



**Google Developer Groups**  
Cummins College of Engineering,Pune

X



# SYNAPSE 2.0

## Co-Teaching Session

## DAY 2

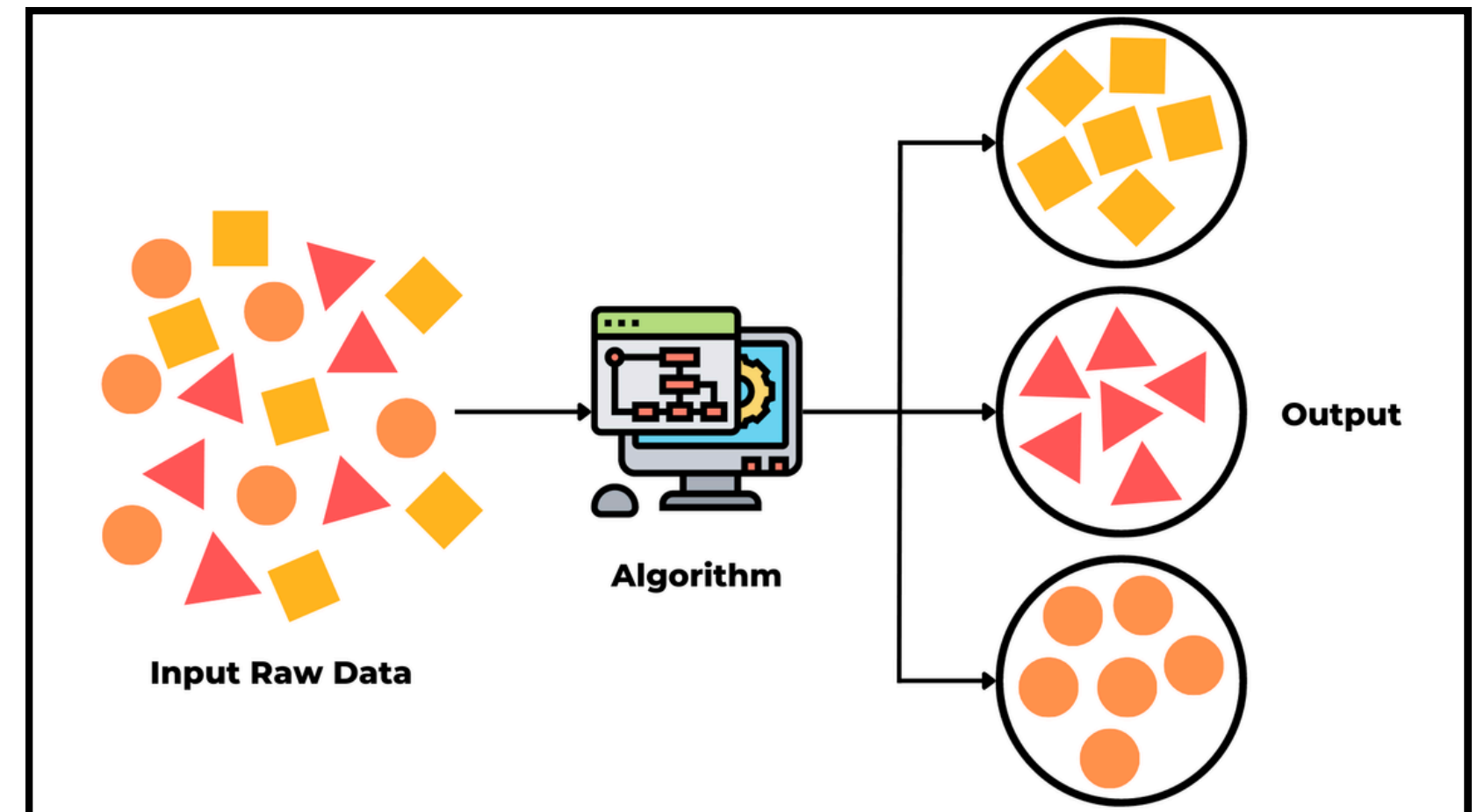
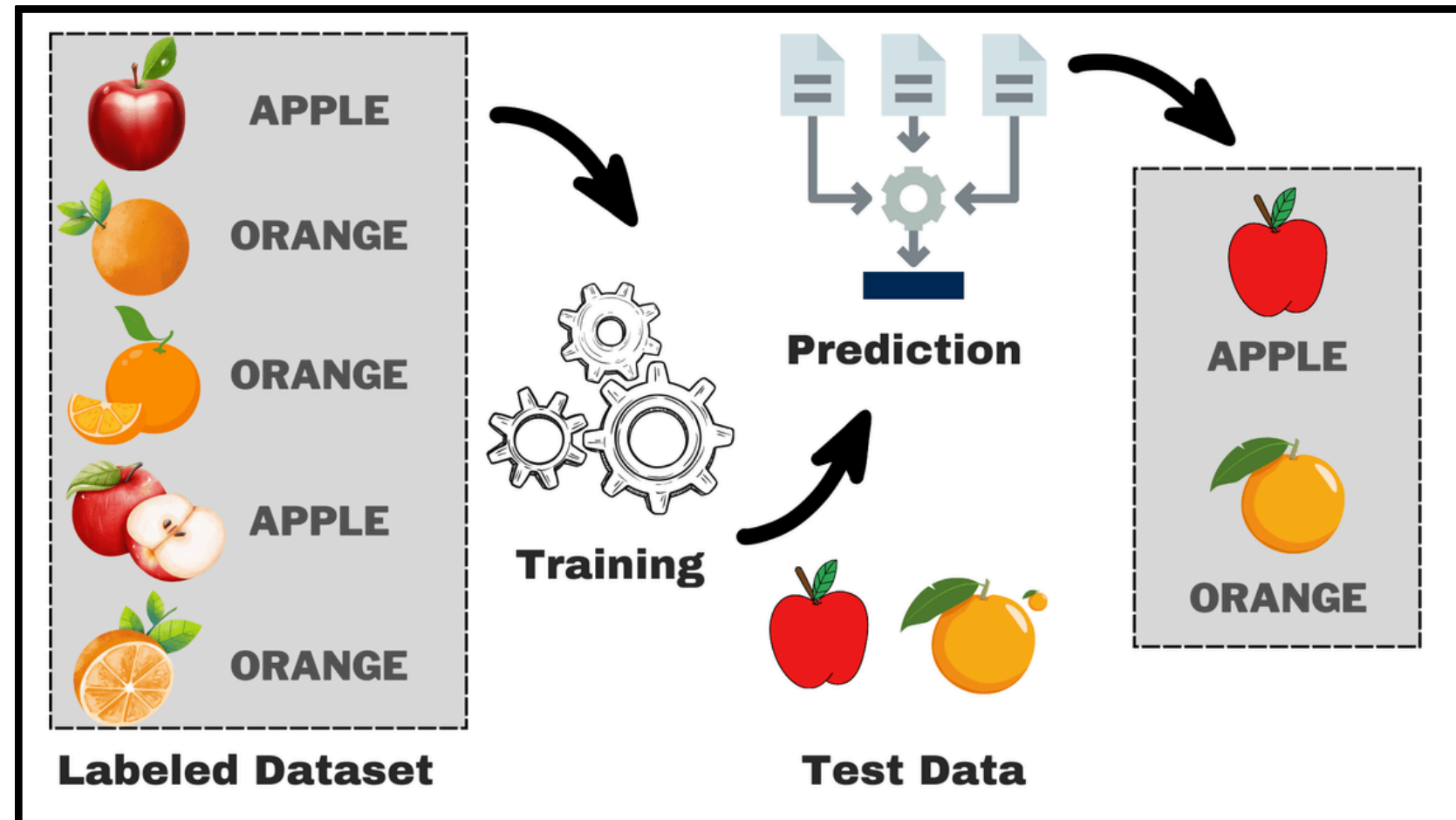
# QUIZ TIME

---



# Recap of Day 1

- ML: Supervised vs Unsupervised Learning



# Recap of Day 1

- Probabilistic Approach: Naive Bayes Theorem
- ML vs DL with respect to NLP
- Important NLP libraries for text preprocessing
- Techniques like Tokenization, stopwords removal, stemming and lemmatization, and punctuation removal
- Bag of Words
- TF-IDF
- N-grams
- Vectorization

# Table of contents

01

Sentiment  
Analysis

02

Pretrained  
Models

03

Prerequisites

04

TextBlob

05

VADER

06

BERT



# WHAT IS SENTIMENT ANALYSIS

---

# Types of Sentiments

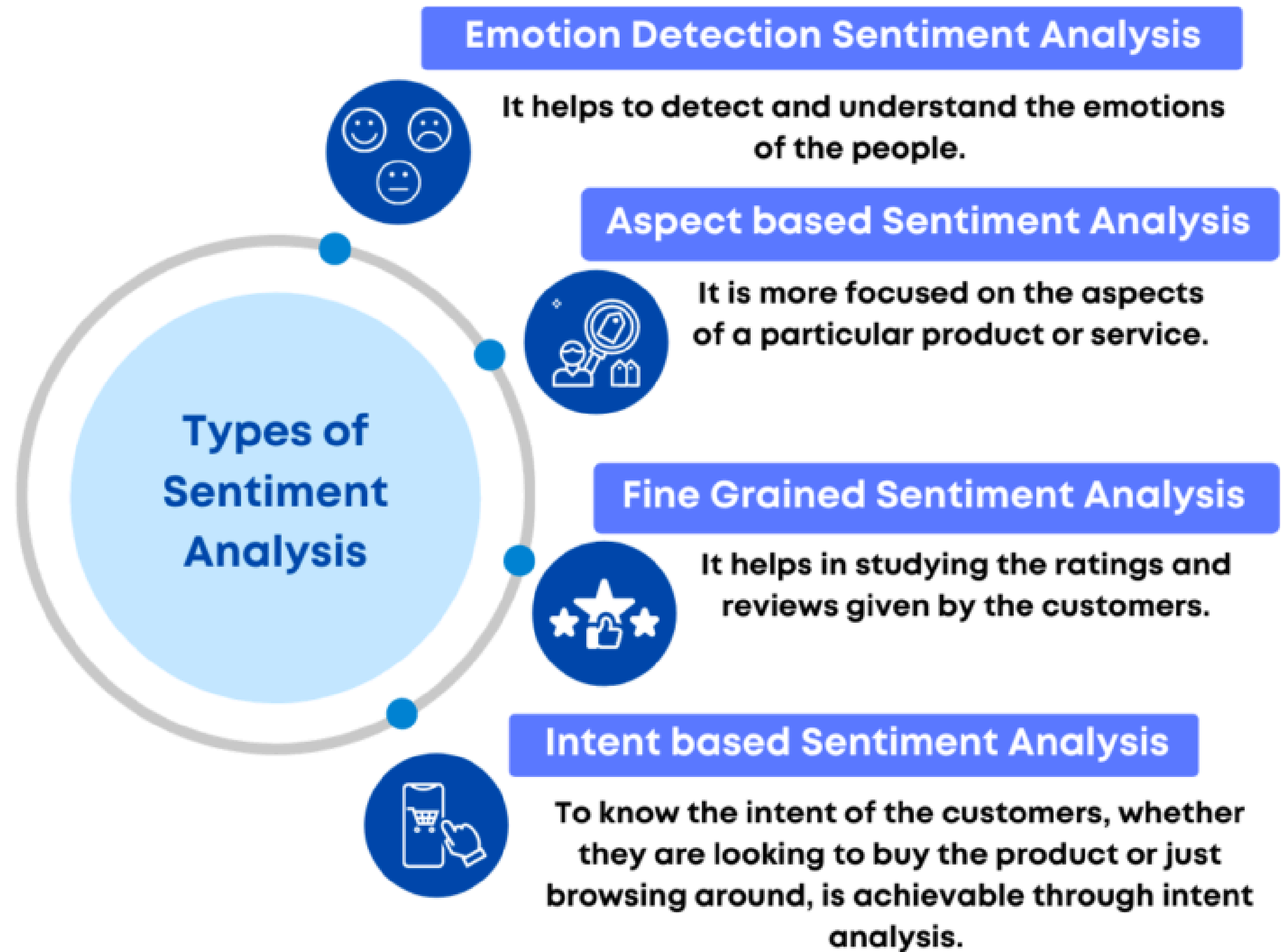
Types of sentiment analysis models-

3 principle sentiments-positive negative and neutral

But, we can even break these principal sentiments(positive, negative and neutral) into smaller sub sentiments such as “Happy”, “Love”, ”Surprise”, “Sad”, “Fear”, “Angry” etc. as per the needs or business requirement.

We can use any classification algorithm from ML to do this. For that we need to vectorize the data using the techniques you learnt before- can you guys tell me which ones?







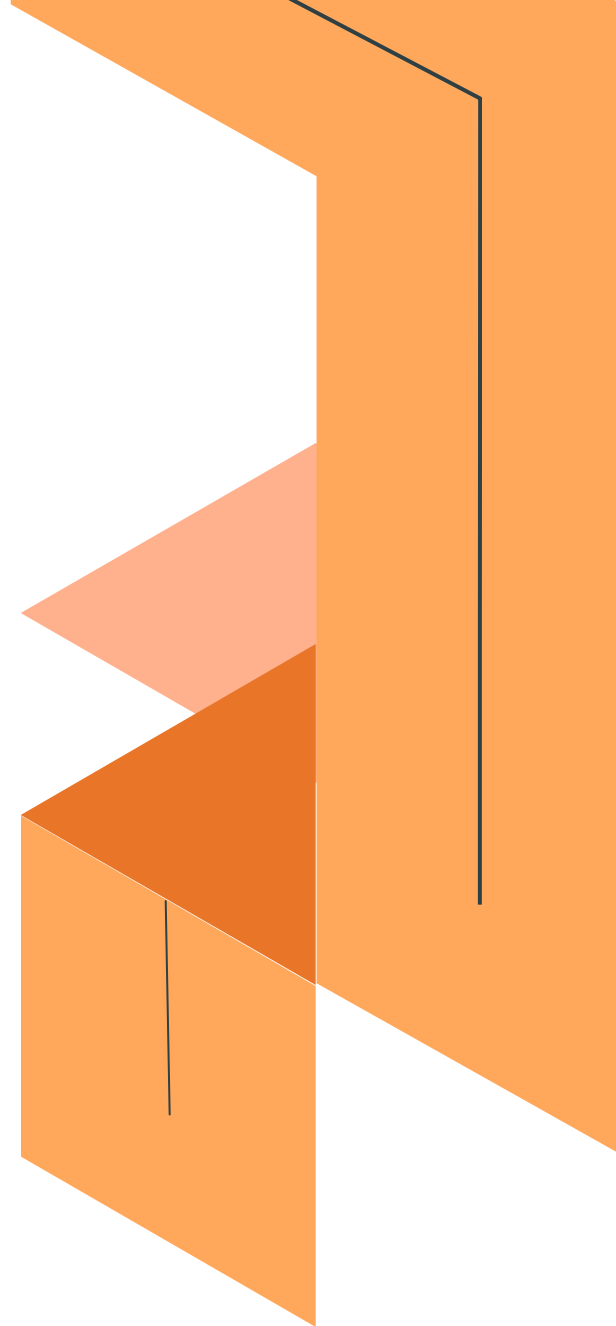
# Automated or rule-based?

## **Automated/Machine learning**

- Rely on having labelled historical data
- Might take a while to train
- Latest machine learning models can be quite powerful

## **Rule/lexicon-based**

- Rely on manually crafted valance scores
- Different words might have different polarity in different contexts
- Can be quite fast





# INTRO TO PRETRAINED MODELS

---

# What are pretrained models

Imagine you're learning a new language. What's easier — starting from scratch or learning from someone who already knows the language?

That's exactly the idea behind pretrained models! Instead of training an NLP model from the ground up, we use models that have already been trained on massive datasets.

## Examples of Popular Pretrained Models:

- Word Embeddings: Word2Vec, GloVe, FastText
- Transformers: BERT, GPT, T5, RoBERTa

# What is a Corpus?

- A corpus is a collection of authentic text or audio organized into datasets.
- Authentic here means text written or audio spoken by a native of the language or dialect.
- A corpus can be made up of everything from newspapers, novels, recipes, radio broadcasts to television shows, movies, and tweets.
- Corpora are the material basis for NLP to build any tool.

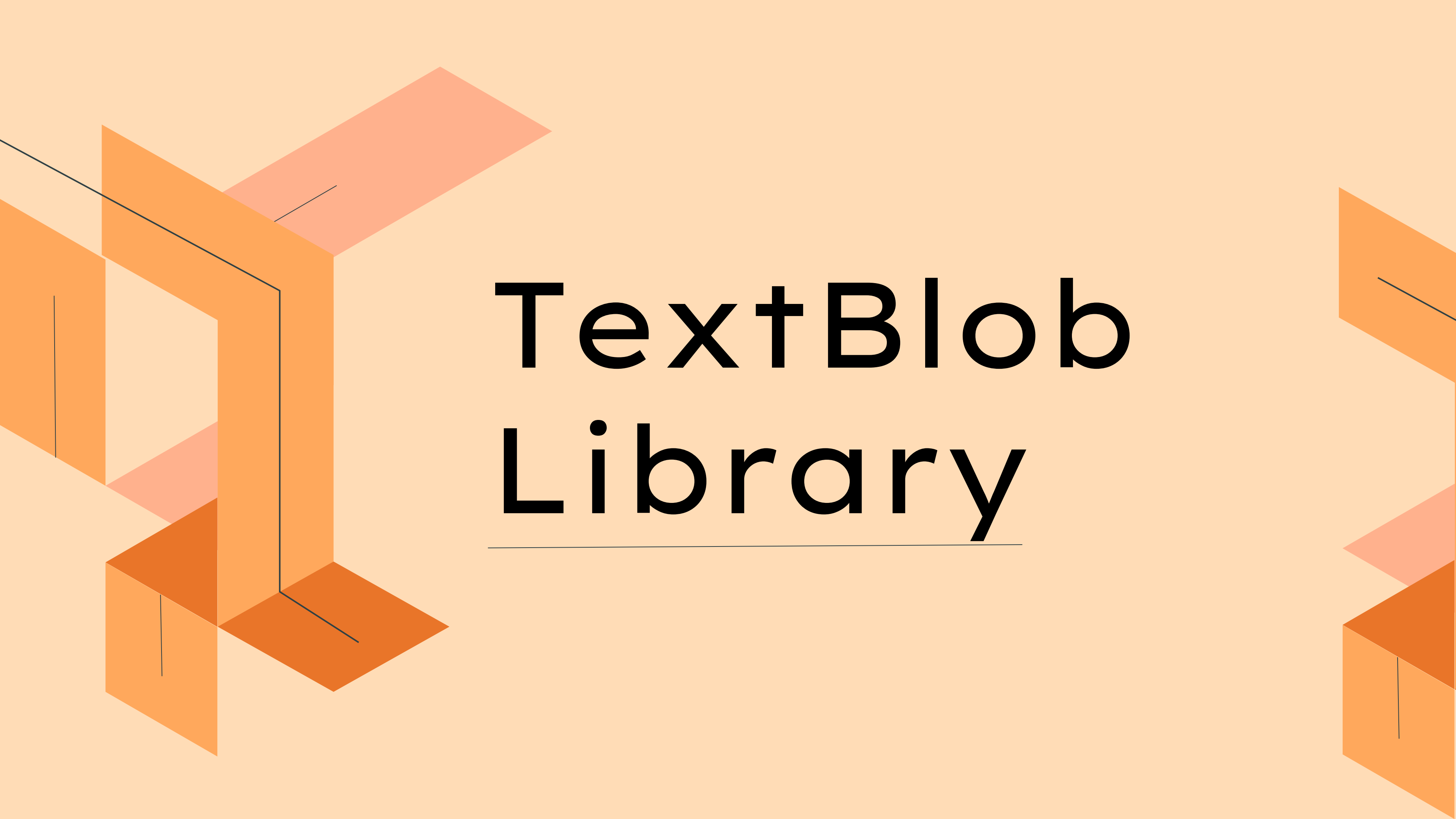


# WHY PRETRAINED MODELS?

---

# Advantages of pretrained models

- Reduces the computational burden
- The learned knowledge can be used for various applications.
- Models can be fine-tuned according to the task
- Less labelled data is required for fine-tuning specific tasks.



# TextBlob Library

---



# TextBlob

It's an NLP library in Python that makes tasks like tokenization, POS-Tagging, Words inflection, Noun phrase extraction, lemmatization, N-grams, and sentiment Analysis super easy.

TextBlob has a few more features than NLTK, such as Spelling correction, Creating a summary of a text, Translation, and language detection.

# TextBlob


```
print(blob.sentiment)
```

Polarity ranges from -1 to 1:

- -1 → Completely negative sentiment
- 0 → Neutral sentiment
- +1 → Completely positive sentiment

Subjectivity ranges from 0 to 1:

- 0 → Very objective (factual statements)
- 1 → Highly subjective (opinion-based, personal feelings)



# VADER

---

Valence Aware Dictionary and sEntiment  
Reasoner

# About VADER

Vader (Valence Aware Dictionary and Sentiment Reasoner)

It is a rule-based sentiment analysis tool that is **specifically designed for analyzing social media texts** and **requires no preprocessing!!**

Vader uses a dictionary of words and rules to determine the sentiment of a piece of text.

It uses a valence score for each word to determine its positivity or negativity. The valence score ranges from -4 to +4, with -4 being the most negative and +4 being the most positive.

Vader also takes into account the **intensity of the sentiment**, which can be determined by capitalization and punctuation.

For example, all caps or exclamation marks can indicate a stronger sentiment.



# About VADER

```
!pip install nltk  
import nltk
```

```
nltk.download('vader_lexicon')  
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```
analyzer = SentimentIntensityAnalyzer()
```

The SentimentIntensityAnalyzer class provides a method called `polarity_scores()` that takes a piece of text as input and returns a dictionary containing the sentiment scores for the text. The dictionary contains four keys: `neg`, `neu`, `pos`, and `compound`.

`neg`: the negative sentiment score (between 0 and 1)

`neu`: the neutral sentiment score (between 0 and 1)

`pos`: the positive sentiment score (between 0 and 1)

`compound`: the overall sentiment score (between -1 and 1)



# Example

We can pass the text as it is, without any pre-processing

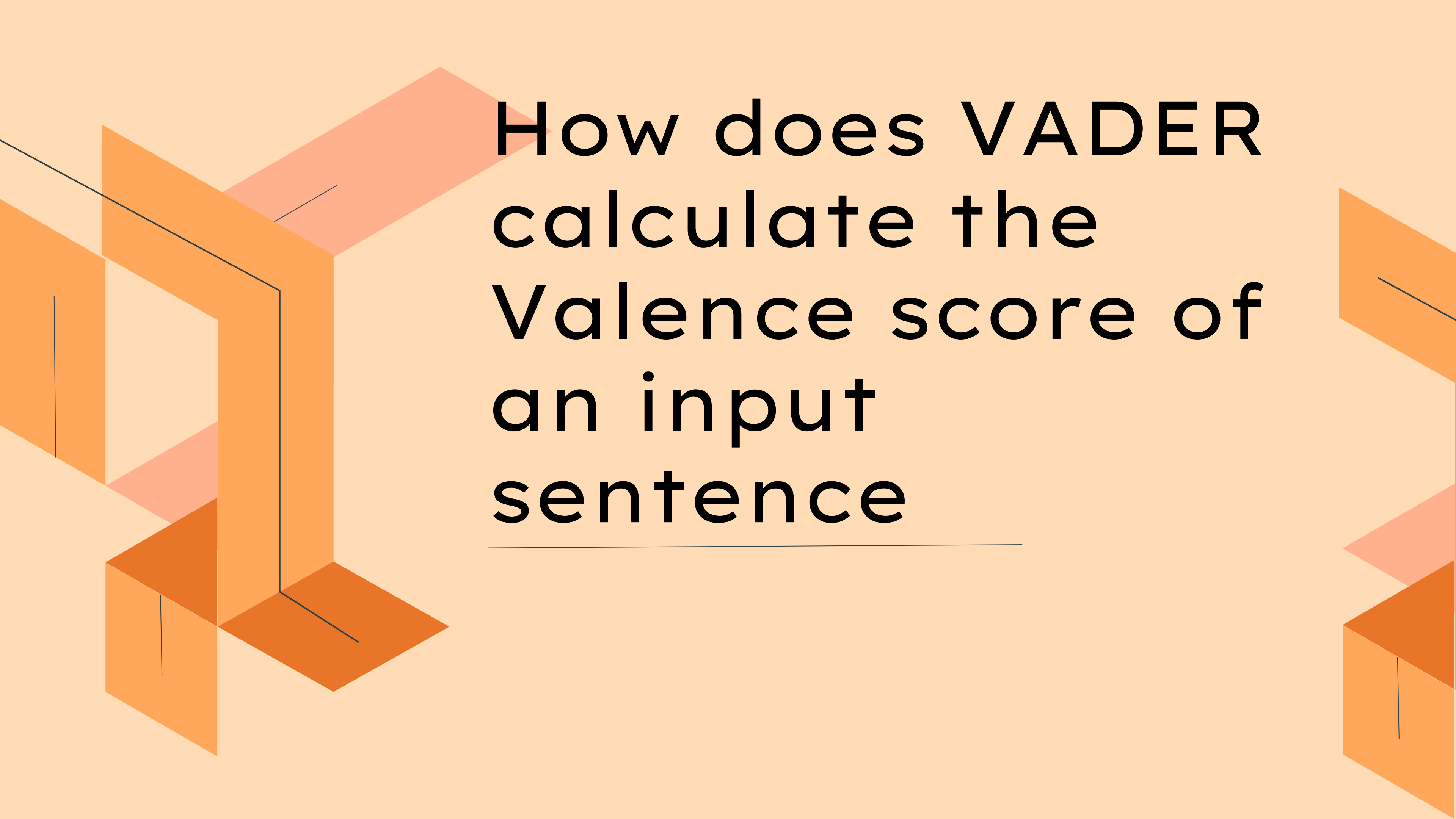
```
text = "I love Python!"
```

```
scores = analyzer.polarity_scores(text)
```

```
print(scores)
```

```
{'neg': 0.0, 'neu': 0.238, 'pos': 0.762, 'compound': 0.6369}
```





How does VADER  
calculate the  
Valence score of  
an input  
sentence

---

# Five Heuristics of VADER

VADER makes use of certain rules to incorporate the impact of each sub-text on the perceived intensity of sentiment in sentence-level text. These rules are called Heuristics. There are 5 of them.

NOTE: These heuristics go beyond what would normally be captured in a typical bag-of-words model. They incorporate word-order sensitive relationships between terms.



1. Punctuation, namely the exclamation point (!), increases the magnitude of the intensity without modifying the semantic orientation.

**For example: “The weather is hot today!!!”  
is more intense than  
“The weather is hot today.”**

2. Capitalization, specifically using ALL-CAPS to emphasize a sentiment-relevant word in the presence of other non-capitalized words, increases the magnitude of the sentiment intensity without affecting the semantic orientation.

**For example: “OMG WHAT.” conveys more intensity than “omg what.”**

3. Degree modifiers (also called intensifiers, booster words, or degree adverbs) impact sentiment intensity by either increasing or decreasing the intensity.

**For example:**

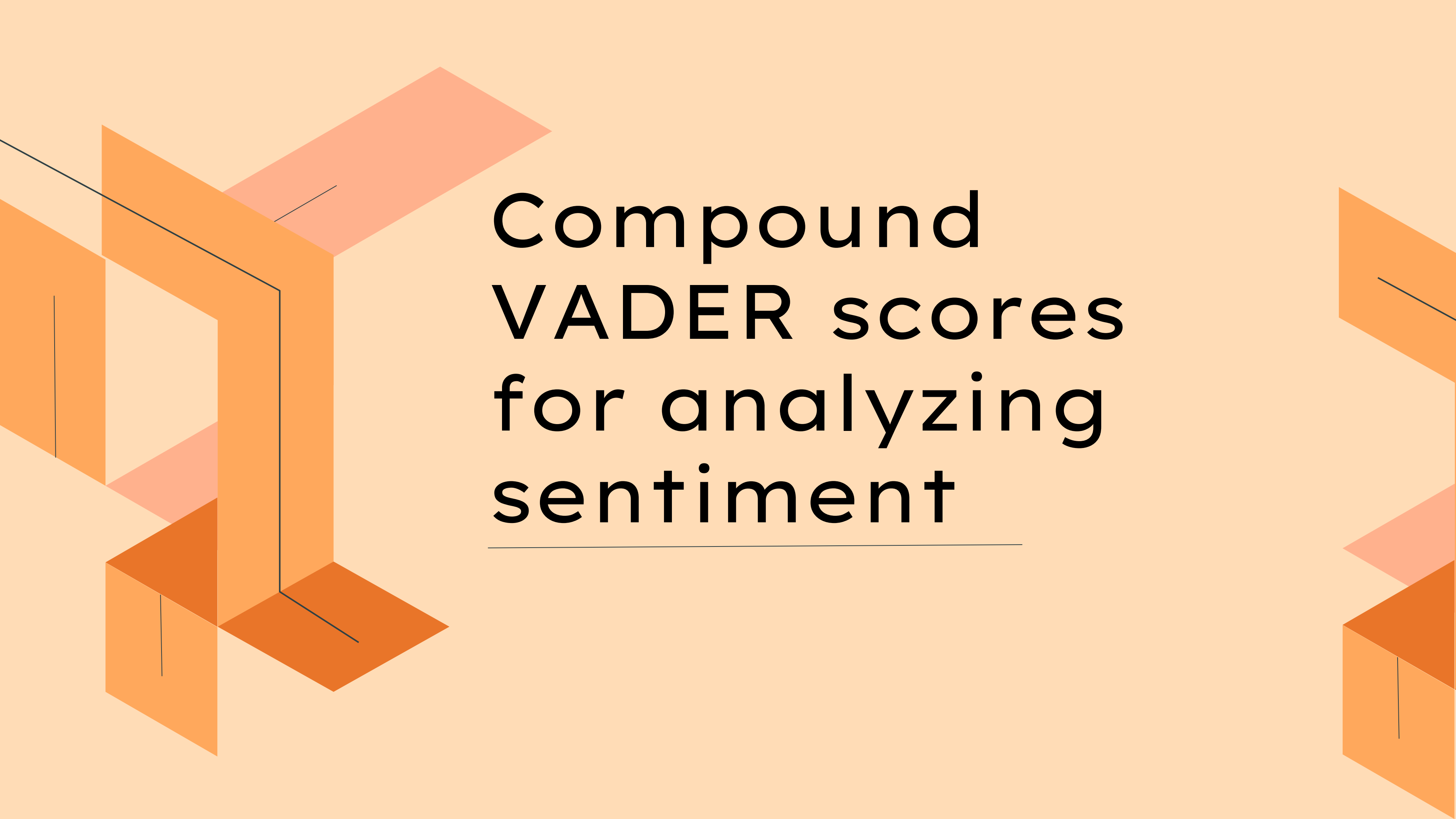
**“The weather is extremely hot.” is more intense than “The weather is hot.”, whereas “The weather is slightly hot.” reduces the intensity.**

4. Polarity shift due to Conjunctions, The contrastive conjunction “but” signals a shift in sentiment polarity, with the sentiment of the text following the conjunction being dominant.

**For example: “The weather is hot, but it is bearable.” has mixed sentiment, with the latter half dictating the overall rating.**

5. Catching Polarity Negation, By examining the contiguous sequence of 3 items preceding a sentiment-laden lexical feature, we catch nearly 90% of cases where negation flips the polarity of the text.

**For example a negated sentence would be  
“The weather isn't really that hot.”.**



# Compound VADER scores for analyzing sentiment

---

# Compound Score

The compound score is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive). This is the most useful metric if you want a single unidimensional measure of sentiment for a given sentence.

As explained in the paper, researchers used below normalization.

$$x = \frac{x}{\sqrt{x^2 + \alpha}}$$

where  $x$  = sum of valence scores of constituent words, and  $\alpha$  = Normalization constant (default value is 15)





# Hands On

Classify sentiment using VADER



# Word Cloud- A pre-cursor to Topic extraction/modeling

## What is Topic Modeling?

- Identifies themes/topics in text data using document-term matrix decomposition.

## Models & Key Concepts:

- *LDA (Latent Dirichlet Allocation)* – Probabilistic model, assumes documents are mixtures of topics. Computationally expensive but good for large datasets.
- *NMF (Non-Negative Matrix Factorization)* – Linear algebraic model, decomposes data into lower-rank matrices. Works well for short texts but lacks semantic understanding.
- *BERTopic* – Uses BERT embeddings, clustering (UMAP + HDBSCAN), and class-based TF-IDF. Captures semantic meaning but assumes single-topic documents.

## Choosing the Right Model:

- LDA – Best for large datasets; interpretable but computationally heavy.
- NMF – Efficient for small datasets but ignores semantic relations.
- BERTopic – Captures meaning but may produce redundant topics.

## Comments 1,264

Top

★ Topics

Newest

### AI-generated topics

Quality and accuracy may vary ⓘ

¡Una receta deliciosa y fácil de hacer!



Genial para una comida rápida y fácil que a todos les encantará



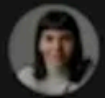
Positive feedback.



These instructions are so easy to follow and such a yummy recipe!



This recipe is great!



This has become one of my go to meals for a delicious weeknight dinner



Great recipe!



Literally my favorite ever! I made this

## ← Topic

¡Una receta deliciosa y fácil de hacer!

Translate to English



@mariarosales · 4hrs ago



Genial para una comida rápida y fácil que a todos les encantará

Translate to English

👍 102



2 replies



@abuea · 7h ago



a toda mi familia le encanta este plato

Translate to English



Add a comment

# Intro to Huggingface and open source pre-trained models

**Hugging Face is a machine learning (ML) and data science platform and community that helps users build, deploy and train machine learning models.**

It provides the **infrastructure to demo, run and deploy** artificial intelligence (AI) in live applications. Users can also browse through models and data sets that other people have uploaded.

Hugging Face is **often called the GitHub of machine learning** because it lets developers share and test their work openly.

Hugging Face is known for its **Transformers Python library**, which simplifies the process of downloading and training ML models. The library gives developers an efficient way to include one of the ML models hosted on Hugging Face in their workflow and create ML pipelines.

The platform is important because of its **open source nature** and deployment tools.





## The AI community building the future

### **Hugging Face – The AI community building the future.**

We're on a journey to advance and democratize artificial intelligence through open source and open science.

 huggingface

<https://huggingface.co>



## HF Learn

[`huggingface.co/learn`](https://huggingface.co/learn)

### **Hugging Face - Learn**

We're on a journey to advance and democratize artificial intelligence through open source and open science.

 huggingface

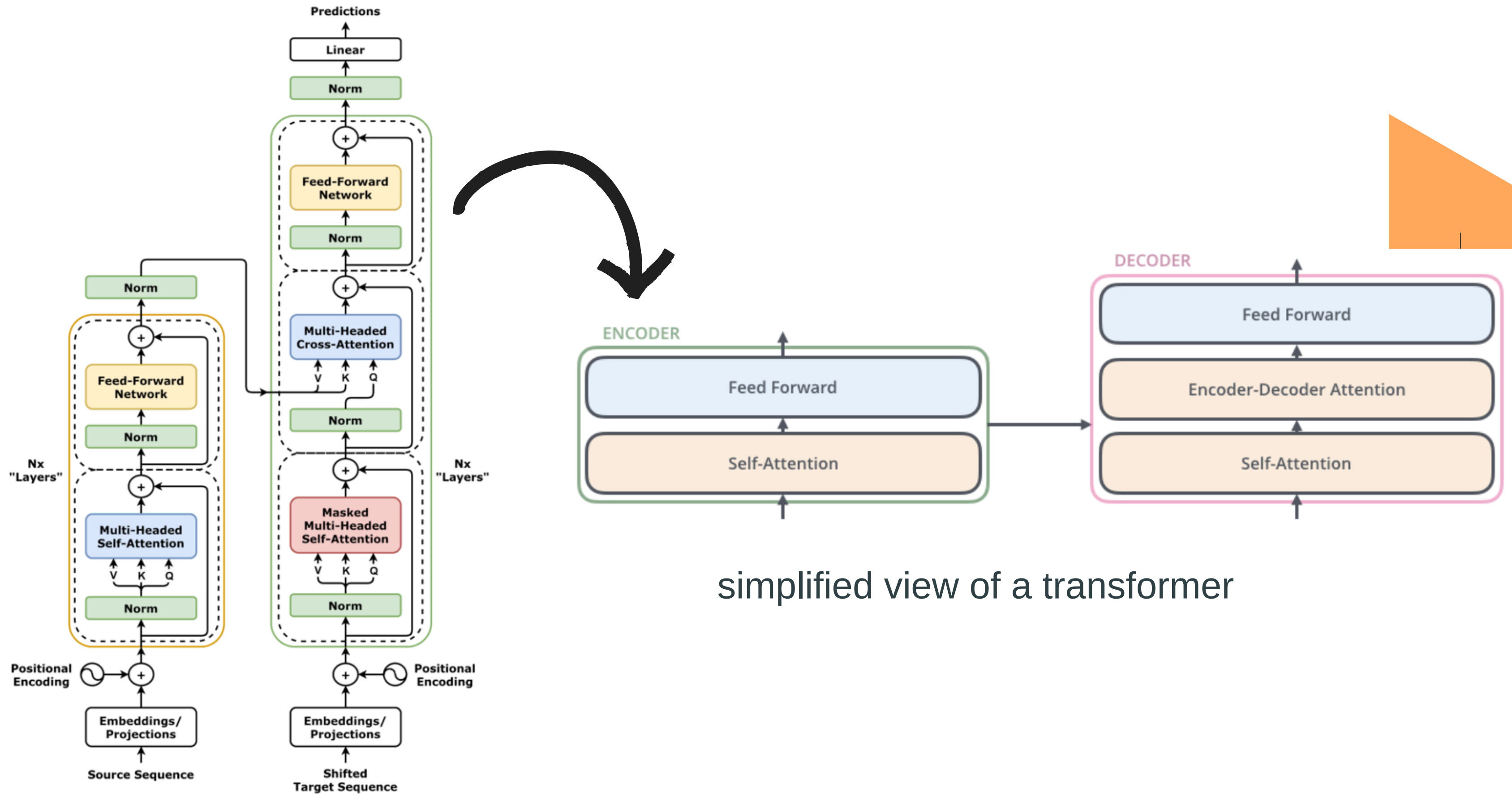
<https://huggingface.co/learn>



# Transformers

---

# What does a transformer look like?



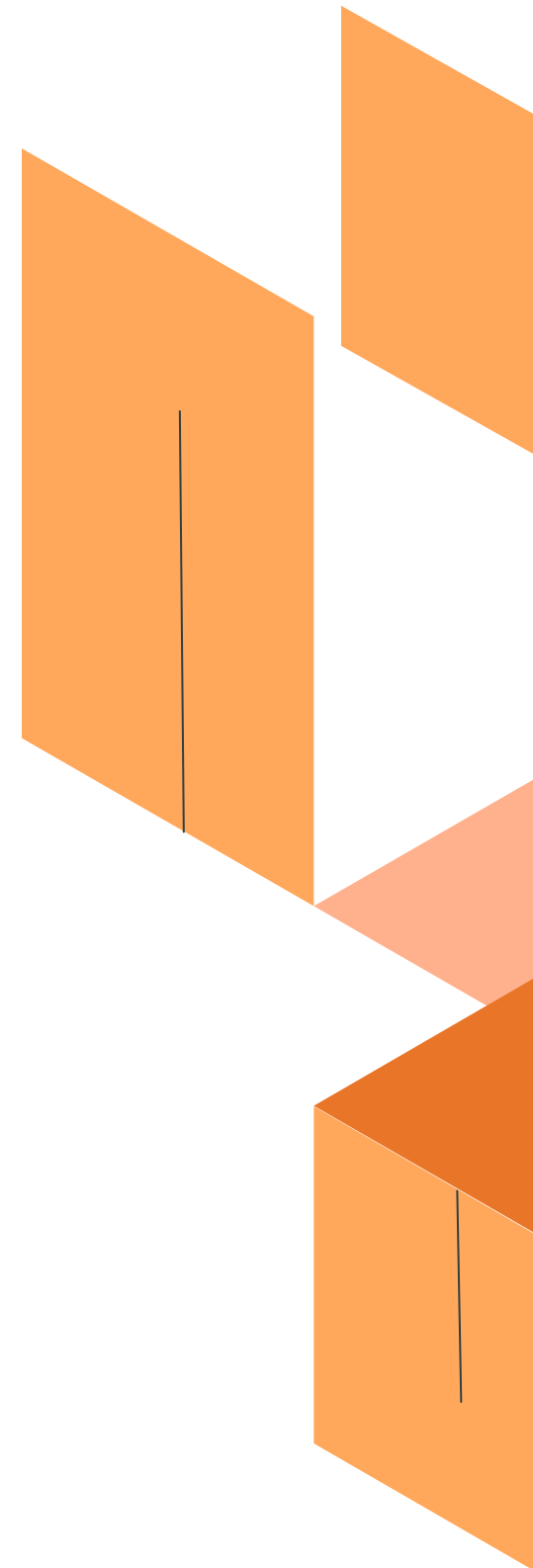


## Main components:

1. Tokeniser: convert sentences to tokens
2. Embedding Layer: convert token to vectors
3. Transforming Layers: encoding and decoding layers (alternating between feedforward and self-attention layers).
4. Unembedding layer

## Why transformers?:

- Parallelization: Unlike RNNs, which process sequentially, Transformers process entire sequences simultaneously.
- Long-range dependencies: Can relate distant words effectively, unlike RNNs that struggle with long-term dependencies.
- Scalability: Forms the foundation of large models like BERT, GPT, and T5.



# BERT

---

Bidirectional Encoder  
Representations from Transformer





# Where do we use Bert?

## Sentiment analysis

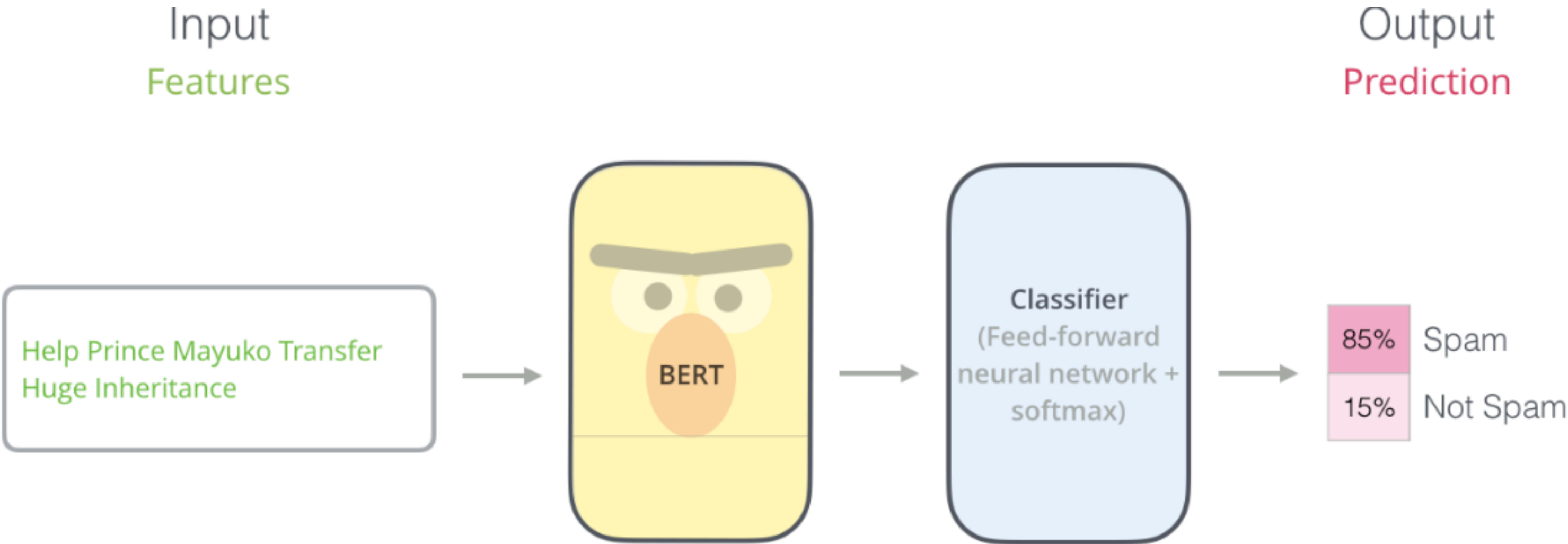
Input: Movie/Product review.  
Output: is the review positive or negative?

## Fact-checking

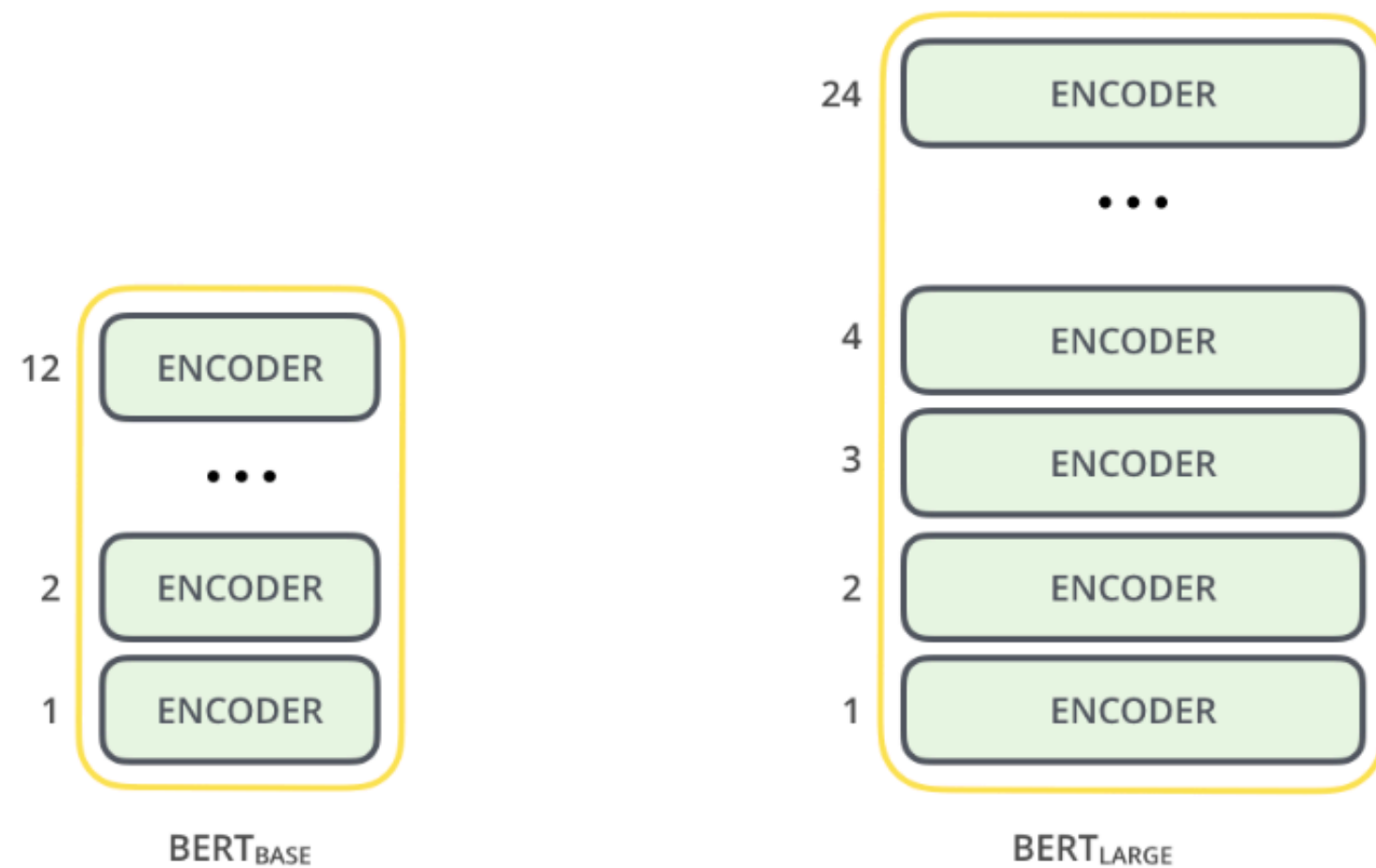
Input: sentence.  
Output: “Claim” or “Not Claim”

## Sentence classification

Input: Any sentence  
Output: Spam or ham



# Bert Architecture:



1. BERT takes a sequence of words as input which keep flowing up the stack. Each layer applies self-attention, and passes its results through a feed-forward network, and then hands it off to the next encoder.
2. Each position outputs a vector of size *hidden\_size* (768 in BERT Base).
3. That vector can now be used as the input for a classifier of our choosing.

# How is Bert trained?

1 - **Semi-supervised** training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

## Semi-supervised Learning Step

**Model:**



**Dataset:**



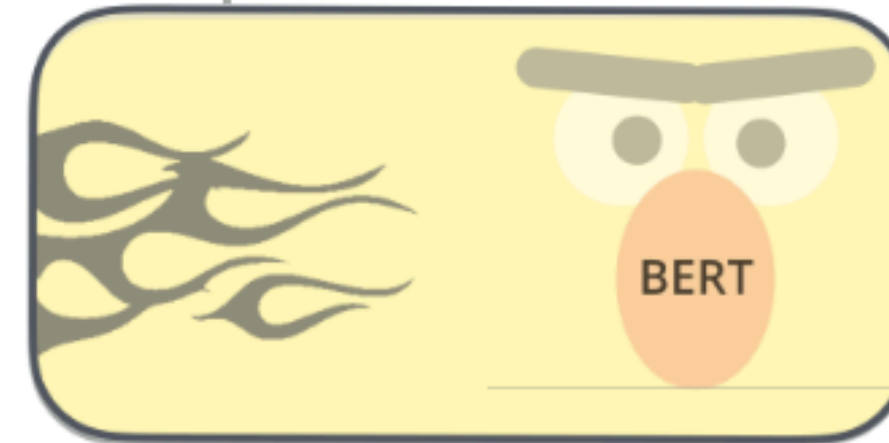
**Objective:**

Predict the masked word  
(language modeling)

2 - **Supervised** training on a specific task with a labeled dataset.

## Supervised Learning Step

**Model:**  
(pre-trained  
in step #1)



**Classifier**

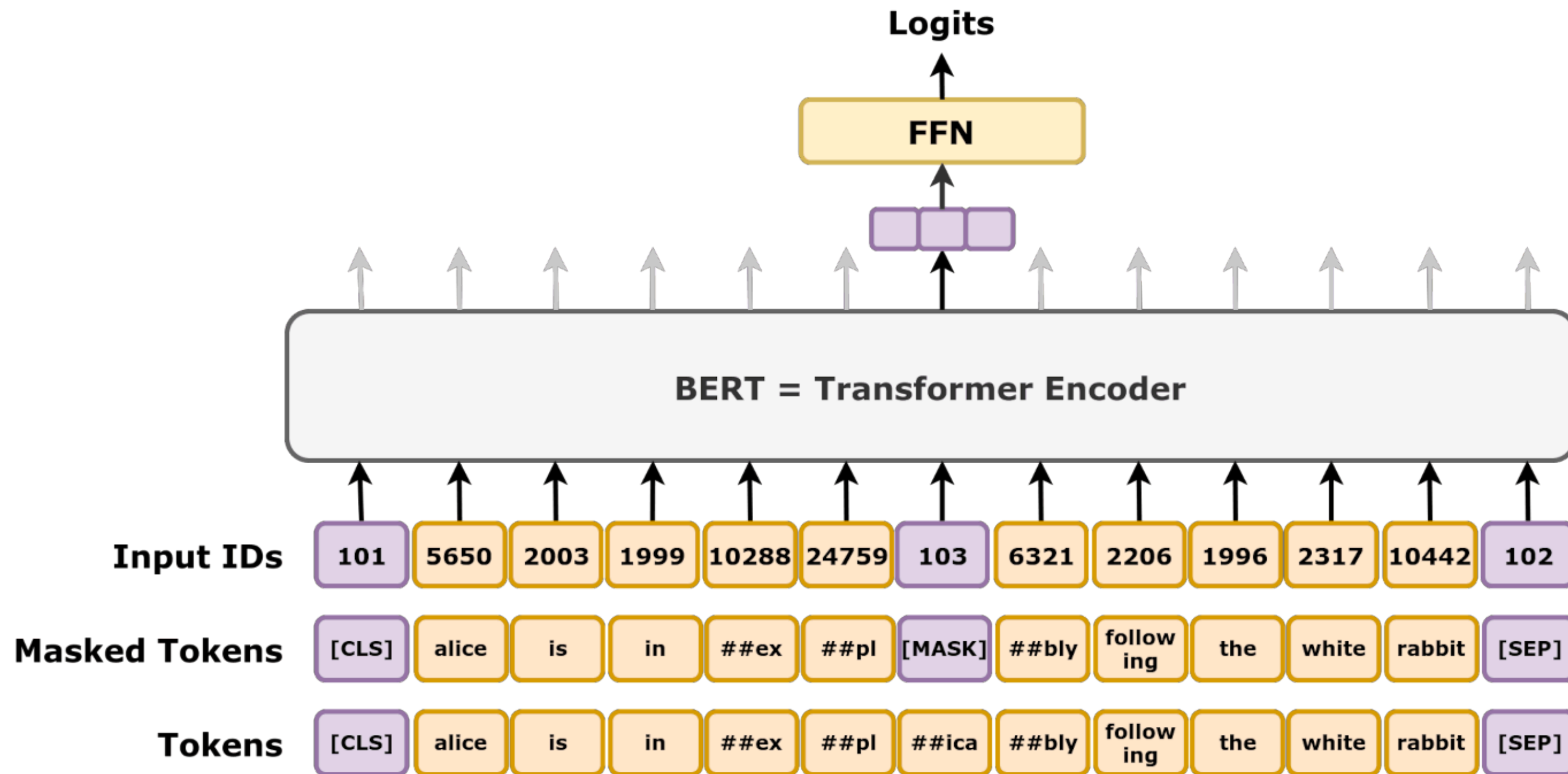
75% Spam  
25% Not Spam

**Dataset:**

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached...	Not Spam

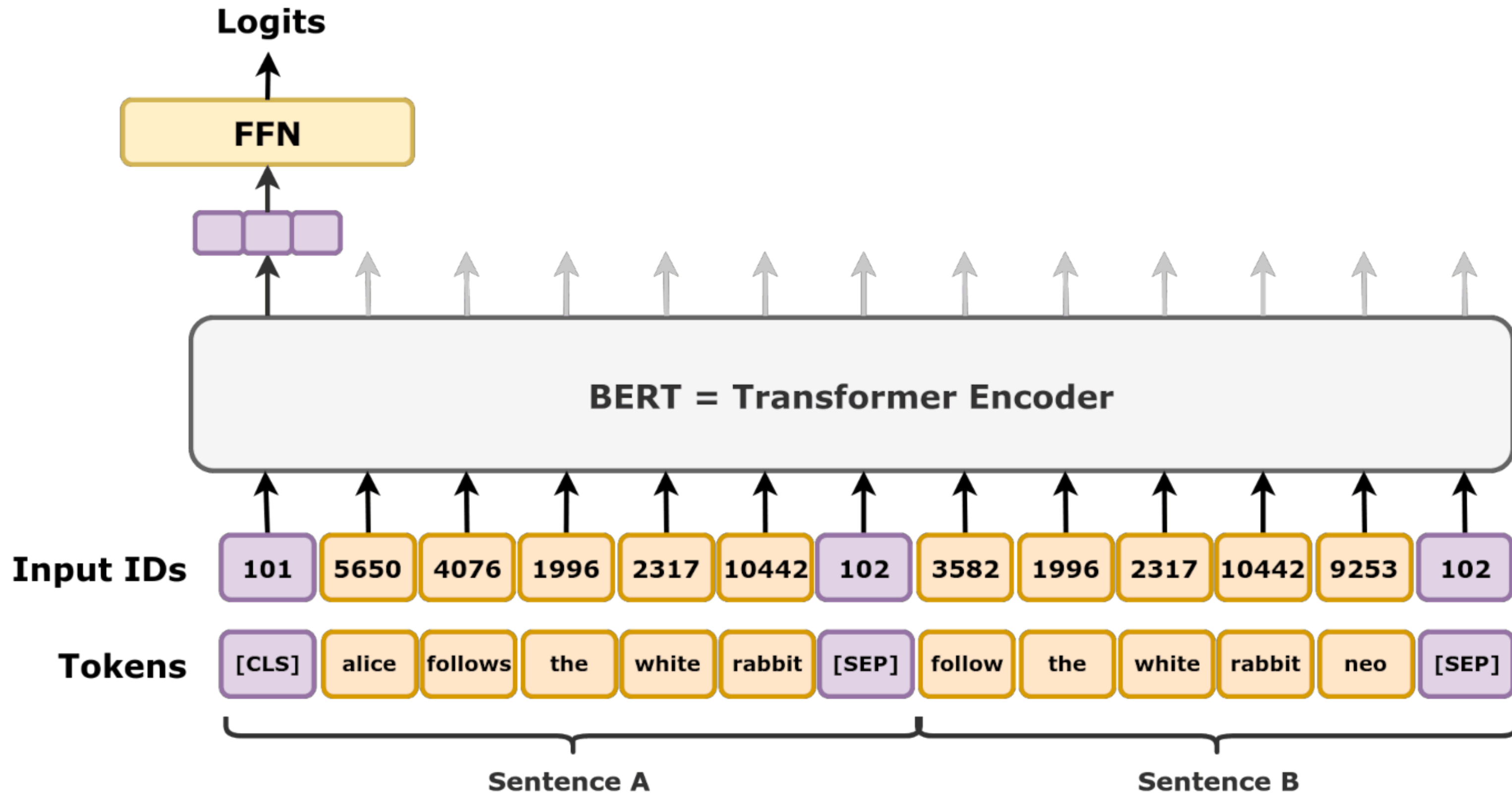
# How is Bert trained?: The pre-training phase

## Step 1: Masked Language Modelling



# How is Bert trained?: The pre-training phase

## Step 2: Next Sentence Prediction





# Hands On

## Let's use Bert!

The background features abstract geometric shapes in shades of orange and peach. On the left, there is a large, complex shape resembling a stylized 'L' or a corner of a cube. On the right, there are several smaller, angular shapes, some of which look like they might be parts of boxes or architectural elements. The overall aesthetic is clean and modern.

# PROJECTS / OPEN DISCUSSION

Let's see how much you learnt!  
Quiz Time!





**Your feedback matters!**  
Let us know your thoughts here



# Synapse Hackathon Form

