SENTIMENT ANALYSIS FOR MARKETING

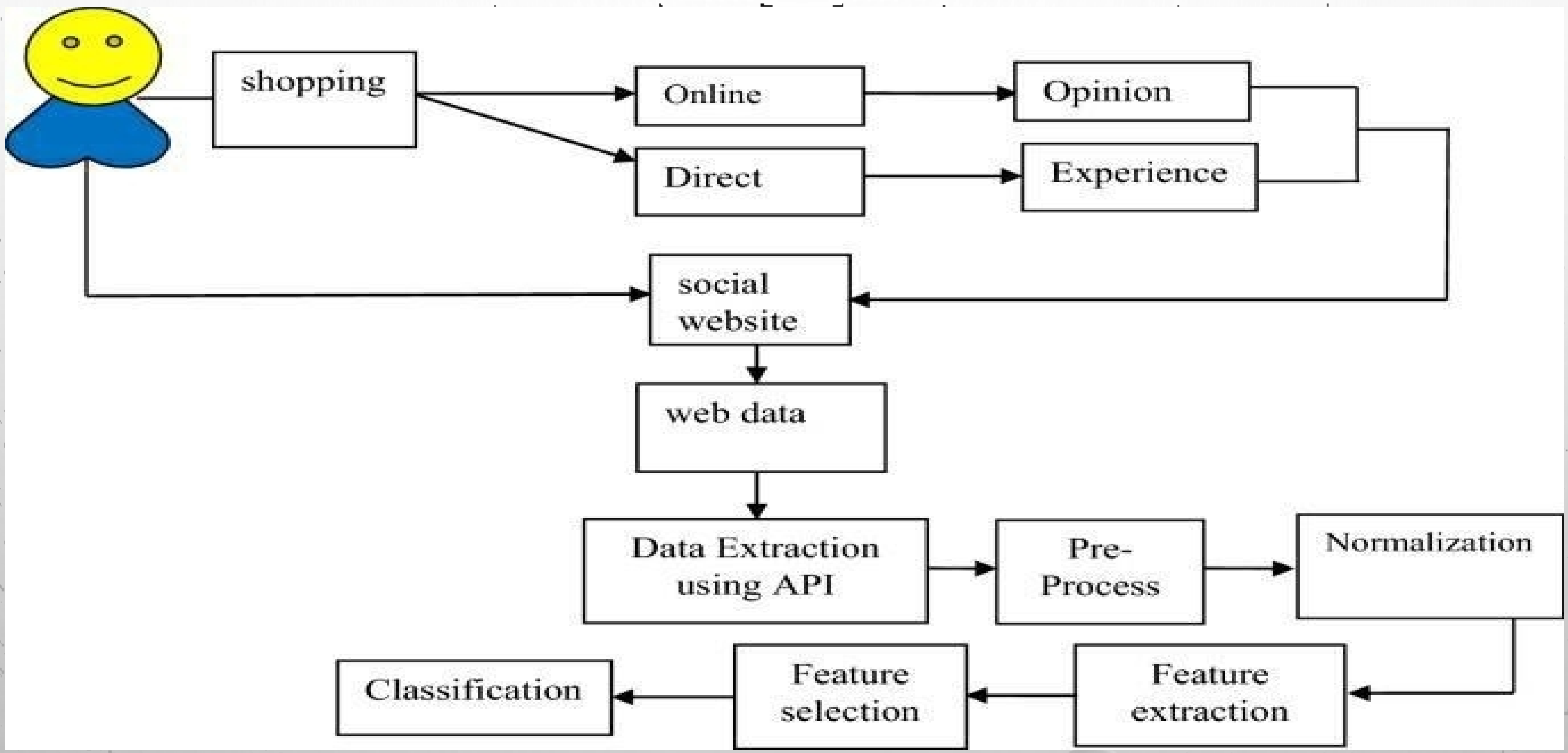
Sentiment Analysis is a Marketing

- SENTIMENT ANALYSIS IS A MARKETING TOOL THAT HELPS YOU EXAMINE THE WAY PEOPLE INTERACT WITH A BRAND ONLINE. THIS METHOD IS MORE COMPREHENSIVE THAN TRADITIONAL ONLINE MARKETING TRACKING, WHICH MEASURES THE NUMBER OF ONLINE INTERACTIONS THAT CUSTOMERS HAVE WITH A BRAND, LIKE COMMENTS AND SHARES. USING SENTIMENT ANALYSIS, YOU CAN LABEL INDIVIDUAL INTERACTIONS AS POSITIVE, NEGATIVE OR NEUTRAL. ONCE YOU'VE FIGURED OUT HOW TO DETERMINE AND TRACK THESE LABELS, YOU CAN USE THIS NEW DATA SET FOR A VARIETY OF MARKETING PURPOSES, INCLUDING YOUR ONLINE STRATEGY.

3 TYPES OF SENTIMENT ANALYSIS

- MANUAL ANALYSIS: THIS TYPE USES MANUALLY CREATED RULES BASED ON NEUROLINGUISTIC PRINCIPLES, SUCH AS STEMMING AND TOKENIZATION. IT TAKES A LONG TIME TO SET UP, BUT IT'S EASY TO CHANGE AND CUSTOMIZE.
- AUTOMATIC ANALYSIS: THIS TYPE USES MACHINE LEARNING TECHNIQUES THAT USE NEURAL NETWORKS AND STATISTICAL MODELS TO CLASSIFY LANGUAGE. IT CAN BE CHALLENGING TO CHANGE, BUT IT'S EASY TO SET UP AND MANAGE.
- HYBRID ANALYSIS: THIS TYPE USES BOTH RULES-BASED AND MACHINE-LEARNING ANALYSES. IT'S A BALANCED APPROACH THAT MOST SOCIAL LISTENING APPLICATIONS EMPLOY.

BLOCK DIAGRAM



SOURCE CODE

```
FROM TEXTBLOB IMPORT TEXTBLOB

# SAMPLE MARKETING TEXT DATAMARKETING_DATA = [ "I LOVE THIS PRODUCT, IT'S AMAZING!",                    "THE CUSTOMER
SERVICE IS TERRIBLE.",          "THIS AD CAMPAIGN IS SO CREATIVE AND ENGAGING.",       "THE PRODUCT QUALITY IS DISAPPOINTING.",          "I
HAD A GREAT EXPERIENCE WITH THIS BRAND.",]

# ANALYZE SENTIMENT FOR EACH TEXTSENTIMENTS = []FOR TEXT IN MARKETING_DATA:                    ANALYSIS =
TEXTBLOB(TEXT)                                        SENTIMENT = ANALYSIS.SENTIMENT.POLARITY #
RANGE FROM -1 (NEGATIVE) TO 1 (POSITIVE)              SENTIMENTS.APPEND(SENTIMENT)

# CLASSIFY SENTIMENT BASED ON POLARITYDEF CLASSIFY_SENTIMENT(SENTIMENT):

 IF SENTIMENT > 0:     RETURN "POSITIVE"

ELSE SENTIMENT < 0:     RETURN "NEGATIVE“

   ELSE:    RETURN "NEUTRAL"CLASSIFIED_SENTIMENTS = [CLASSIFY_SENTIMENT(S) FOR S IN SENTIMENTS]

# DISPLAY RESULTSFOR I, TEXT IN ENUMERATE(MARKETING_DATA):              PRINT(F"TEXT: {TEXT}")        PRINT(F"SENTIMENT:
{CLASSIFIED_SENTIMENTS[I]} (POLARITY:
{SENTIMENTS[I]:.2F})")     PRINT()

# CALCULATE OVERALL SENTIMENT DISTRIBUTIONPOSITIVE_COUNT = CLASSIFIED_SENTIMENTS.COUNT("POSITIVE")NEGATIVE_COUNT =
CLASSIFIED_SENTIMENTS.COUNT("NEGATIVE")NEUTRAL_COUNT = CLASSIFIED_SENTIMENTS.COUNT("NEUTRAL")PRINT(F"OVERALL SENTIMENT DISTRIBUTION:")

PRINT(F"POSITIVE:

{POSITIVE_COUNT}")

PRINT(F"NEGATIVE:
```

Content for project phase 2

Explore advanced techniques like fine-tuning pre- trained sentiment analysis models (BERT, RoBERTa) for more accurate sentiment predictions.

Dataset Link: https://www.kaggle.com/datasets/crowdflower/twitter- airline-sentiment

BERT Embeddings

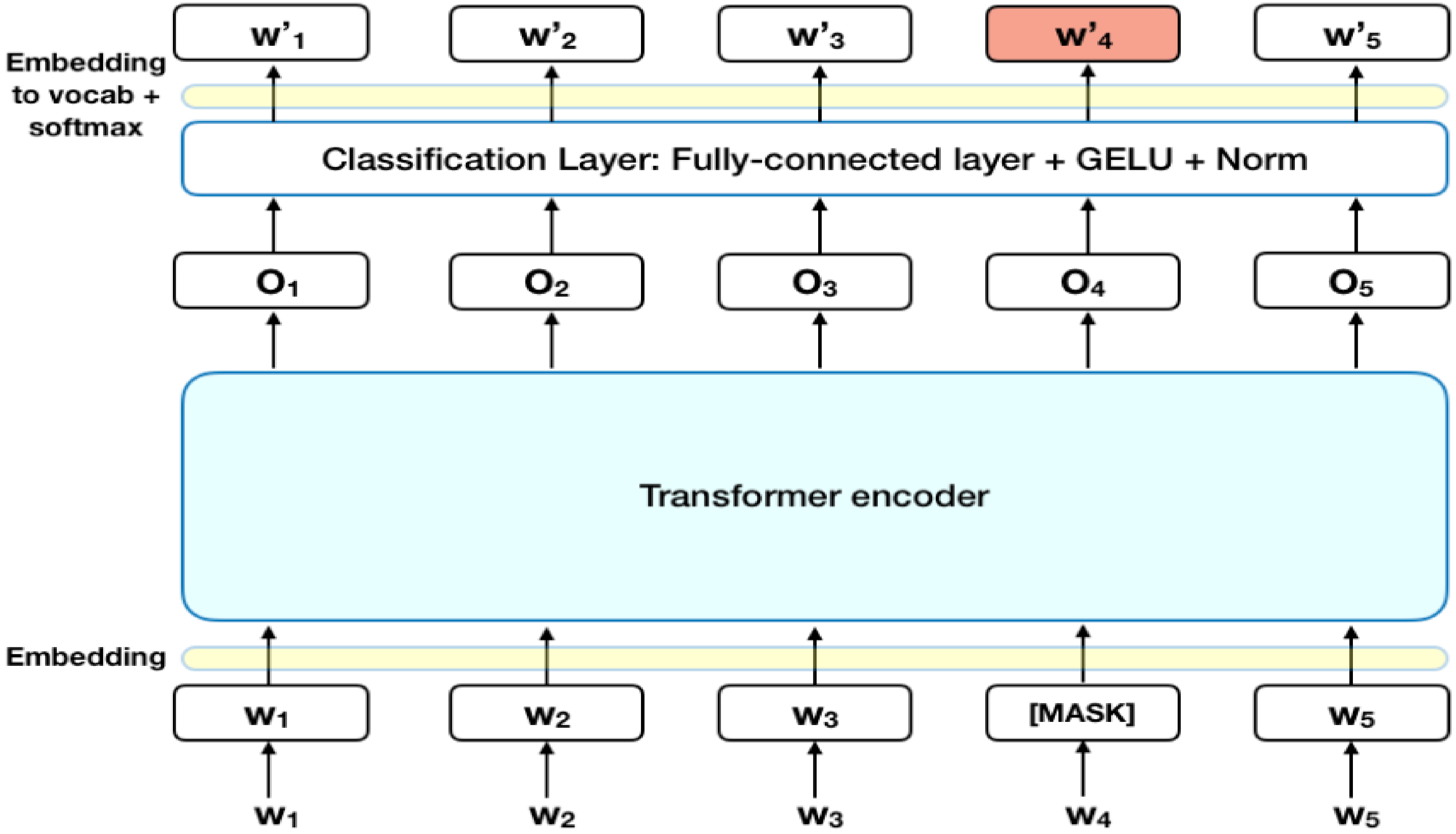
BERT is a traditional SOTA transformer architecture published by Google Research which uses bidirectional pretraining . The importance of using BERT is that it has 2 important aspects:

- Msked Language Model (MLM)
- Next Sentence Prediction(NSP)

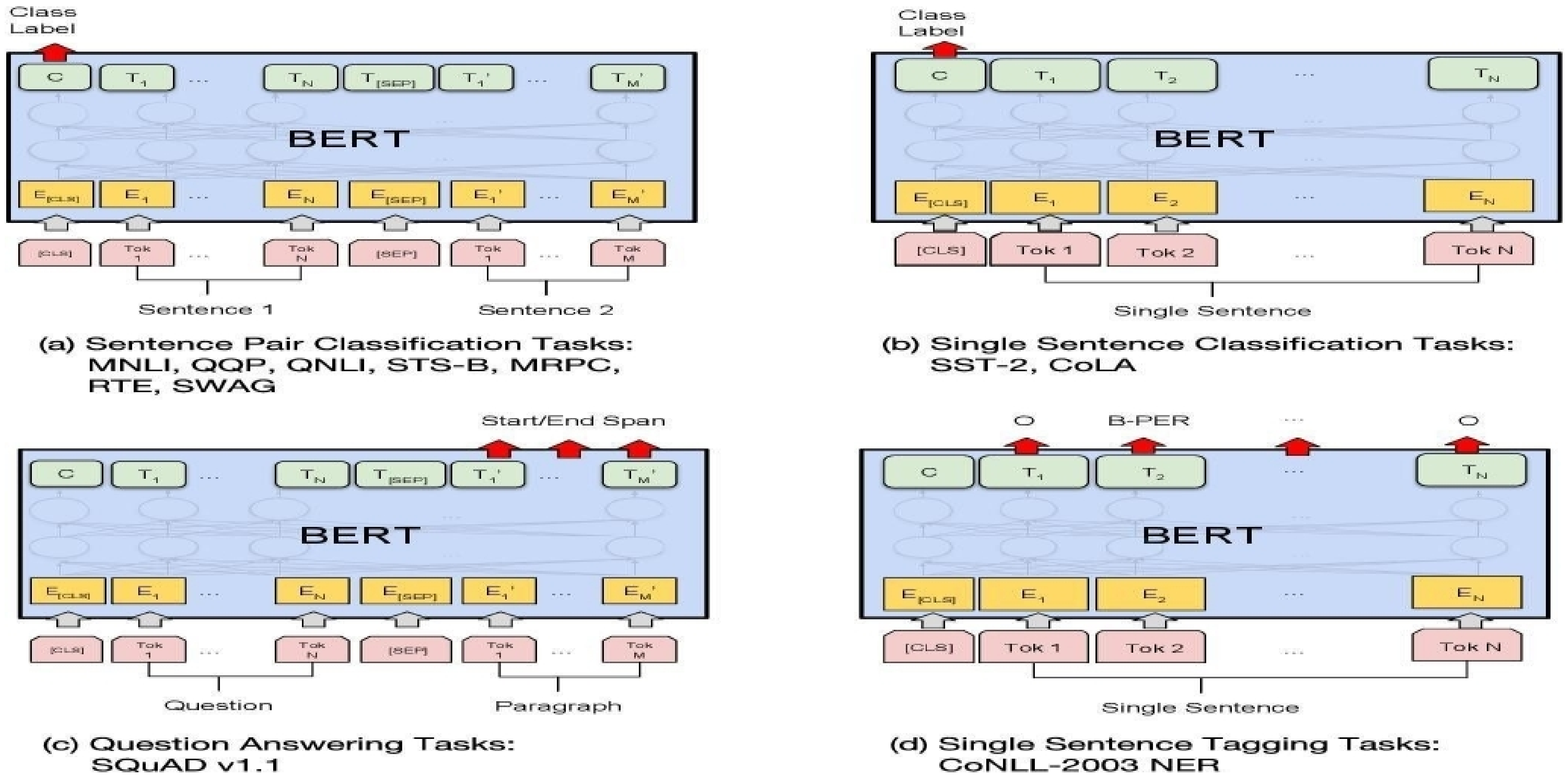
The bidirectional pre-training is essentially helpful to be used for any tasks. The Huggingface implementation is helpful for fine-tuning

BER

T for any language modelling task. The BERT architecture falls under an encoder- decoder(Transformer) model as


w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

For fine-tuning and pre-training for different downstream tasks like Q/A, Classification, Language Modelling, Multiple Choice, NER etc. different layers of the BERT are used.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Finetuning BERT for Embeddings

For finetuning, it is to be kept in mind, there are many ways to do this. We are using BERT from Huggingfacerepository while it can also be used from [TF-HUB](#) or from [Google-Research repository](#). The reason for using HuggingFace is that the same codebase is applicable for all language models. The 3 most important input features that any language model asks for is:

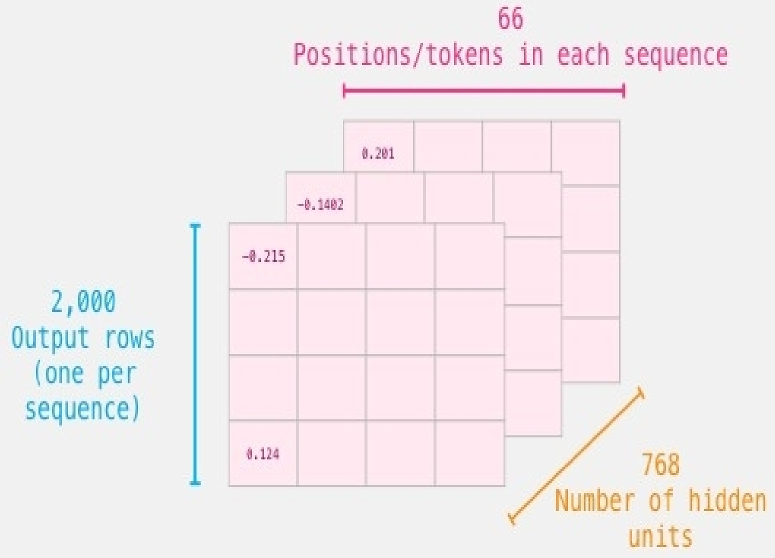
- *input_ids*
- *attention_masks*
- *token_ids*

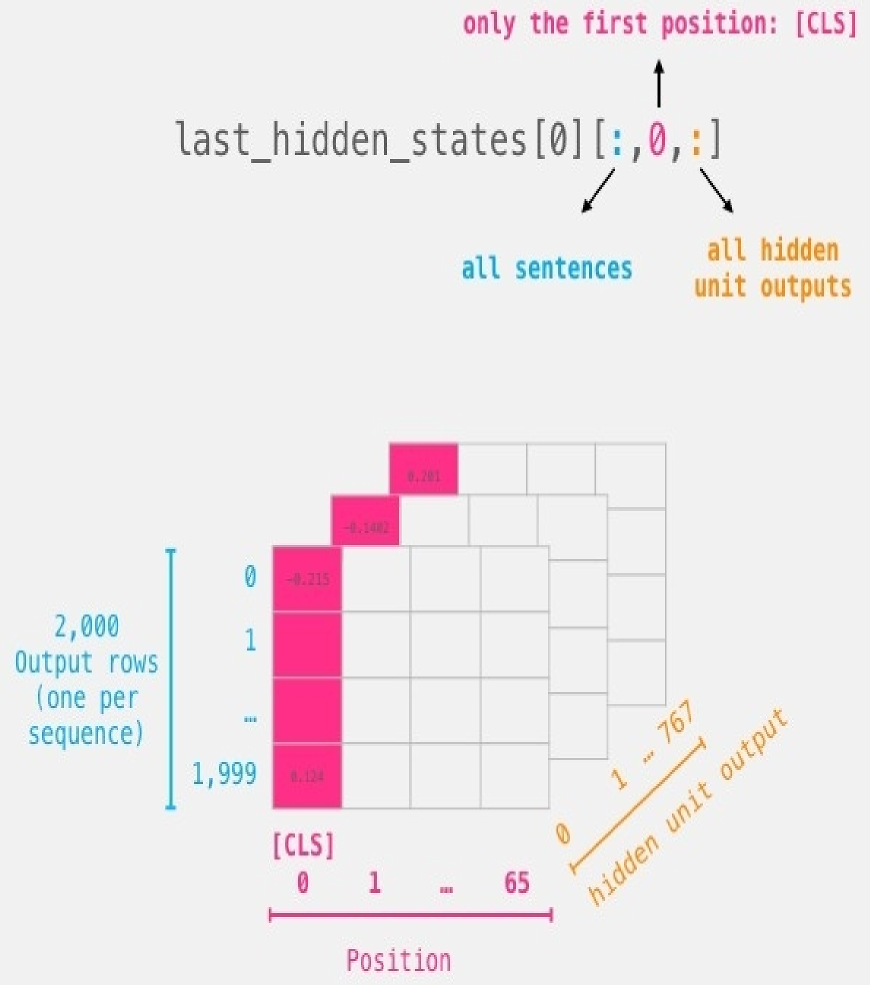
Let us first try to analyse and understand how BERT tokenizers, and model can be used in this context. The [BERT](#) documentation provides an outline of how to use BERT tokenizers and also modify it for downstream tasks.

Generally by virtue of transfer learning through weight transfer, we use pretrained [BERT models](#) from the list. This allows us to finetune it to extract only the embeddings. Since we are using Keras, we have to build up a small model containing an Input Layer and apply the tokenized(encoded) input ids, attention masks as input to the pretrained and loaded BERT model.This is very similar to creating a very own classification model for BERT using Keras/Tensorflow, but since we will be needing only the Embeddings it is safe to extract only the sentence vectors in the last layer of the model output. In most of the cases , we will see that the dimensions of the output vector is (x,768) where x depends on the number of tokenized input features. For this we extract the [CLS] tokenized feature from the ouput to just extract the sentence embeddings.

`last_hidden_states[0]`

BERT Output Tensor/predictions





Some important resources which may be helpful:

- Blog
- Extensive Nice Blog
- Good Kernel

```
In [1]:

#tokenize    and  encode   the  inputs

import transformers from transformers import
BertTokenizer,TFBertModel tokenizer =
transformers.BertTokenizer.from_pretrained('bert-large-
uncased', do_lower_case=True) bert_model =
transformers.TFBertModel.from_pretrained('bert-large-
uncased') def bert_encode(data,maximum_length):
 input_ids = []
 attention_masks = []
```

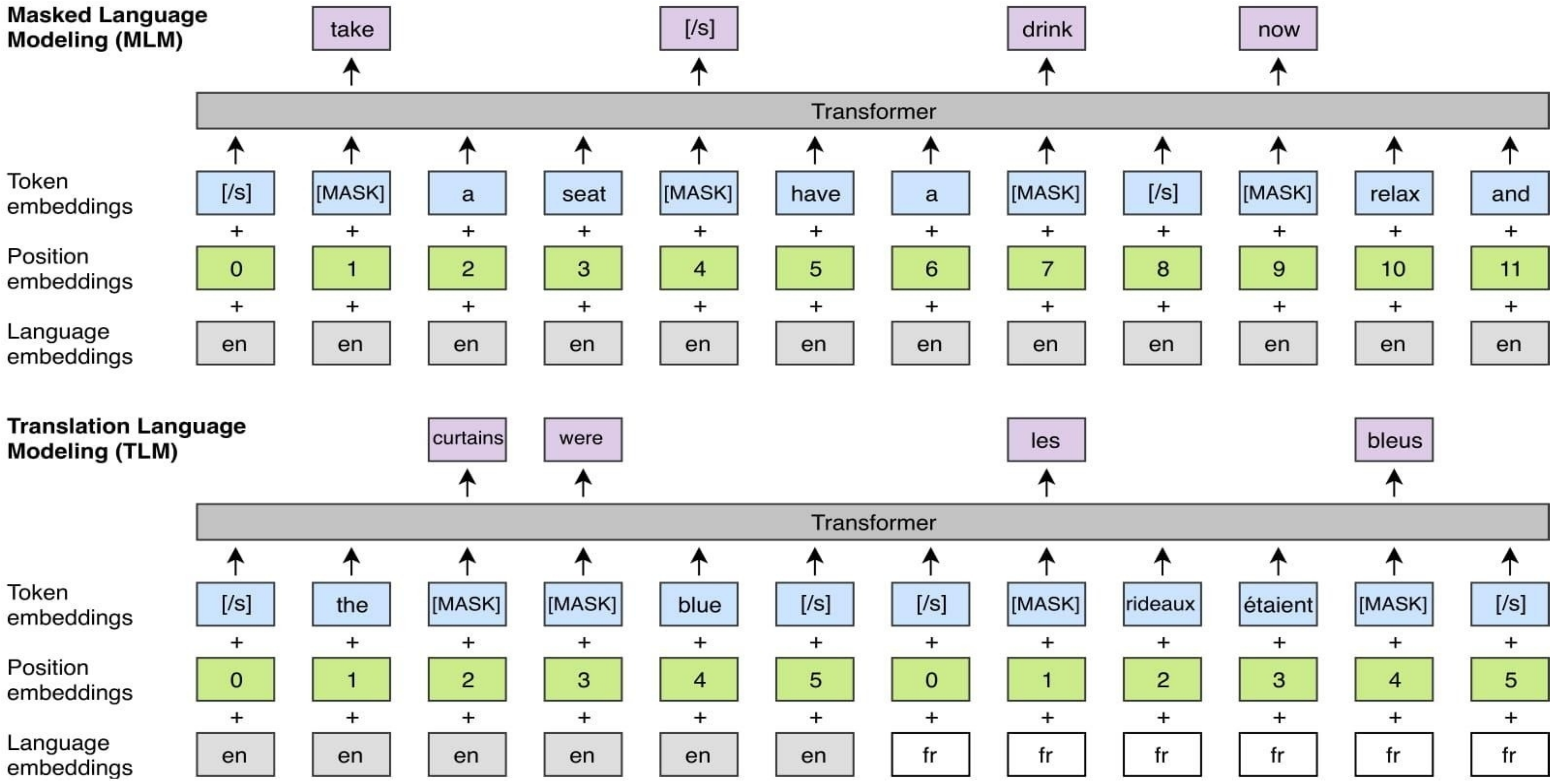
```
for i in
range(len(data)):
encoded = tokenizer.encode_plus(

   data[i],
   add_special_tokens=True,
   max_length=maximum_len
   gth,
   pad_to_max_length=True,

     return_attention_mask=True,

     )

     input_ids.(encoded['input_ids'])
     attention_masks.append(encoded['attention_m
     ask'])
       return
       np.array(input_ids),np.array(attention_masks)

     train_input_ids,train_attention_mas
     ks =
Some layers from the model checkpoint at bert-large-uncased were not used
when initializing TFBertModel: ['mlm__cls', 'nsp______cls']

 - This IS expected if you are initializing TFBertModel from the checkpoint of a model
trained on  another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).

- This IS NOT expected if you are initializing TFBertModel from the checkpoint of a
model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a  BertForSequenceClassification model).
All the layers of TFBertModel were initialized from the  model checkpoint at bert-
large-uncased.

If your task is similar to the task the model of the checkpoint was trained on, you can
already use
TFBertModel for predictions without further training.

Truncation was not explicitly activated but `max_length` is provided a specific value,
please use
`truncation=True` to explicitly truncate examples to max length. Defaulting to
'longest_first'  truncation strategy. If you encode pairs of sequences (GLUE-style) with
the tokenizer you can select this strategy more precisely by providing a specific
strategy to `truncation`.

/opt/conda/lib/python3.7/site-
packages/transformers/tokenization_utils_base.py:2142:  FutureWarning: The
```

Roberta Model

- Roberta Model is a robust and large model built by Facebook Research, to alleviate undertrained nature of BERT. It trains in much larger mini-batch sizes. This provides a good model of how to train Roberta on Google cloud.The original paper can be found here, and the model architecture is provided.

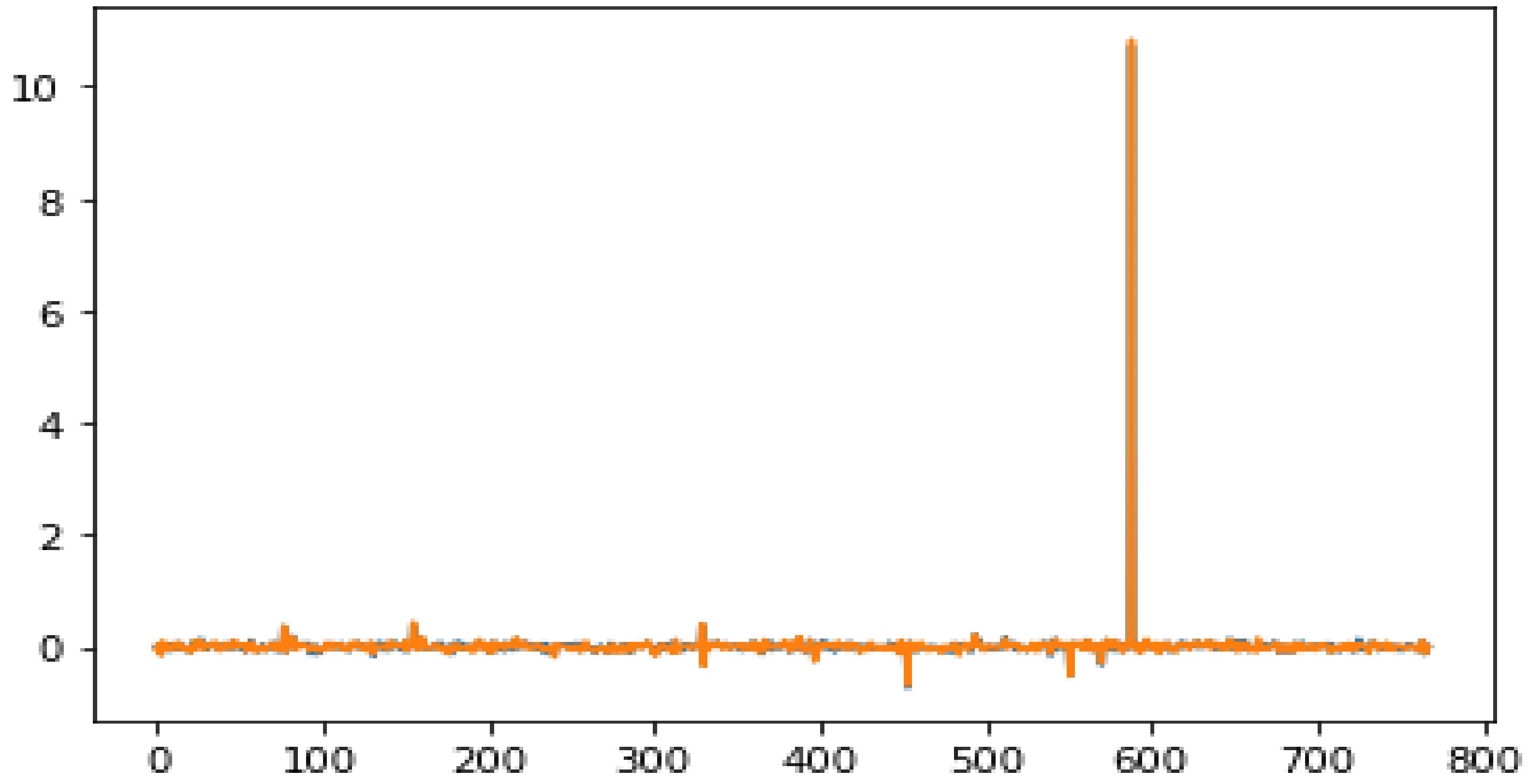
Resources:

- Blog
- Blog-2

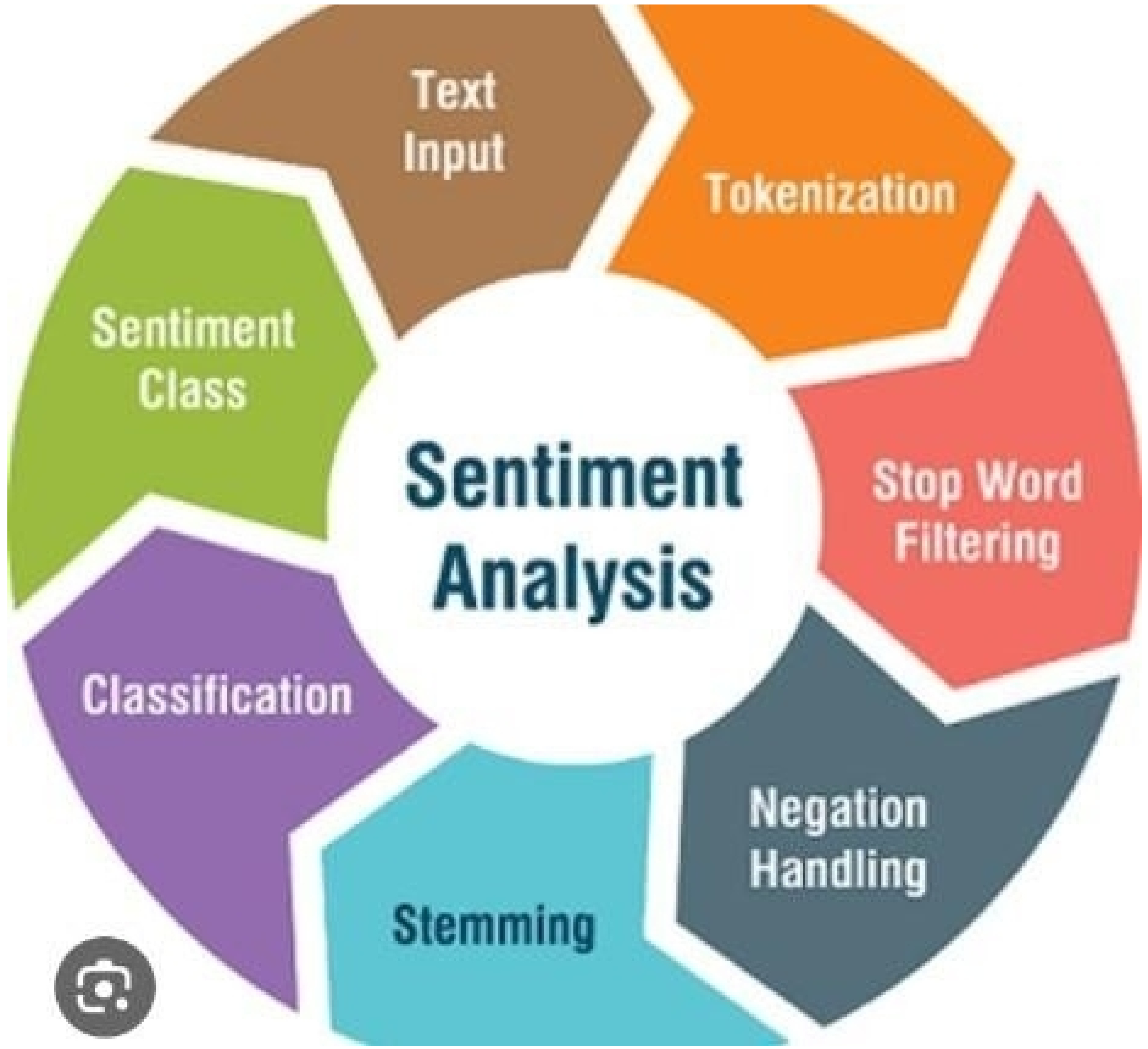

Masked Language Modeling (MLM)
take
[/s]
drink
now
Transformer
Token embeddings
[/s] [MASK] a seat [MASK] have a [MASK] [/s] [MASK] relax and
Position embeddings
0 1 2 3 4 5 6 7 8 9 10 11
Language embeddings
en en en en en en en en en en en en
Translation Language Modeling (TLM)
curtains
were
les
bleus
Transformer
Token embeddings
[/s] the [MASK] [MASK] blue [/s] [/s] [MASK] rideaux étaient [MASK] [/s]
Position embeddings
0 1 2 3 4 5 0 1 2 3 4 5
Language embeddings
en en en en en en fr fr fr fr fr fr

```
##Roberta Embeddings from transformers import
AutoTokenizer,  pipeline, TFRobertaModel
roberta_features1=transformer_embedding('roberta-
base',z[0],TFRobertaModel)
roberta_features2=transformer_embedding('roberta-
base',z[1],TFRobertaModel) distance=1-
cosine(roberta_features1[0],roberta_features2[0])
print(distance)  plt.plot(roberta_features1[0])
                                        plt.plot(roberta_features2[0])  plt.show
()
```

```
Some layers from the model checkpoint at roberta-base were
not used  when initializing TFRobertaModel: ['lm_head'] - This
IS expected if you  are initializing TFRobertaModel from the
checkpoint of a model trained  on another task or with another
architecture (e.g. initializing a  BertForSequenceClassification
This IS expected if you are initializing TFRobertaModel from
the  checkpoint of a model trained on another task or with
another  architecture (e.g. initializing a
BertForSequenceClassification model from a
BertForPreTraining model).
 - This IS NOT expected if you are initializingTFRobertaModel
from the checkpoint of a model that you expect to be exactly
identical  (initializing           a BertForSequenceClassification
                model from a  BertForSequenceClassification
                model).
All the layers of TFRobertaModel were initialized from
the model checkpoint at roberta-base.
If your task is similar to the task the model of the
checkpoint was trained on, you can already use
TFRobertaModel for predictions  without further training.
0.9966970127889705
```



10
8
6
4
2
0
0
100
200
300
400
500
600
700
800


Sentiment Analysis
Text Input
Tokenization
Stop Word Filtering
Negation Handling
Stemming
Classification
Sentiment Class

Introduction

- Sentiment analysis, also referred to as opinion mining, is an approach to natural language processing (NLP) that identifies the emotional tone behind a body of text. This is a popular way for organizations to determine and categorize opinions about a product.

Examples for sendiment analysis

Sentiment analysis is a powerful tool, which can be used across industries and teams. Learn about some of the most popular sentiment analysis business applications, below:

- [Social media monitoring]()
- [Brand monitoring]()
- [Customer support analysis]()
- [Customer feedback analysis]()
- [Market research]()

Data Preprocessing :

Our data generally comes from a variety of different sources and is often in a variety of different formats. For
this reason, cleaning our raw data is an essential part of preparing our dataset. However,
cleaning is not a simple process, as textual data often contains redundant and/or repetitive
words.

Before training the model, we will perform various pre-processing steps on the dataset such as:

- Removing stop words.
- Removing emojis.
- Removing of mentions.
- Removal of numbers.
- Removal of whitespaces.
- Removal of duplicated rows.
- Removal of unuseful columns.
- Converting the text document to lowercase for better generalization.
- Cleaning the ponctuation (to reduce unnecessary noise from the dataset).
- Removing the repeating characters from the words along with removing the URLs/hyperlinks as they do not have any significant importance.

  and much more, we will see this in detail later...

We will then performe:

Stemming : reducing the words to their derived stems.

Lemmatization : reducing the derived words to their root form known as lemma for better

- **Lowering Case**:

Lowering case is very imprtant since it allows us to make words with same value equal. This will be very useful to reduce the dimensions of our vocabulary.

```
# Lowering Case :
 print("=========== Before Lowering case ============\n")
print("\t" + df.loc[10, "text"])
 print("\n=========== After Lowering case ===========\n")
df['text'] = df['text'].str.lower()
print("\t" + df.loc[10, "text"])
=========== Before Lowering case ============ spring break in
plain city... it's snowing
```

```
=========== After Lowering case ===
=========
spring break in plain city... it's snowing
```

Lower case was successfully applied to our data **Removal of Mentions:**

In social media, Mentions are used to call/mention another user into our post. Generally, mentions don't have an added value to our model. So we will remove them.

A mention has a special pattern: **@UserName**, So we will remove all string which starts with @

- *# Removal of Mentions:*

```
## Creating a fucntion that will be applied to our
datset :
def RemoveMentions(text):
text_ = re.sub(r"@\S+", "",
text) return text_
## Applying the function to each row of the data
print("============ Before
Removing M entions
============\n")
print("\t" + df.loc[5, "text"])
print("\n============ After
Removing Mentions
============\n")
df["text"] =
=========== Before Lowering case ===
=========
spring break in plain city... it's snowing
 =========== After Lowering case ===
========
spring break in plain city... it's snowing
```

Lower case was successfully applied

to our data Removal of Mentions:

In social media, Mentions are used to call/mention another user into our post. Generally, mentions don't have an added value to our model. So we will remove them.

A mention has a special pattern: **@UserName**, So we will remove all string which starts with @

```
# Removal of Mentions:
 ##        Creating  a   fucntion       that  will  be    applied    to
   our     datset    :
 def RemoveMentions(text):
text_ = re.sub(r"@\S+", "", text)
return text_
##         Applying    the     function  to   each row   of    the    data
   print("=========== Before Removing M
entions ============\n")
print("\t" + df.loc[5, "text"])
print("\n=========== After Removing
Mentions ===========\n")
 df["text"] = df["text"].apply(Remove
Mentions) print("\t" + df.loc[5, "text"])
=========== Before Removing Mentions
 ============
@kwesidei not the whole crew
 =========== After Removing Mentions ===========
not the whole crew
```

Removal of Mentions was successfully applied to our data Removal of Special Characters:

Special characters are every where, since we have punctuation marks in our tweets. In order to treat, for example, **hello!** and **hello** in the same way. we have to remove the punctuation mark

```
 # Defining a list containing punctuation signs of
 english :
 punctuations_list = string.punctuation
## Defining that will be applied to our datset :
 def  RemovePunctuations(text):
transformator  =  str.maketrans
('', '', punctuations_list)
 return  text.translate(transformator)
```

```
##Applying the fucntion to all rows : print("===========
Before Removing Punctuations =============\n")

print("\t" + df.loc[10, "text"]) print("\n===========
After Removing
 Punctuations \===========\n")

 df["text"] = df["text"].apply(RemovePunctuations)
print("\t" + df.loc[10."text"])

=========== Before Removing Punctuations =============
spring break in plain city...
it's snowing

 =========== After Removing Punctuations \===========

 spring break in plain city its snowing
```

- **Removal of Stop words**:
- Stopwords are the most common words in any natural language. For the purpose of analyzing text data and building NLP models, these stopwords might not add much value to the meaning of the document.
- Generally, the most common words used in a text are "the", "is", "in", "for", "where", "when", "to", "at" etc.
- Consider this text string – "There is a pen on the table". Now, the words "is", "a", "on", and "the" add no meaning to the statement while parsing it. Whereas words like "there", "book", and "table" are the keywords and tell us what the statement is all about.

Loading the Dataset :

In order to build our classifier model, we need a dataset which contains a huge number of tweets and the corresponding feeling being expressed at.

In any project related to the manipulation and analysis of data, we always start by collecting the data on which we are going to work. In our case, we will import our data from a .csv file. The dataset provided is the Sentiment140 Dataset which consists of 1,600,000 tweets that have been extracted using the Twitter API.

The various columns present in the dataset are:

target: the polarity of the tweet (positive or negative)

ids: Unique id of the tweet

date: the date of the tweet

flag: It refers to the query. If no such query exists then it is NO QUERY.

user: It refers to the name of the user that tweeted

text: It refers to the text of the tweet

Inpu t:

```
# Importing the dataset :
DATASET_COLUMNS=['target','ids','date','flag','use
r','text'] DATASET_ENCODING = "ISO-8859-1”

Df=pd.read_csv('../input/tweets/training.1600000.processed.n
oemotic on.csv', encoding=DATASET_ENCODING,
                                                  names=DATASET_COLU
MNS)

# Display of the first 5 lines : df.sample(5)
```

Output

	Target	Ids	Date	Flag	User	Text
553277	0	2203589156	Wed Jun 17 00:04:43 PDT 2009	NO_QUERY	Ausste	Art Center assignment due in two weeks, Im fee...
1077806	4	1967737627	Fri May 29 20:07:22 PDT 2009	NO_QUERY	trishysabel	theres crickets in the pepsi center. u guys ...
784549	0	2324197774	Thu Jun 25 02:32:28 PDT 2009	NO_QUERY	aamy23	wishing i was at the coast, sitting by the poo...
1157521	4	1979188875	Sun May 31 01:59:52 PDT 2009	NO_QUERY	poposkidimitar	Leaving Zadar, heading to Zagreb this afternoon
197871	0	1971093961	Sat May 30 06:11:43 PDT	NO_QUERY	MYCBA	in london at last. missed the connecting fligh...

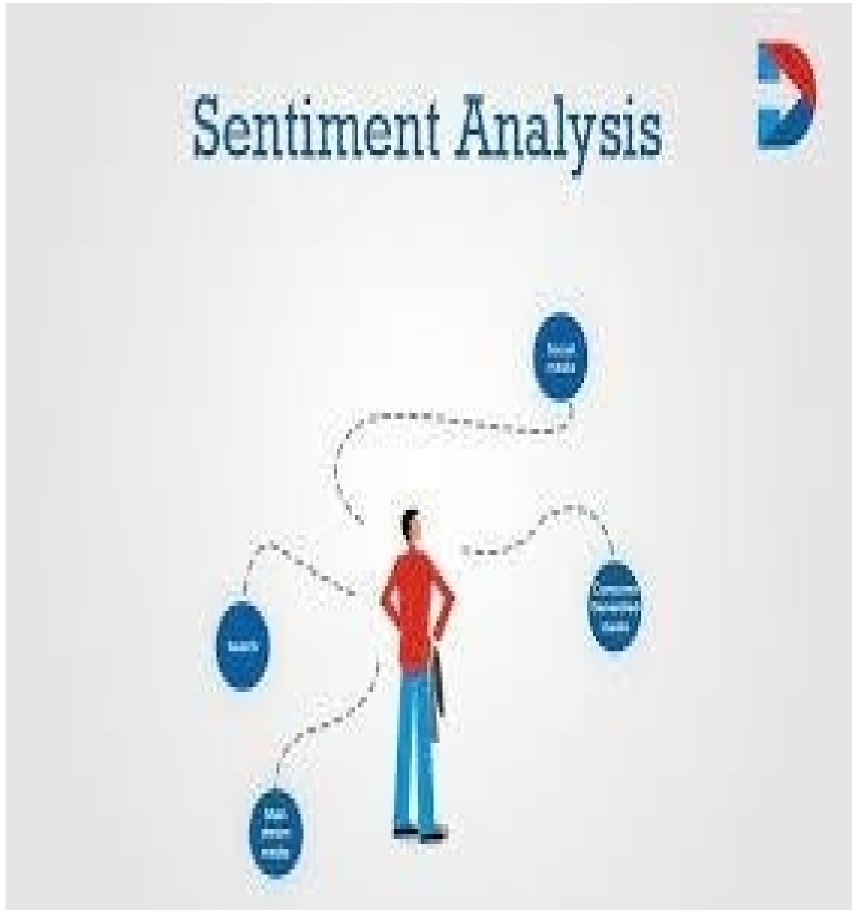
Conclusion

Sentiment analysis is a technique used to understand the emotional tone of the text. It can be used to identify positive, negative, and neutral sentiments in a piece of writing.

SENTIMENT ANALYSIS

Sentiment analysis is an automated process that attaches an emotional label or subjective opinion to text. For example, sentiment analysis may examine a social media post and determine that it carries a positive, negative or neutral opinion

"This was great!" "I had a horrible experience." The sentiments in these sentences can be inferred from certain words, such as "great" and "horrible". By analyzing the sentiments of reviews, feedback, and other customer interactions, businesses can improve their marketing campaigns.



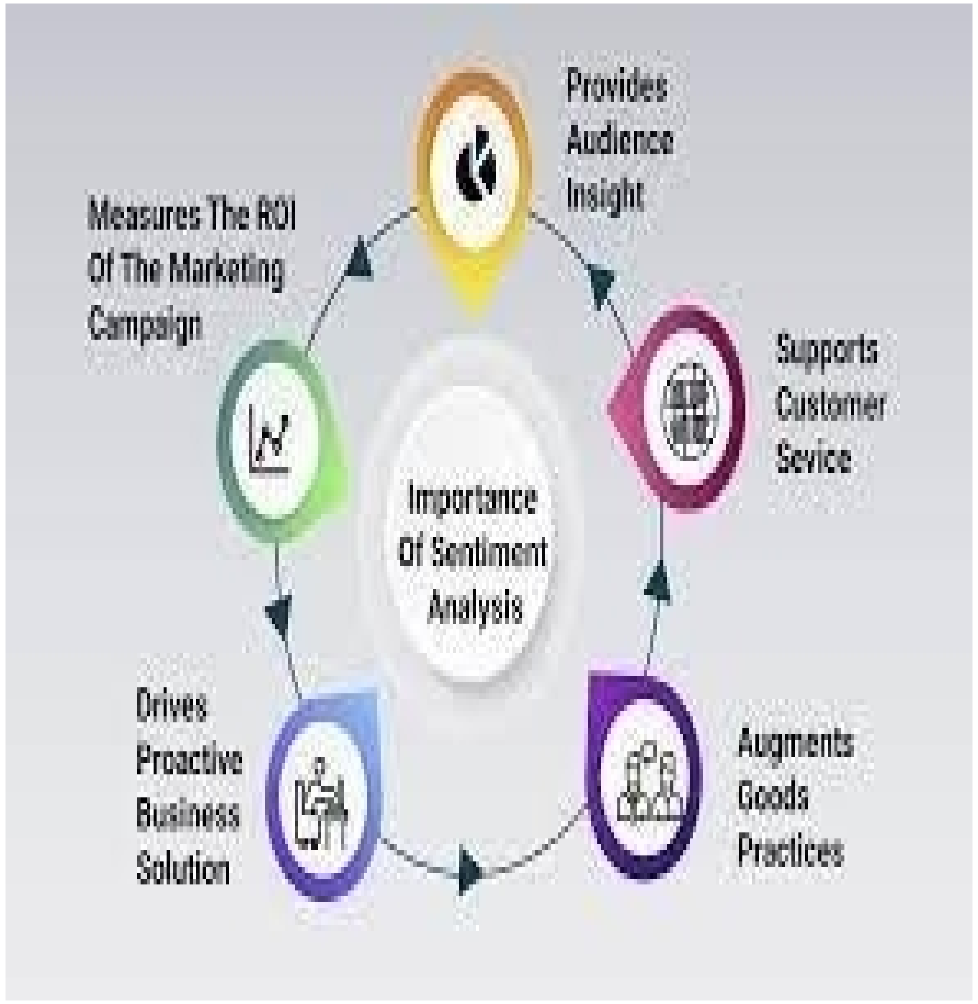
ROLE OF SENTIMENT ANALYSIS IN TRADING

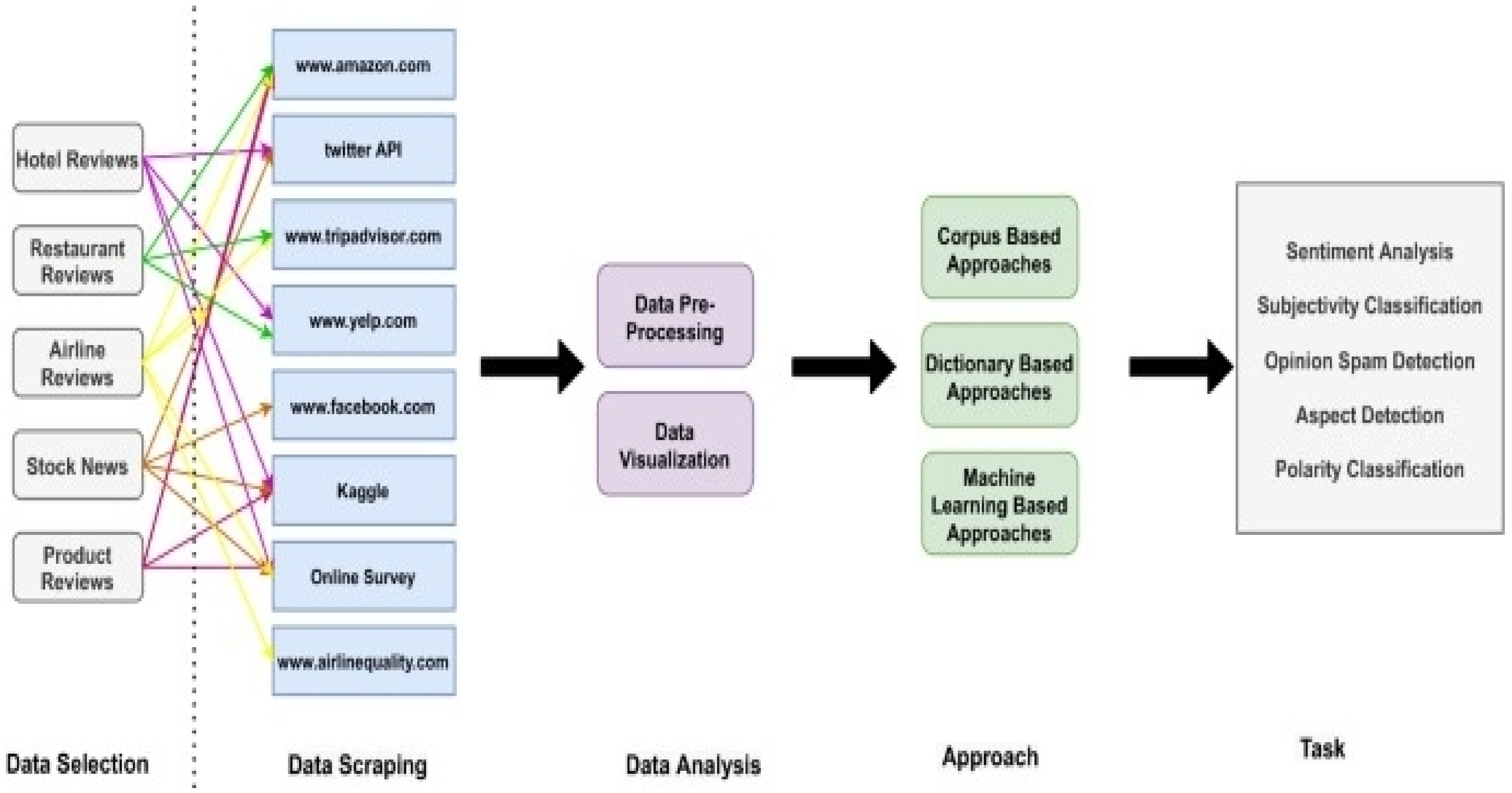
Sentiment analysis plays a significant role in trading by providing valuable insights into market sentiment and helping traders make informed decisions. Here are some key roles of sentiment analysis in trading:

- Market sentiment gauging
- Captures sentiment shifts
- News and event impact assessment
- Risk management
- Algorithmic trading strategies and Quantitative trading strategies

Adjusting messaging and product development

Sentiment analysis is an inexpensive way to improve messaging and product development. Knowing what customers value about a product or service can tell you what to emphasize in your promotional material. For example, if there's been a sudden and unexplained spike in sales of a certain product, you can check your positive mentions to see what customers are saying. You may find one of your products has suddenly become popular when someone posted about a feature that isn't in your other products.


Provides Audience Insight
Supports Customer Sevice
Augments Goods Practices
Drives Proactive Business Solution
Measures The ROI Of The Marketing Campaign
Importance Of Sentiment Analysis


Hotel Reviews
Restaurant Reviews
Airline Reviews
Stock News
Product Reviews
www.amazon.com
twitter API
www.tripadvisor.com
www.yelp.com
www.facebook.com
Kaggle
Online Survey
www.airlinequality.com
Data Pre-Processing
Data Visualization
Corpus Based Approaches
Dictionary Based Approaches
Machine Learning Based Approaches
Sentiment Analysis
Subjectivity Classification
Opinion Spam Detection
Aspect Detection
Polarity Classification
Data Selection
Data Scraping
Data Analysis
Approach
Task

USES OF SENTIMENT ANALYSIS

- Sentiment analysis can do wonders for any marketer. By understanding what your target audience is thinking on a scale that only sentiment analysis can achieve, you can tweak a product, campaign, and more, to meet their needs and let your customers know you're listening
- Sentiment analysis is an artificial intelligence technique that uses machine learning and natural language processing (NLP) to
  analyze text for polarity of opinion (positive to negative). It's one of
  the hardest tasks of natural language processing but, with the right tools, you can gain in-depth insights from social media conversations, online reviews, emails, customer service tickets, and more

TOOLS FOR SENTIMENT ANALYSIS MARKETING

There are several useful and dynamic sentiment analysis tools out there that can make sentiment marketing easy and cost-effective.

- Monkey Learn
- Brand watch
- Meltwater
- Social Searcher
- Repu state
- Hootsuite

APPLICATIONS

Below are some of the top applications to help increase customer acquisition, improve customer service, and keep your clientele happy:

- Social media monitoring
- Analyze marketing campaign success
- Gauge consumer sentiment around a new product launch
- Keep an eye on your competition
- Prevent PR crises
- Market research
- Identify influencers

BENEFITS

- Social Media Sentiment Analysis
- Brand Experience Insights
- Patient Insights
- Improve Customer Service
- Multilingual Insight
- News Trend Analysis
- Real-Time Sentiment Insights
- Customer Feedback