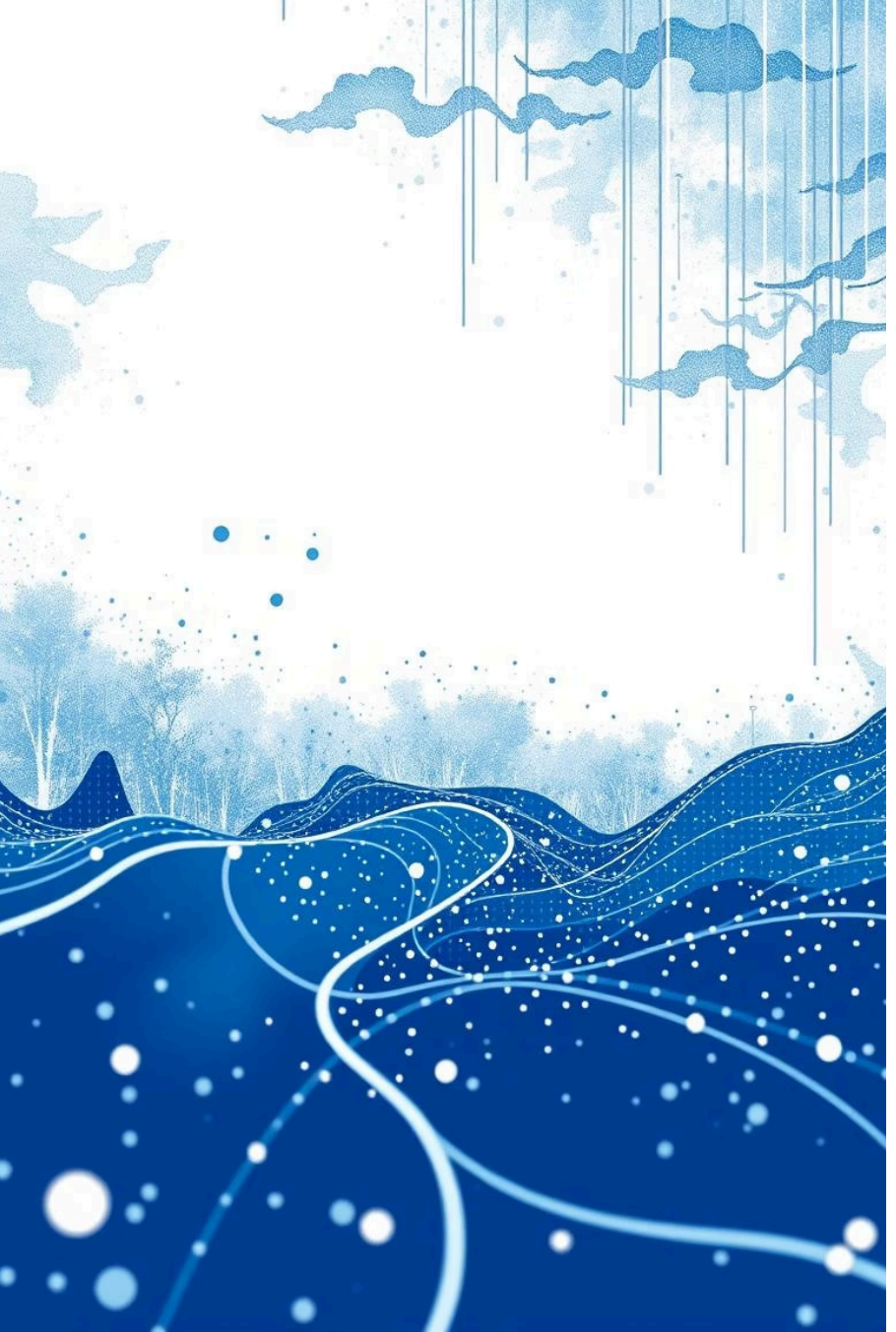
Understanding Text Preprocessing: Cleaning Synthetic Text (Real-World)

From messy raw data to robust NLP applications



Introduction

- User-generated text is powerful but messy.
- Full of noise: links, emojis, hashtags, misspellings.
- Preprocessing ensures reliable NLP performance.

Agenda

01

The Need for Clean Text

02

Data Loading & Overview

03

Raw Data Issues

04

Text Cleaning Operations

05

Normalization

06

Dataset Consolidation

07

Sentiment Labeling

08

TF-IDF Vectorization

09

Train/Test Split

10

Logistic Regression Model

11

Predictions & Evaluation

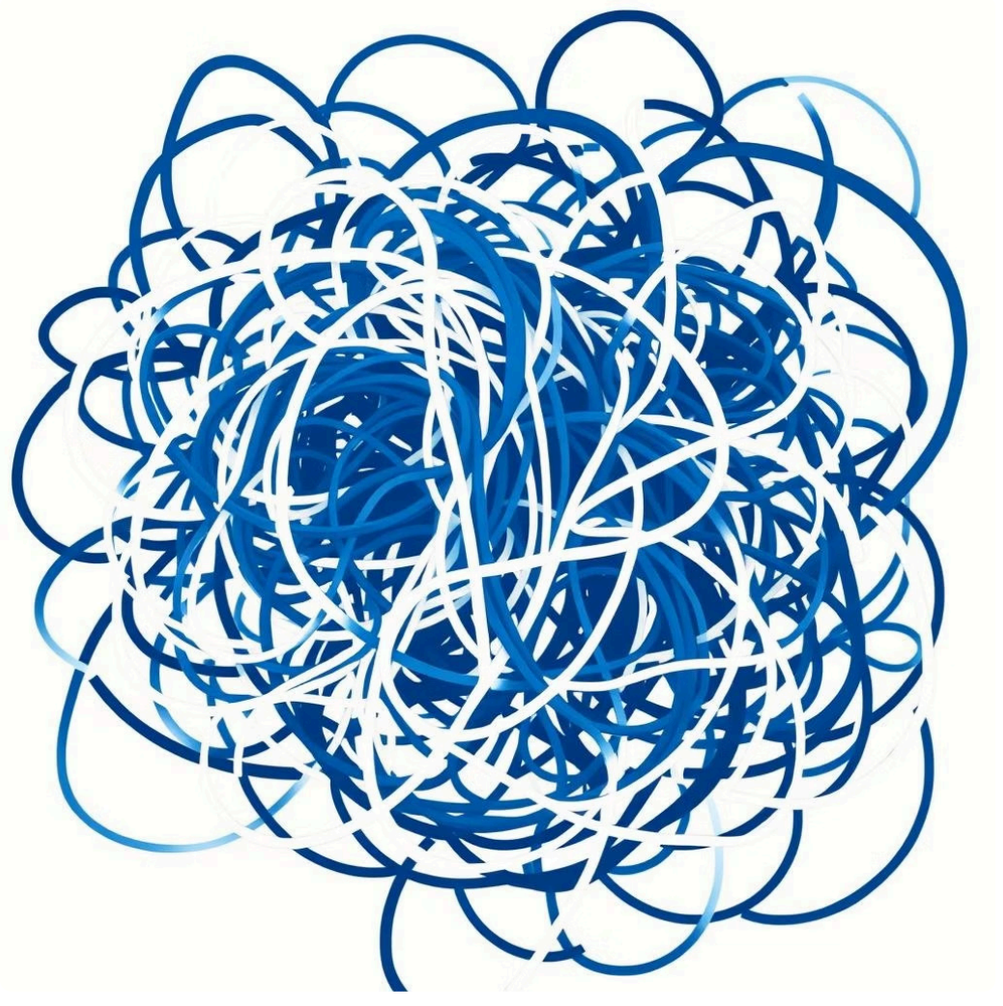
12

Key Takeaways

The Need for Clean Text

UGC = valuable but unstructured.

Without cleaning → inaccurate models.



Example:

> "Download now >>> http://spam.com"
>
> "Café München is beautïful but costly $$$"

These examples highlight the diverse noise present in real-world text data, which can significantly impact NLP model performance if not addressed.

Data Loading & Overview

```
import pandas as pd, zipfile, io, requests
url = "https://github.com/.../archive (4).zip"
response = requests.get(url)
z = zipfile.ZipFile(io.BytesIO(response.content))
df = pd.read_csv(z.open("synthetic_text_cleaning_realworld.csv"))
print(df.shape)
df.head()
```

✅ Dataset: 2000 rows × 30 columns

✅ Fields: tweet_text, comment_text, review_text, misc_text

	id	user_id	username	email	phone_number	tweet_text	post_text	comment_text
0	1	Download now >>> free link http://spam.com	Data cleaning is very IMPORTANT.....	Check this out >>> https://openai.com	U r the best!! btw I'll c u 2morrow 😉	Data cleaning is very IMPORTANT.....	Email me at test_user123@gmail.com asap!!!	U r the best!! btw I'll c u 2morrow 😉
1	2	Stopwords should be removed from this text in ...	Check this out >>> https://openai.com	Email me at test_user123@gmail.com asap!!!	Working on NLP??? It's fun, isn't it???	Café München is beautïful but costly $$$	Heeeyyy broooo wassuppp??? 😂😂😂	Heeeyyy broooo wassuppp??? 😂😂😂
2	3	Heeeyyy broooo wassuppp??? 😂😂😂	Stopwords should be removed from this text in ...	Email me at test_user123@gmail.com asap!!!	Café München is beautïful but costly $$$	Breaking news!!!! #AI #NLP @research	Email me at test_user123@gmail.com asap!!!	Data cleaning is very IMPORTANT.....
3	4	Café München is beautïful but costly $$$	Data cleaning is very IMPORTANT.....	U r the best!! btw I'll c u 2morrow 😉	Stopwords should be removed from this text in ...	U r the best!! btw I'll c u 2morrow 😉	Data cleaning is very IMPORTANT.....	Data cleaning is very IMPORTANT.....
4	5	This TEXT has MULTIPLE spaces and --- dash...	This TEXT has MULTIPLE spaces and --- dash...	Check this out >>> https://openai.com	Working on NLP??? It's fun, isn't it???	123456789 Numbers should be removed!!!	Hello!!! This is a SAMPLE text 😊.	U r the best!! btw I'll c u 2morrow 😉

Raw Data Issues

URLs, emails, phone numbers

Unwanted web addresses and contact information.

Emojis, hashtags, mentions

Social media specific elements that add noise.

Non-ASCII characters

Special characters that can cause encoding issues.

Inconsistent casing & whitespace

Variations in text formatting that need standardisation.

Cleaning Function

```
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'http\S+|www\S+', '', text) # URLs
    text = re.sub(r'\S+@\S+', '', text) # Emails
    text = re.sub(r'\d+', '', text) # Numbers
    text = text.translate(str.maketrans('', '', string.punctuation))
    return text.strip()
```

This simplified Python function demonstrates the core logic for removing common noise elements from text data, ensuring a cleaner input for NLP tasks.

id	user_id	username	email	phone_number	tweet_text	post_text	comment_text	review_text	bio_text	...	product_description
1	download now free link	data cleaning is very important	check this out	u r the best btw ill c u morrow	data cleaning is very important	email me at asap	u r the best btw ill c u morrow	caf mnchen is beautful but costly	download now free link	...	heeeyyy broooo wassuppp
2	stopwords should be removed from this text in ...	check this out	email me at asap	working on nlp its fun isnt it	caf mnchen is beautful but costly	heeeyyy broooo wassuppp	heeeyyy broooo wassuppp	numbers should be removed	caf mnchen is beautful but costly	...	caf mnchen is beautful but costly
3	heeeyyy broooo wassuppp	stopwords should be removed from this text in ...	email me at asap	caf mnchen is beautful but costly	breaking news	email me at asap	data cleaning is very important	check this out	hello this is a sample text	...	breaking news
4	caf mnchen is beautful but costly	data cleaning is very important	u r the best btw ill c u morrow	stopwords should be removed from this text in ...	u r the best btw ill c u morrow	data cleaning is very important	data cleaning is very important	this text has multiple spaces and dashes	email me at asap	...	breaking news
5	this text has multiple spaces and dashes	this text has multiple spaces and dashes	check this out	working on nlp its fun isnt it	numbers should be removed	hello this is a sample text	u r the best btw ill c u morrow	u r the best btw ill c u morrow	heeeyyy broooo wassuppp	...	email me at asap

Before & After Cleaning

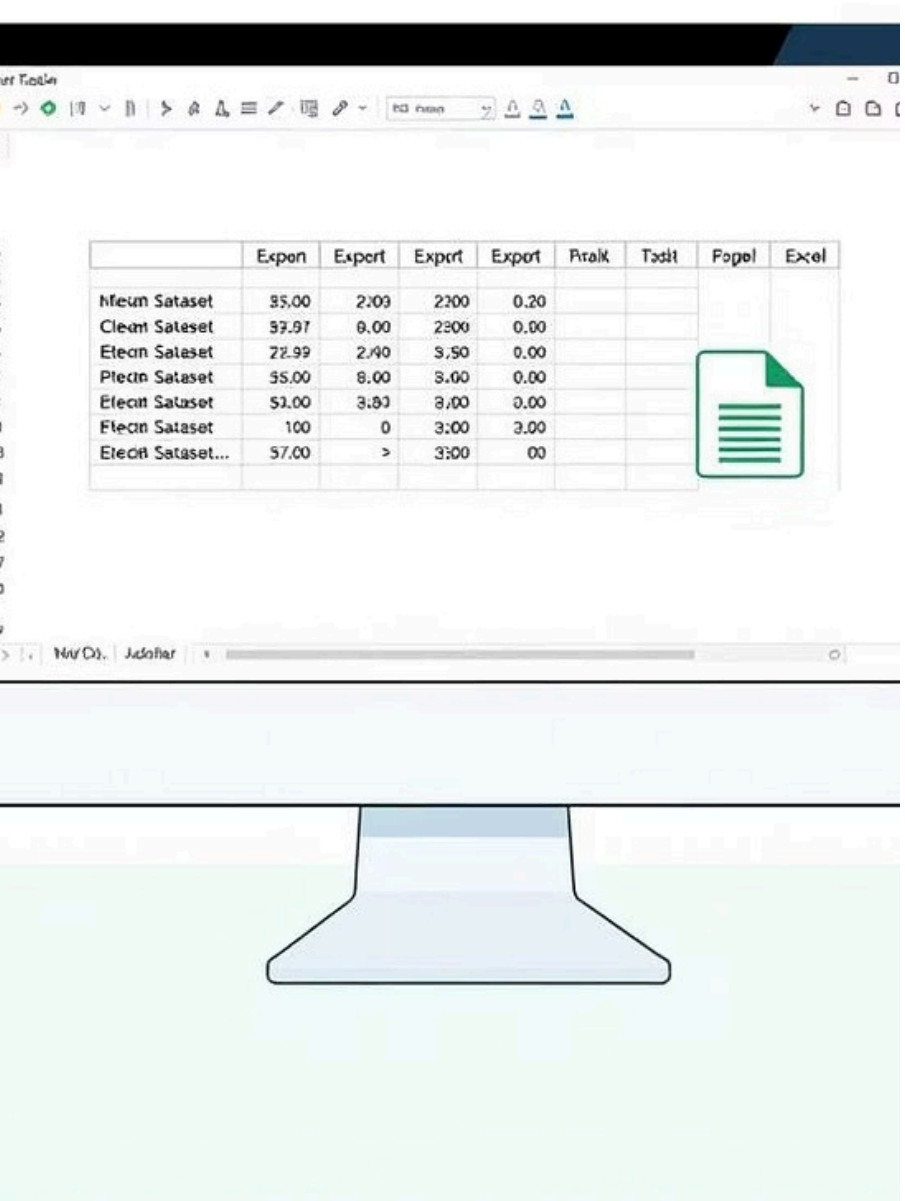
Before:

> "Café München is beautïful but costly $$$"

After:

> "caf mnchen is beautful but costly"

📌 Shows removal of noise → uniform clean text.



Exporting Clean Dataset

```
cleaned_filename = "cleaned_dataset.xlsx"
df.to_excel(cleaned_filename, index=False)
```

✅ Clean dataset saved for modeling.

Creating Unified Text Column

```
drop_cols = ['id','user_id','username','email','phone_number']
df = df.drop(columns=drop_cols)
df_melted = df.melt(value_vars=text_columns, var_name='source',
value_name='text')
df_melted = df_melted.dropna(subset=['text'])
```

✅ All text sources merged → single text column

	source	text	sentiment
0	tweet_text	data cleaning is very important	positive
1	tweet_text	caf mnchen is beautful but costly	neutral
2	tweet_text	breaking news	neutral
3	tweet_text	u r the best btw ill c u morrow	positive
4	tweet_text	numbers should be removed	neutral
...	...	...	...
49995	misc_text	download now free link	positive
49996	misc_text	python is great visit now	positive
49997	misc_text	download now free link	positive
49998	misc_text	caf mnchen is beautful but costly	neutral
49999	misc_text	working on nlp its fun isnt it	positive

Sentiment Assignment

```
from textblob import TextBlob
def get_sentiment(text):
    polarity = TextBlob(str(text)).sentiment.polarity
    if polarity > 0.05: return "positive"
    elif polarity < -0.05: return "negative"
    else: return "neutral"
df_melted['sentiment'] = df_melted['text'].apply(get_sentiment)
```

This Python snippet demonstrates how sentiment is assigned to text data using TextBlob, categorising each entry as positive, negative, or neutral based on its polarity score.

Example:

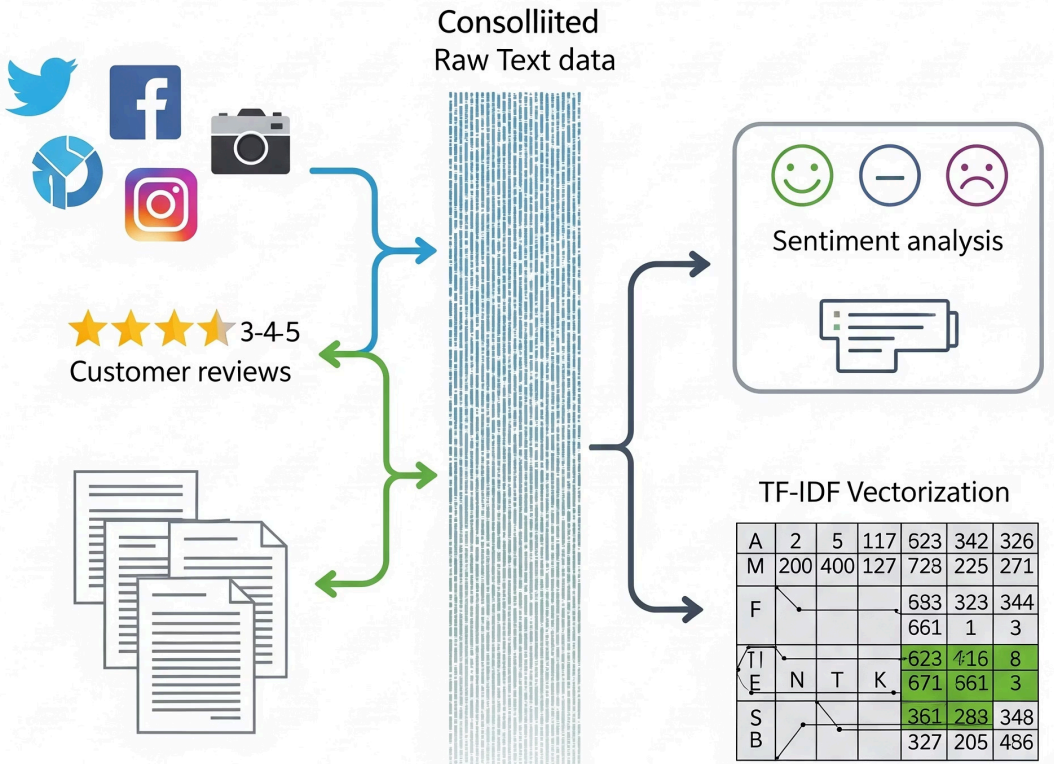
> “python is great” → Positive
>
> “caf mnchen costly” → Neutral

TF-IDF Vectorization

```
from sklearn.feature_extraction.text import TfidfVectorizer
X = df_melted['text']
y = df_melted['sentiment']
tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
X_tfidf = tfidf.fit_transform(X)
```

TF-IDF converts text into numerical features by quantifying word importance. It helps capture the relevance of terms in a document relative to a corpus. Here, stop words are removed, and the vocabulary is limited to the top 5000 features for efficient model training.

✅ Converts text → numerical features.


Consolliited
Raw Text data
3-4-5
Customer reviews
Sentiment analysis
TF-IDF Vectorization
A 2 5 117 623 342 326
M 200 400 127 728 225 271
F 683 323 344
661 1 3
623 416 8
671 661 3
361 283 348
327 205 486

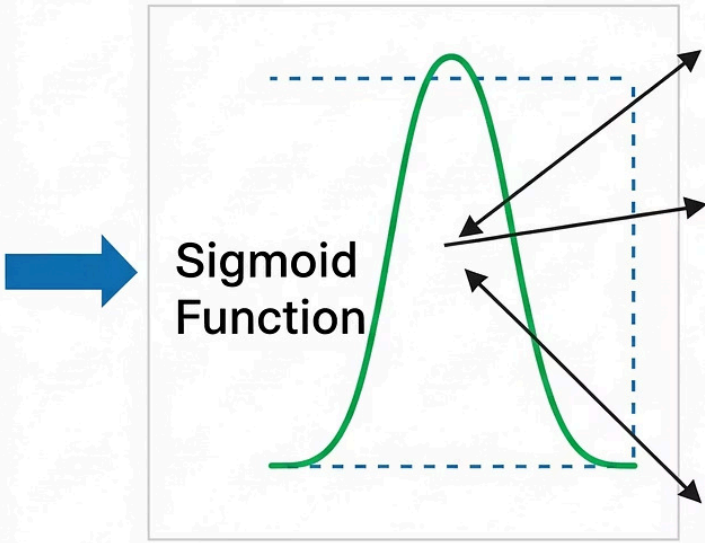
Train/Test Split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X_tfidf, y, test_size=0.2, random_state=42, stratify=y
)
```

This crucial step divides the dataset into training (80%) and testing (20%) sets. `random_state` ensures reproducibility, while `stratify=y` maintains the proportion of sentiment categories in both sets, preventing biased evaluations and ensuring robust model performance on unseen data.

eatures

4	1	08
1		13
11		10
10		10
17		18
25		17
16		04
10		12
25		05
25		00
24		29
24		10
15		49



Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

✅ The logistic regression model is now trained to classify sentiment, learning from the TF-IDF vectorized text data and assigned sentiment labels. The `max_iter` parameter is set to a sufficiently high value to ensure convergence.

Predictions & Next Steps

```
y_pred = model.predict(X_test)
```

With predictions now generated, the immediate next step involves a comprehensive evaluation of our model's performance. We'll leverage standard metrics such as the classification report and confusion matrix to assess accuracy, precision, recall, and F1-score, providing a clear picture of its effectiveness.