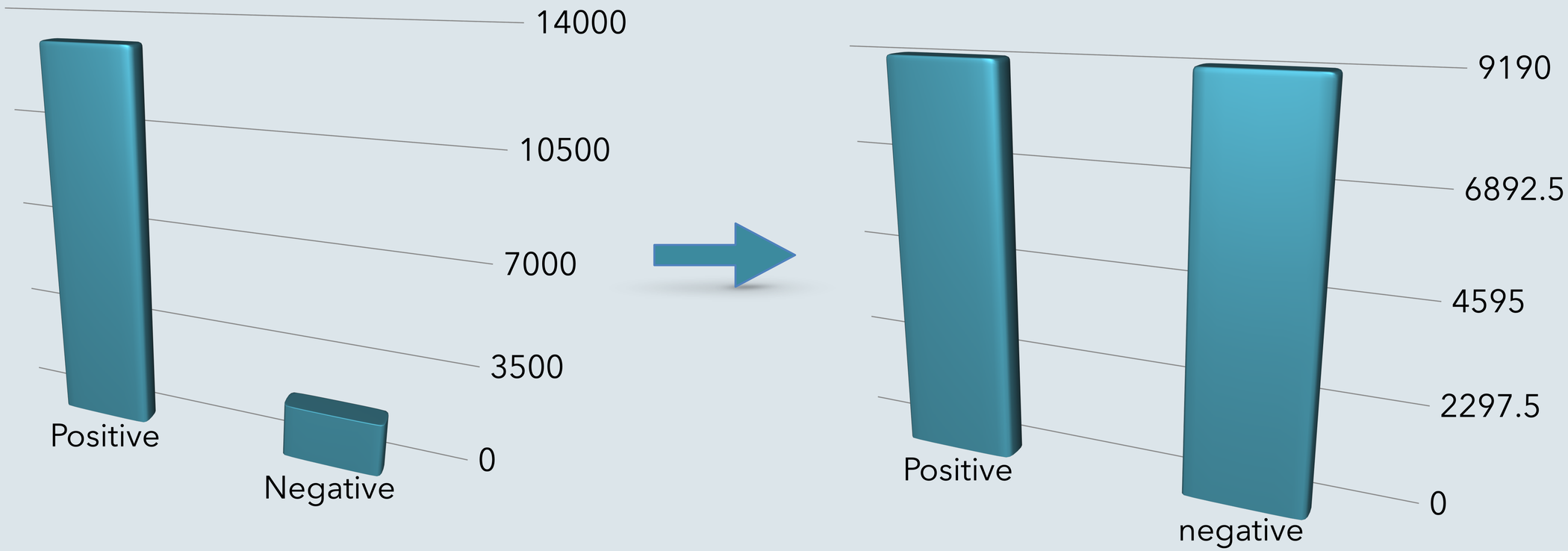
<10563x4138 sparse matrix of type '<class 'numpy.float64'>'
with 114089 stored elements in Compressed Sparse Row format>

The phrase feature vector by BOW & TF-IDF

<10563x4138 sparse matrix of type '<class 'numpy.float64'>'
with 76490 stored elements in Compressed Sparse Row format>

Balancing the classes

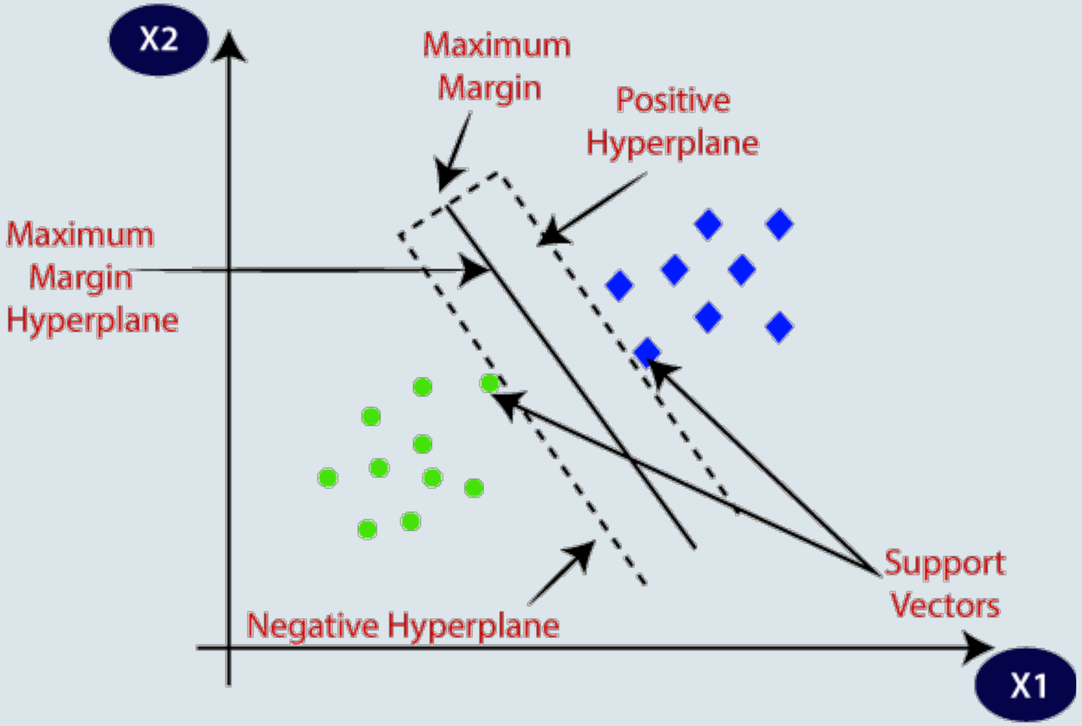
Training a machine learning model on an imbalanced dataset can introduce unique challenges to the learning problem. The classes are balanced by the sampling technique - ***SMOTETomek.*** It does so by combining under-sampling and over-sampling techniques.



Training the classifier

Classifier chosen - *Support Vector Machine(SVM)*

A support vector machine (**SVM**) is a supervised machine learning model that uses classification algorithms for two-group classification problems. The algorithm creates an optimal hyperplane which separates the data into classes. It is highly preferred by many as it produces significant accuracy with less computation power.




```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(x_train data, y_train data)
```

Code snippet to train the classifier

Short note on train test split

The entire dataset is split into train and test where,

- train data (80%) - for fitting into the model
- Test data (20%) - for testing the performance of the model

Evaluation of the classifier

Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data. The three main metrics used to evaluate a classification model are accuracy, precision, and recall. The scores obtained by both feature extraction methods namely the BOW and TF-IDF for both approaches namely Key-words and Key-phrases are shown below

Method	Overall accuracy	Precision		Recall		F1-score	
		Positive	Negative	Positive	Negative	Positive	Negative
Key-words BOW	0.80	0.88	0.33	0.82	0.63	0.88	0.43
Key-words TF-IDF	0.88	0.95	0.51	0.91	0.69	0.93	0.58
Key-phrases BOW	0.65	0.95	0.22	0.74	0.64	0.77	0.34
Key-phrases TF-IDF	0.68	0.95	0.24	0.75	0.67	0.79	0.37

Observation

Features extracted from 'key words' helps model in performing better. They have better positive and negative predictions. Also the TF-IDF technique is the recommended technique for this data.

Inferences and conclusion

The overall sentiment of the entity ‘iOS 14’ is concluded to be **positive** though the entity has to work on some serious issues to satisfy the customers. The generated documents will help through the process.

The predictive models predict the sentiment with a decent accuracy of 88% with key-phrases as input and TF-IDF as the feature extraction technique.

Thank you