AI VIET NAM
@aivietnam.edu.vn

Module 03 – Project

Sentiment Analysis
for IMDB Review Dataset

Nguyen Quoc Thai

AI VIET NAM
@aivietnam.edu.vn

Objectives

Text Classification

- Introduction
- Token-level Text Classification
- Document-level Text Classification
- Sentiment Analysis
- IMDB Dataset

Text Preprocessing

- Duplicate Handling
- Text Cleaning
- EDA
- Tokenization

Text Representation

- Numeric Representation
- One-hot Encoding
- Bag-of-Words (BoW)
- TF-IDF

Classification

- Decision Tree Classifier
- Random Forest Classifier
- Evaluation
- Inference

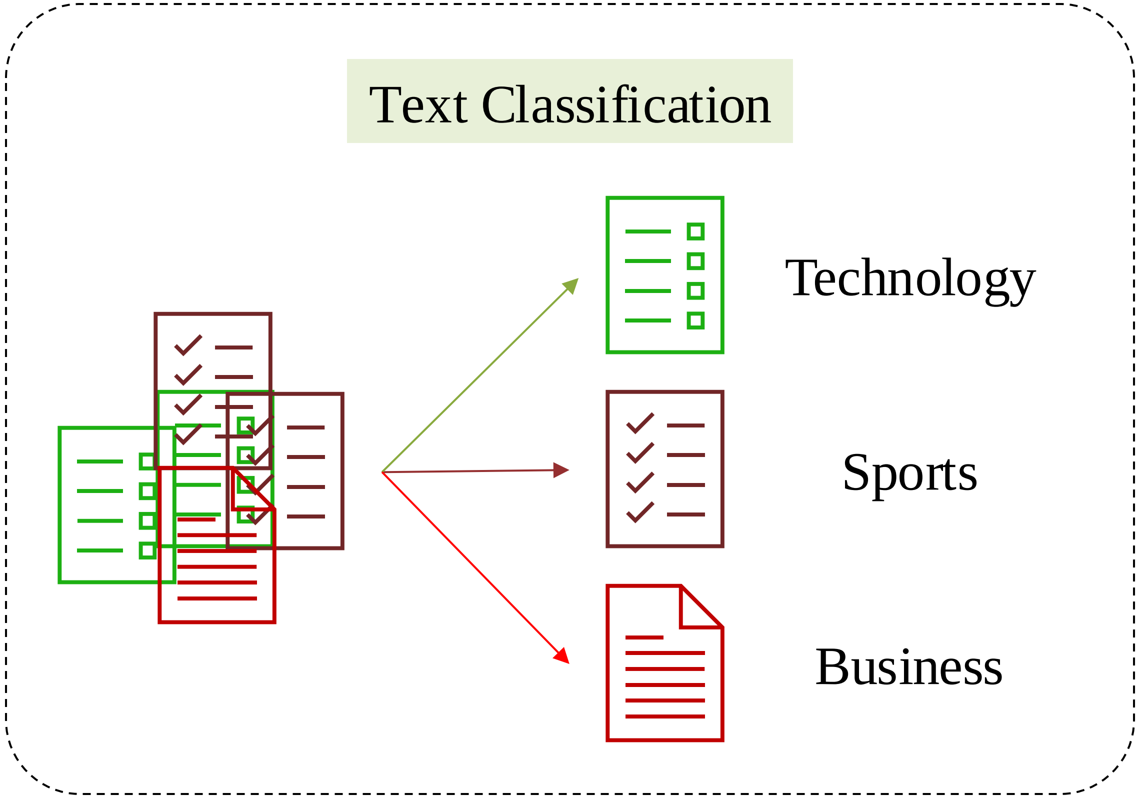
Outline

SECTION 1
Text Classification

SECTION 2
Text Preprocessing

SECTION 3
Text Representation

SECTION 4
Classification



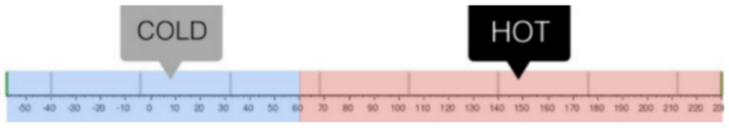
Text Classification

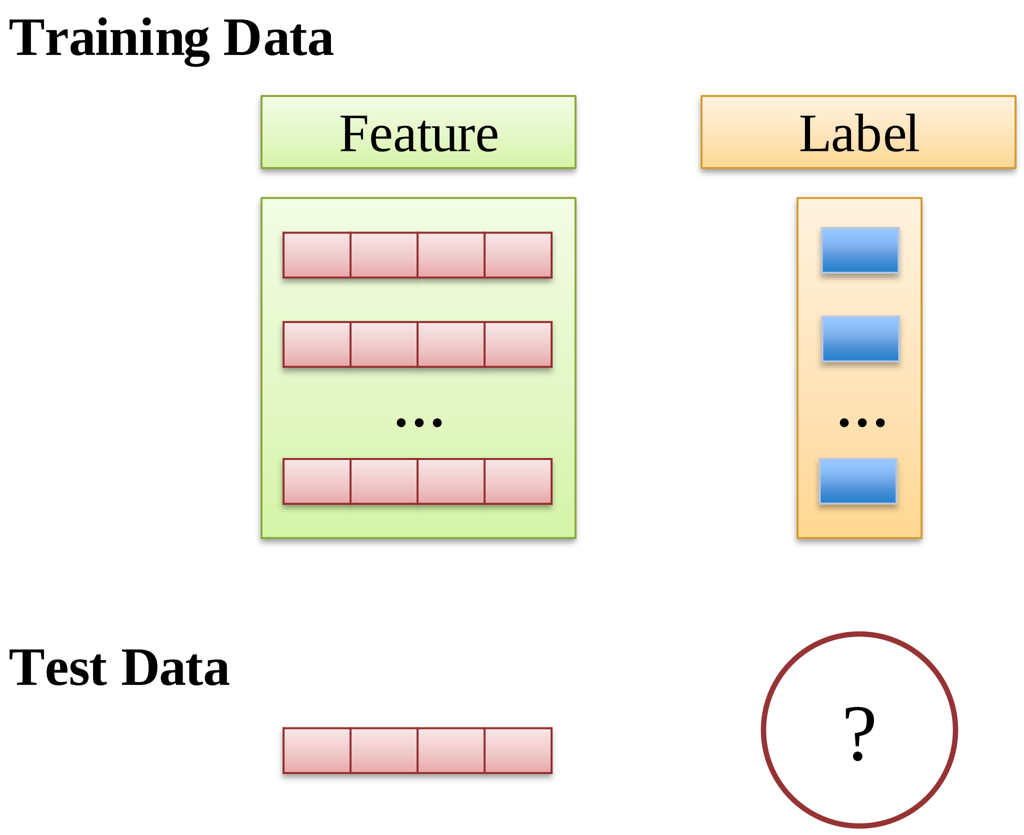
Classification Problem

Classification

- Classify input variables to identify discrete output variables (labels, categories)

Will it be hot or cold tomorrow?





AI VIET NAM
@aivietnam.edu.vn

Text Classification

Classification Problem

- **Input**
  - A fixed set of classes **C** = {c_1, c_2, …, c_N}
  - A training set of **M** hand-labeled documents: (d_1, c_1),…, (d_M, c_N)
  - A document **d**
- **Output**
  - A learned classifier **d** => **c (C)**

Petal_Length	Petal_Width	Label
1	0.2	0
1.3	0.6	0
0.9	0.7	0
1.7	0.5	1
1.8	0.9	1
1.2	1.3	1

Petal_Length	Petal_Width	Label
1.2	0.2	?

Text Classification

Classification Problem

Petal_Length	Petal_Width	Label
1	0.2	0
1.3	0.6	0
0.9	0.7	0
1.7	0.5	1
1.8	0.9	1
1.2	1.3	1

Petal_Width	Label
A wonderful little production. The filming technique is very unassuming- very ole-time-B…	0
A rating of “1” does not begin to express how dull, depressing and relentlessly bad this movie is.	0

Text Classification

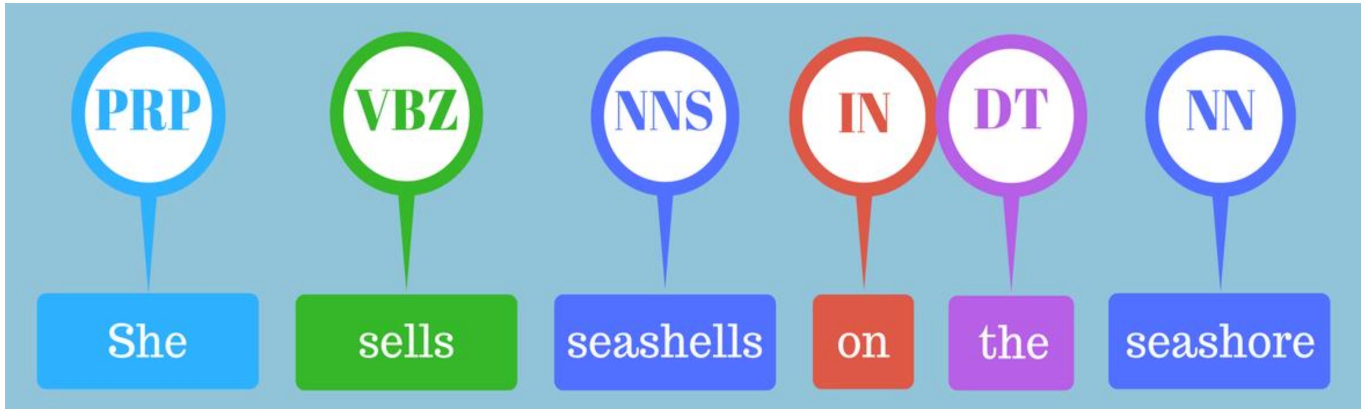
Token-level Tokenization

❖ Sequence Labeling: Word Segmentation, Part Of Speech Tagging (POS), Named Entity Recognition (NER)

Name Entity Recognition (NER)

STATE OF PROVINCE | TIME

I have a flight to New York at 5 pm



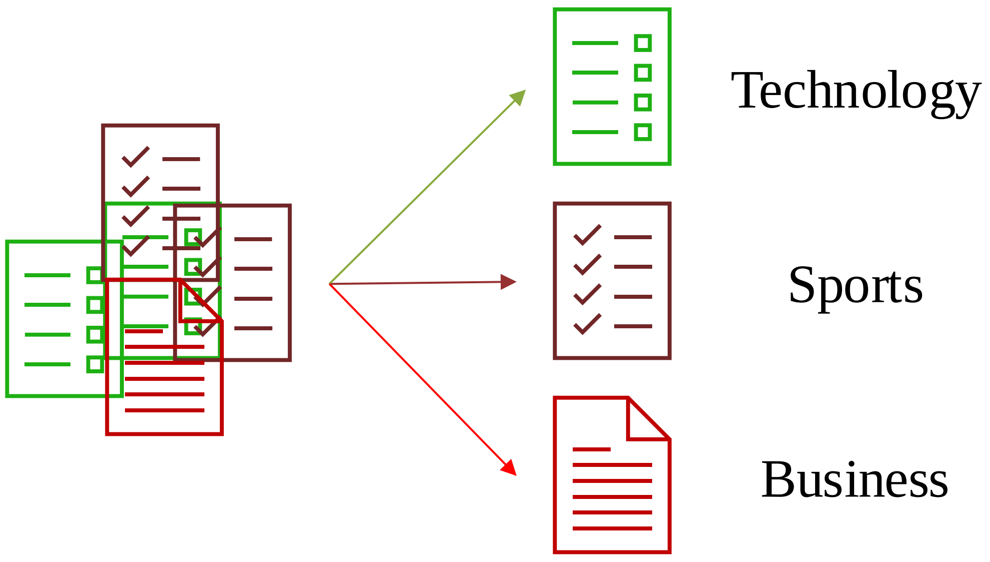
Part Of Speech Tagging (POS)

Text Classification

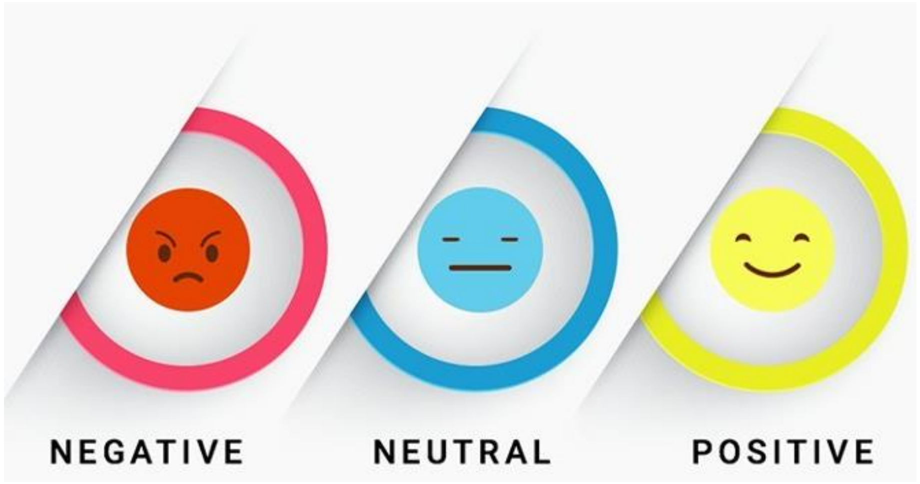
Document-level Tokenization

❖ Sentiment Analysis

Text Classification



Sentiment Analysis



Text Classification

IMDB Review Dataset

Text Cleaning

A wonderful little production.   The filming technique is very unassuming- very ole-time-B…
A rating of "1" does not begin to express how dull, depressing and relentlessly bad this movie is.

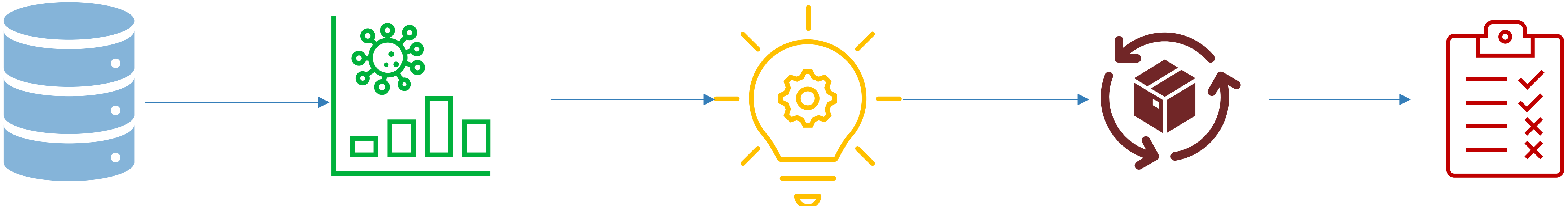
Text Representation

$$\boldsymbol{x} = \begin{bmatrix} 1 & 1.4 & 0.2 \\ 1 & 1.5 & 0.2 \\ 1 & 3.0 & 1.1 \\ 1 & 4.1 & 1.3 \end{bmatrix}$$

Petal_Width	Label
A wonderful little production. The filming technique is very unassuming- very ole-time-B…	0
A rating of "1" does not begin to express how dull, depressing and relentlessly bad this movie is.	0

Text Classification

IMDB Review Dataset



Dataset

Preprocessing

Exploratory Data Analysis

Representation

Modeling

Evaluation

Inference

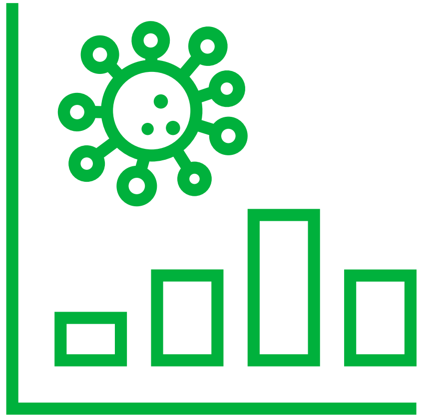
Outline

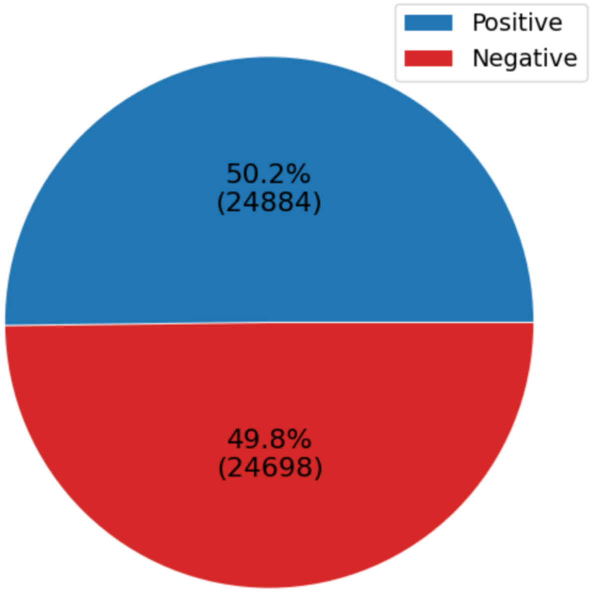
SECTION 1
Text Classification

SECTION 2
Text Preprocessing

SECTION 3
Text Representation

SECTION 4
Classification





Text Preprocessing

Text Preprocessing

- ❑ Removal of URLs and HTML tags
- ❑ Text Standardizing
- ❑ Lowercasing
- ❑ Number and Punctuation Handling

- ❑ Removal Stop Words
- ❑ Removal Rare Words
- ❑ Handle Emoji and Emoticons
- ❑ Spelling Correction

- ❑ Tokenization
  - Sentence
  - Word
  - Character
  - Subwords
- ❑ Stemming
- ❑ Lemmatization

Text Preprocessing

Removal URLs, HTML Tags

- Extract text based on the structure of an HTML document
- URLs: image links, reference links,…
- HTML tags: <p>..</p>, <div>…</div>,…

@82476 🤔 <p> We'd like to help Sam, which number is caling you? </p> Please DM us more info so we can advise further. https://t.co/5pyLDJBC6r

@82476 🤔 We'd like to help Sam, which number is caling you? Please DM us more info so we can advise further.

Text Preprocessing

Text Standardizing

- Lowercasing: Use lower() function in Python
- Using short words and abbreviations to represent the same meaning
- Contrastions: I’m, isn’t, can’t,…

lowercasing

Lowercasing → Convert → lowercasing

DM → Convert → Direct message

I’m / can’t → Convert → I am / can not

```
@82476 🤔 We'd
like to help Sam,
which number is
caling you?
Please DM us more
info so we can
advise further.
```

```
@82476 🤔 we
would like to help
sam, which number
is caling you?
please direct
message us more
information so we
can advise
further.
```

Text Preprocessing

Number and Punctuation Handling

- Removal: Text Classification
- As token: Machine Translation, POS Tagging, Named Entity Recognition

	Removal	As Token
Sam.	Sam	Sam .
You?	You	You ?
Further.	Further	Further .

```
@82476 🤔  We would like to help Sam,  which number is caling you? Please direct message us more information so we can advise further.
```

```
🤔   We would like to help Sam which number is caling you  Please direct message us more information so we can advise further
```

```
@ 82476 🤔   We would like to help Sam ,  which number is caling you ?   Please direct message us more information so we can advise further .
```

Text Preprocessing

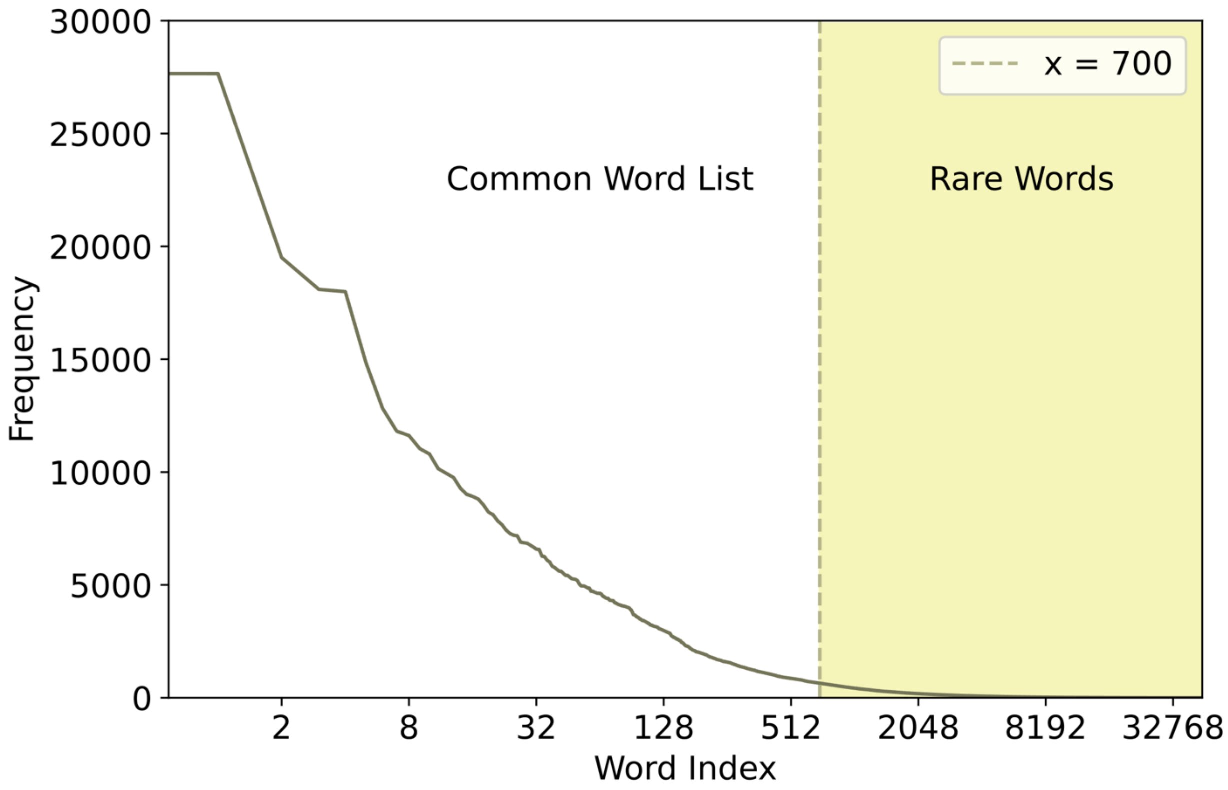
Stop / Rare Words Handling

- Focus on the important keywords
- Stop words: common words - no meaning or less meaning compared to keywords

  English: a, an, that, for,…

  Vietnamese: à, ừ, vậy, thế,…

- Rare words: words that appear only a few times in corpus



Text Preprocessing

Emoji and Emoticons Handling

- Emojis: ☮ 🙂 ❤…
- Emoticons: :-) :-(:-))) :-)
- Some tasks: convert emojis and emoticons to word.
- Example: :-) => happy, :-(=> sad,…

```
@82476 🤔 We would like to help Sam, which number is caling you? Please direct message us more information so we can advise further.
```

```
@82476 thinking_face .We would like to help Sam, which number is caling you? Please direct message us more information so we can advise further.
```

AI VIET NAM
@aivietnam.edu.vn

Text Preprocessing

Stemming and Lemmatization

- Lemmatization:

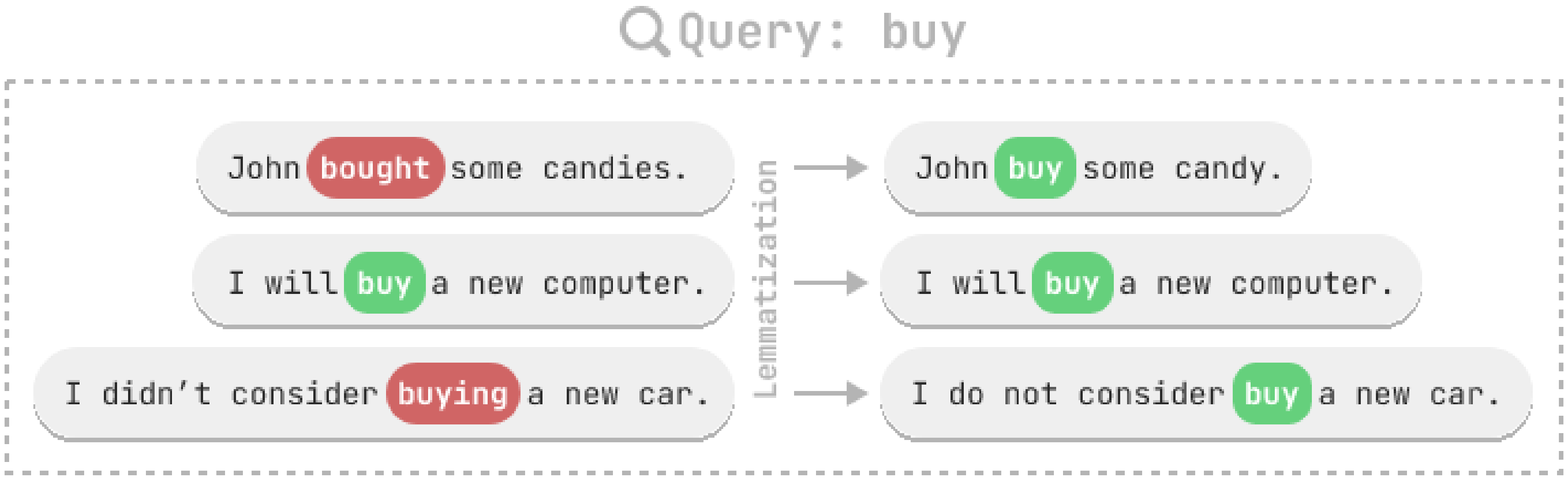
  words have the same root

  despite their surface differences

- Goal: convert words => the same root

  am is are => be

  dinner, dinners => dinner



Text Preprocessing

Stemming and Lemmatization

Morphological parsing

- Morphology: The small meaningful units that make up words
  - Stems: The core meaning-bearing units
  - Affixes: Parts that adhere to stems, often with grammatical functions
- Morphological Parsers:

Highest → High + est

Higher → High + er

Text Preprocessing

Stemming and Lemmatization

Stemming

- Stemming – Simple Lemmatization
  - Naïve version of morphological analysis
  - Chopping off word-final stemming affixes

…ational	→	…ate	relational => relate
…sses	→	…ss	grasses => grass

Text Preprocessing

Tokenization

- Split paragraph, document into sentences
- Use RegEx or library: nltk, genism,… => nltk.sent_tokenize()

Input Text

Tokenization is one of the first step in any NLP pipeline. Tokenization is nothing but splitting the raw text into small chunks of words or sentences, called tokens

Sentence Tokenization

Tokenization is one of the first step in any NLP pipeline.

Tokenization is nothing but splitting the raw text into small chunks of words or sentences, called tokens

Text Preprocessing

Tokenization

- Split paragraph, document into sentences
- Use RegEx or library: nltk, genism,… => nltk.word_tokenize()

Input Text

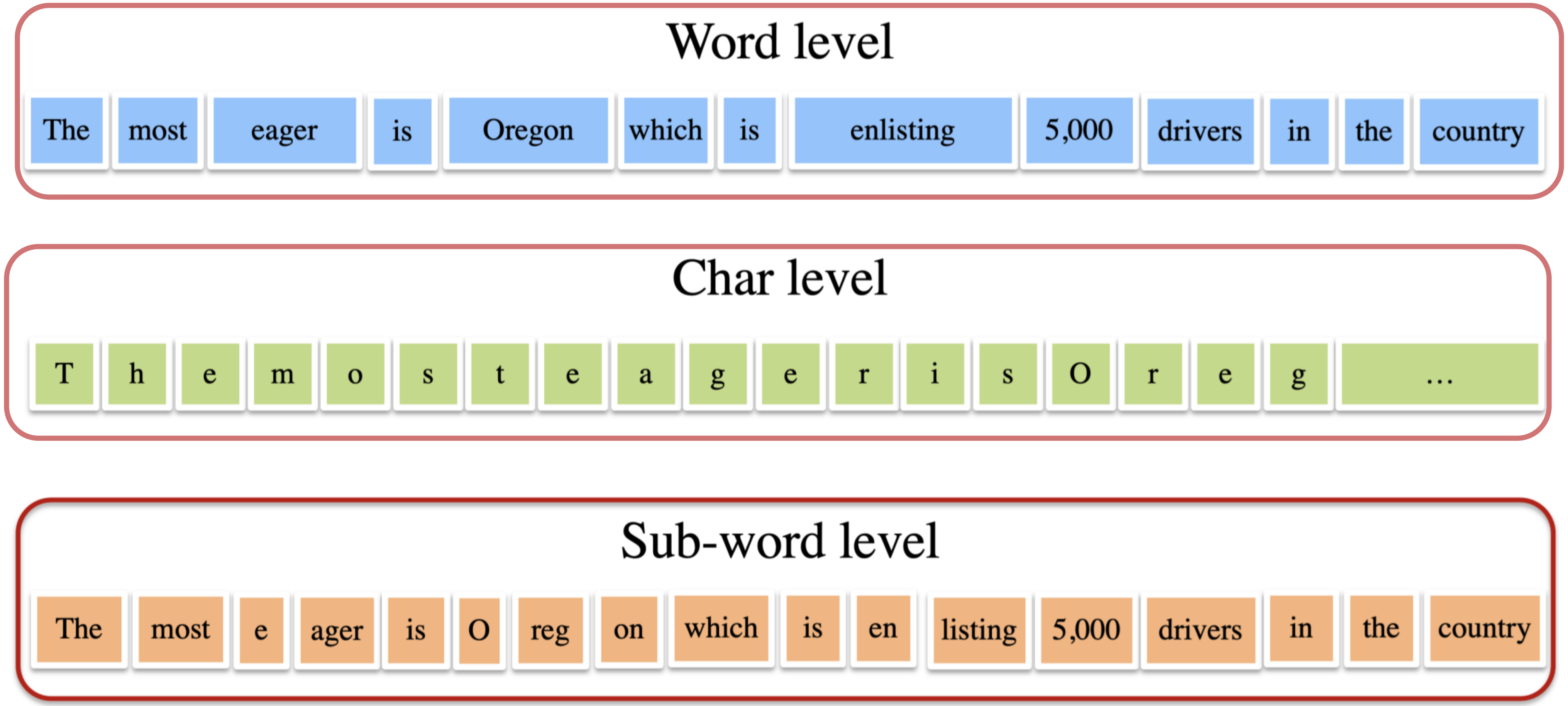
Tokenization is one of the first step in any NLP pipeline.

Word Tokenization

Tokenization	is	one	of
the	first	step	in
any	NLP	pipeline	.

Text Preprocessing

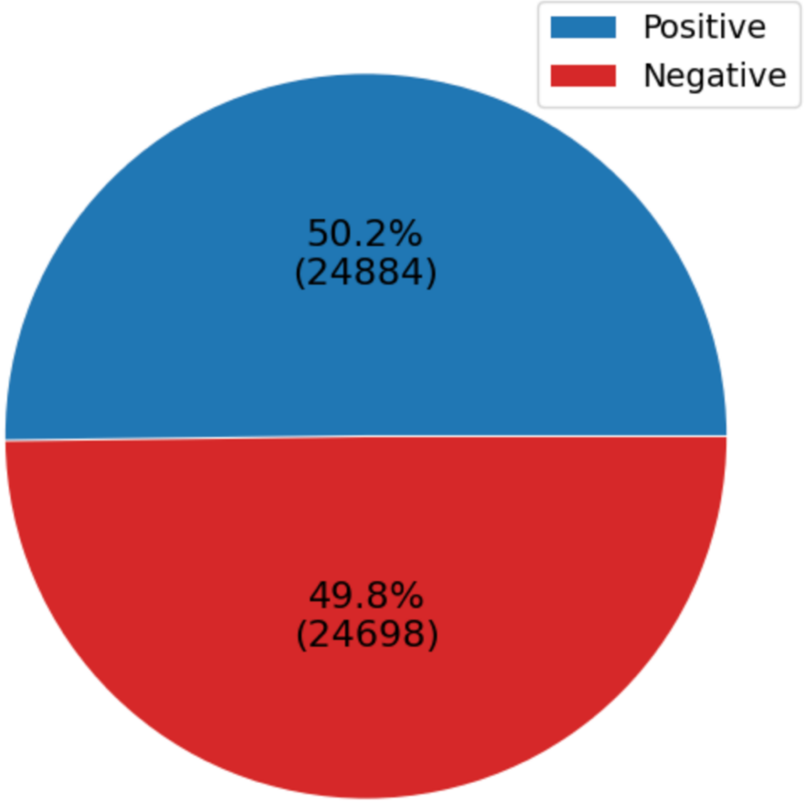
Tokenization



Text Preprocessing

Exploratory Data Analysis (EDA)

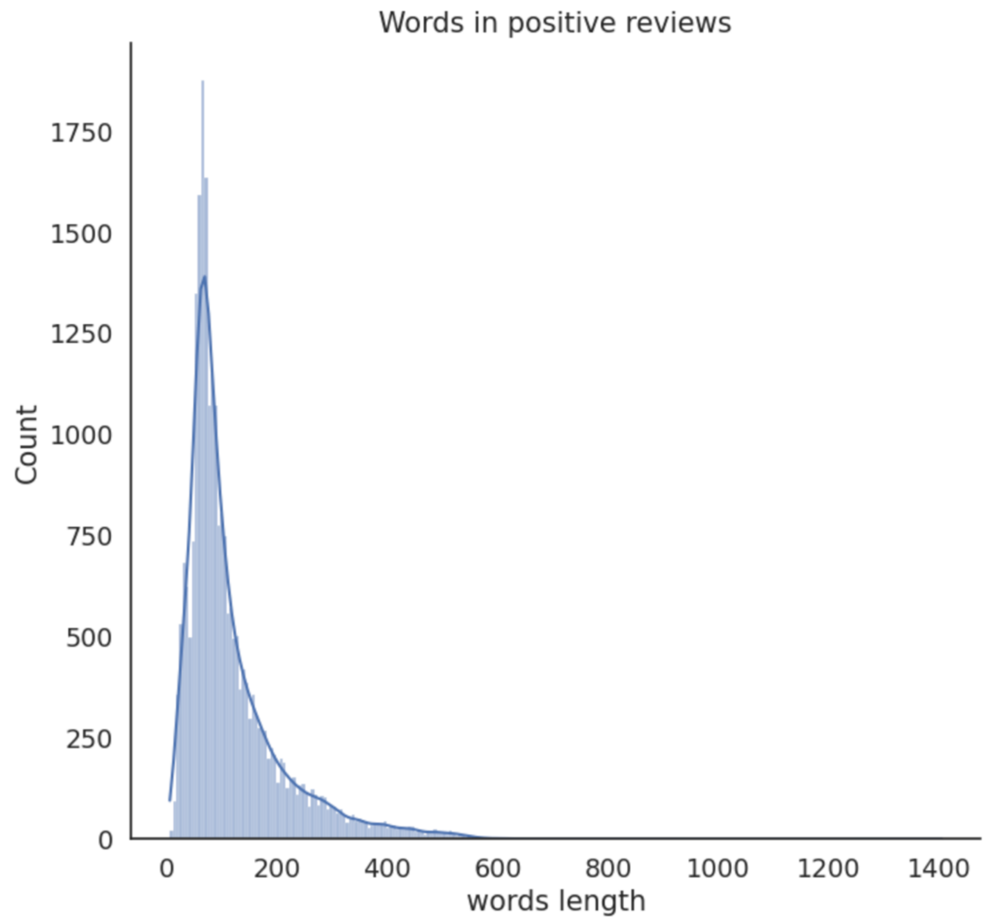
- Frequencies of sentiment labels

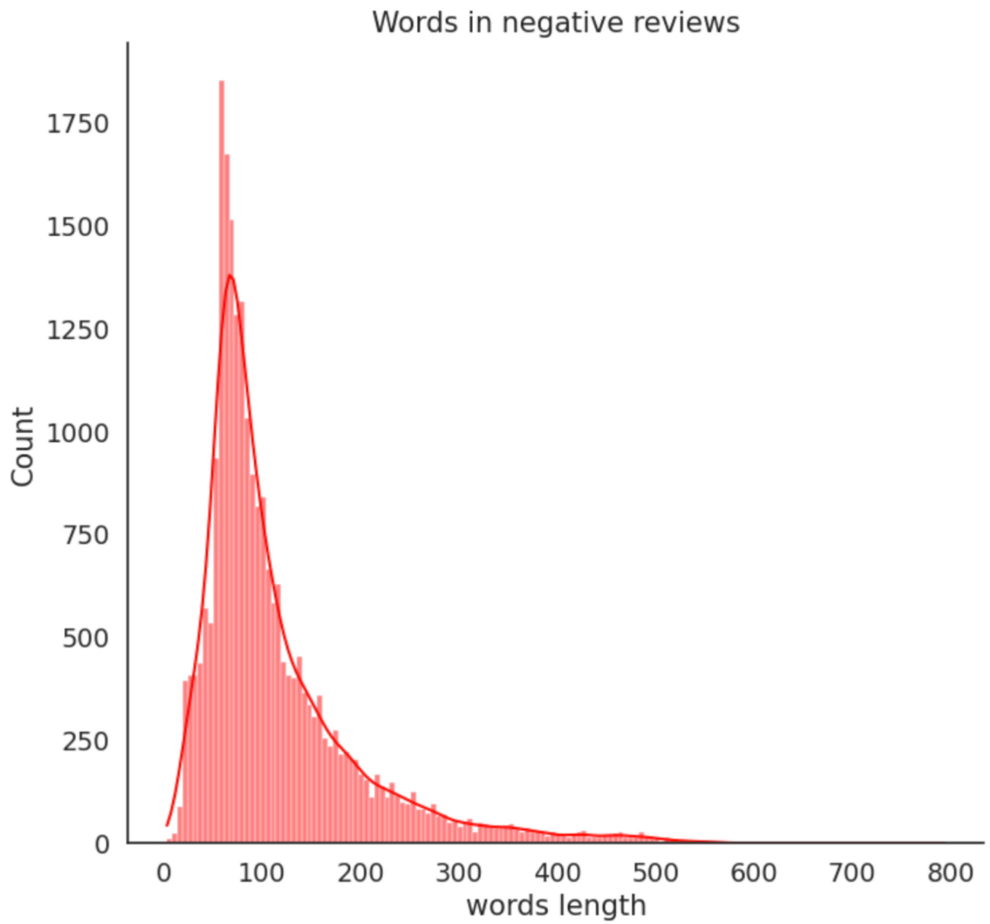


Text Preprocessing

Exploratory Data Analysis (EDA)

- Word lengths





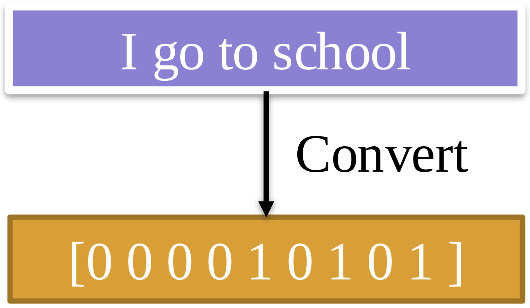
Outline

SECTION 1
Text Classification

SECTION 2
Text Preprocessing

SECTION 3
Text Representation

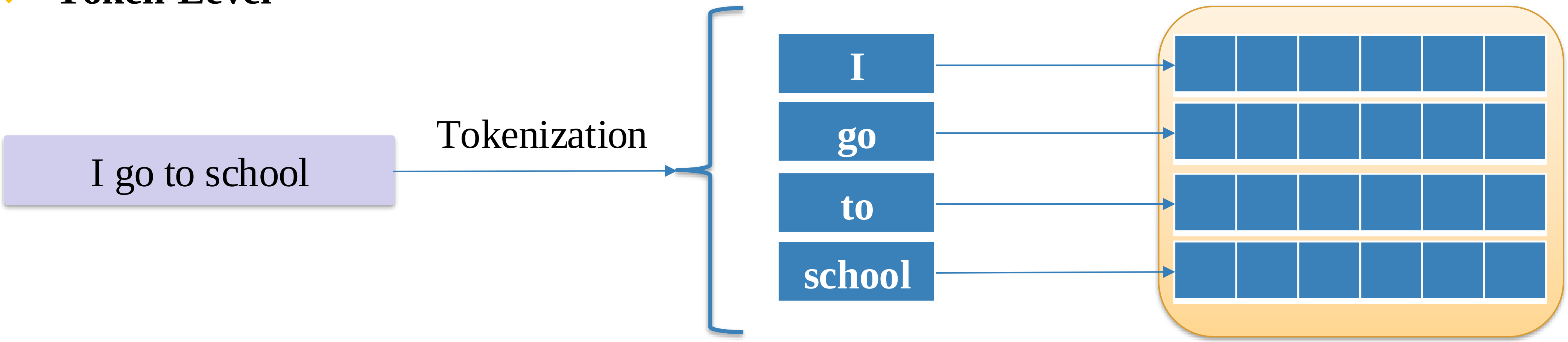
SECTION 4
Classification



Text Representation

Numeric Representation

- **Token-Level**



- **Document-Level**

Text Representation

One-hot Encoding

- Token-Level
- Represented by a V-dimensional binary vector of 0s and 1s
  - All 0s barring the index, index = w_{id}
  - At this index, put 1

Dog bites man.
Man bites dog.
Dog eats meat.
Man eats food.

Preprocessing → Tokenization

[dog, bites, man]
[man, bites, dog]
[dog, eats, meat]
[man, eats, food]

Build → Vocabulary

Vocabulary

IDX	Token
0	bites
1	dog
2	eats
3	food
4	man
5	meat

AI VIET NAM
@aivietnam.edu.vn

Text Representation

One-hot Encoding

- Token-Level
- Represented by a V-dimensional binary vector of 0s and 1s
  - All 0s barring the index, index = w_{id}
  - At this index, put 1

Vocabulary

IDX	Token
0	bites
1	dog
2	eats
3	food
4	man
5	meat

Dog bites man.

[dog, bites, man]

dog → | 0 | 1 | 0 | 0 | 0 | 0 |

bites → | 1 | 0 | 0 | 0 | 0 | 0 |

man → | 0 | 0 | 0 | 0 | 1 | 0 |

```
[[0, 1, 0, 0, 0, 0],
 [1, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 1, 0]]
```

Text Representation

Bag Of Word (BoW)

- Document-Level: Consider text as a bag (collection) of words
- Represented by a V-dimensional

Use: the number of occurrences of the word in the document

Vocabulary

IDX	0	1	2	3	4	5
Token	bites	dog	eats	food	man	meat

Counter

[dog, bites, man]

1	1	0	0	1	0

Text Representation

Bag Of Word (BoW)

- Document-Level: Consider text as a bag (collection) of words
- Represented by a V-dimensional

Use: the number of occurrences of the word in the document

[dog, bites, man]	1	1	0	0	1	0
[man, bites, dog]	1	1	0	0	1	0
[dog, eats, meat]	0	1	1	0	0	1
[man, eats, food]	0	0	0	1	1	0

Text Representation

TF-IDF

$$tf_{t,d} = count(t,d)$$

- Some ways to reduce the raw frequency:
  - Using log space + add 1:

$$tf_{t,d} = log(count(t,d) + 1)$$

  - Divide the number of occurrences by the length of document:

$$tf_{t,d} = \frac{count(t,d)}{len(d)}$$

Text Representation

TF-IDF

$$tf_{t,d} = count(t, d)$$

Example

[dog, bites, man]
[man, bites, dog]
[dog, eats, meat]
[man, eats, food]

	bites	dog	eats	food	man	meat
D1	1/3	1/3	0	0	1/3	0
D2	1/3	1/3	0	0	1/3	0
D3	0	1/3	1/3	0	0	1/3
D4	0	0	1/3	1/3	1/3	0

Text Representation

TF-IDF

$$idf_t = \frac{N}{df_t}$$

- Measures the importance of the word across a corpus

  N: The total number of documents in the corpus

  df_t: The number of documents with term t in them

- Using log space:

$$idf_t = log\frac{N}{df_t} \qquad idf_t = log\frac{N}{df_t} + 1 \qquad idf_t = log\frac{N+1}{df_t + 1} + 1$$

Text Representation

TF-IDF

$$idf_t = ln\frac{N+1}{df_t+1}+1$$

Example

[dog, bites, man]
[man, bites, dog]
[dog, eats, meat]
[man, eats, food]

bites	dog	eats	food	man	meat
1.511	1.223	1.511	1.916	1.223	1.916

Text Representation

TF-IDF

$$w_{t,d} = tf_{t,d} \times idf_t$$

- The weighted value $w_{t,d}$ for word t in document d
- IDF weighs down the terms: very common across a corpus and rare terms
- The TF-IDF vector representation for a document is then simply TF-IDF score for each term in that document.

Text Representation

TF-IDF

Example

[dog, bites, man]
[man, bites, dog]
[dog, eats, meat]
[man, eats, food]

bites	1.511
dog	1.223
Eats	1.511
Food	1.916
Man	1.223
meat	1.916

	bites	dog	eats	food	man	meat
D1	1/3	1/3	0	0	1/3	0
D2	1/3	1/3	0	0	1/3	0
D3	0	1/3	1/3	0	0	1/3
D4	0	0	1/3	1/3	1/3	0

	bites	dog	eats	food	man	meat
D1	0.504	0.408	0	0	0.400	0
D2	0.504	0.408	0	0	0.408	0
D3	0	0.408	0.504	0	0	0.639
D4	0	0	0.504	0.639	0.408	0

Text Representation

TF-IDF

```python
label_encode = LabelEncoder()
y_data = label_encode.fit_transform(df['sentiment'])
```

```python
y_data[:5]
```

```
array([1, 1, 1, 0, 1])
```

```python
x_train, x_test, y_train, y_test = train_test_split(
    x_data, y_data, test_size=0.2, random_state=42
)
```

```python
tfidf_vectorizer = TfidfVectorizer(max_features=10000)
tfidf_vectorizer.fit(x_train, y_train)
```

```
▾ TfidfVectorizer
TfidfVectorizer(max_features=10000)
```

```python
x_train_encoded = tfidf_vectorizer.transform(x_train)
x_test_encoded = tfidf_vectorizer.transform(x_test)
```

```python
x_train_encoded.shape
```

```
(39665, 10000)
```

QUIZ TIME

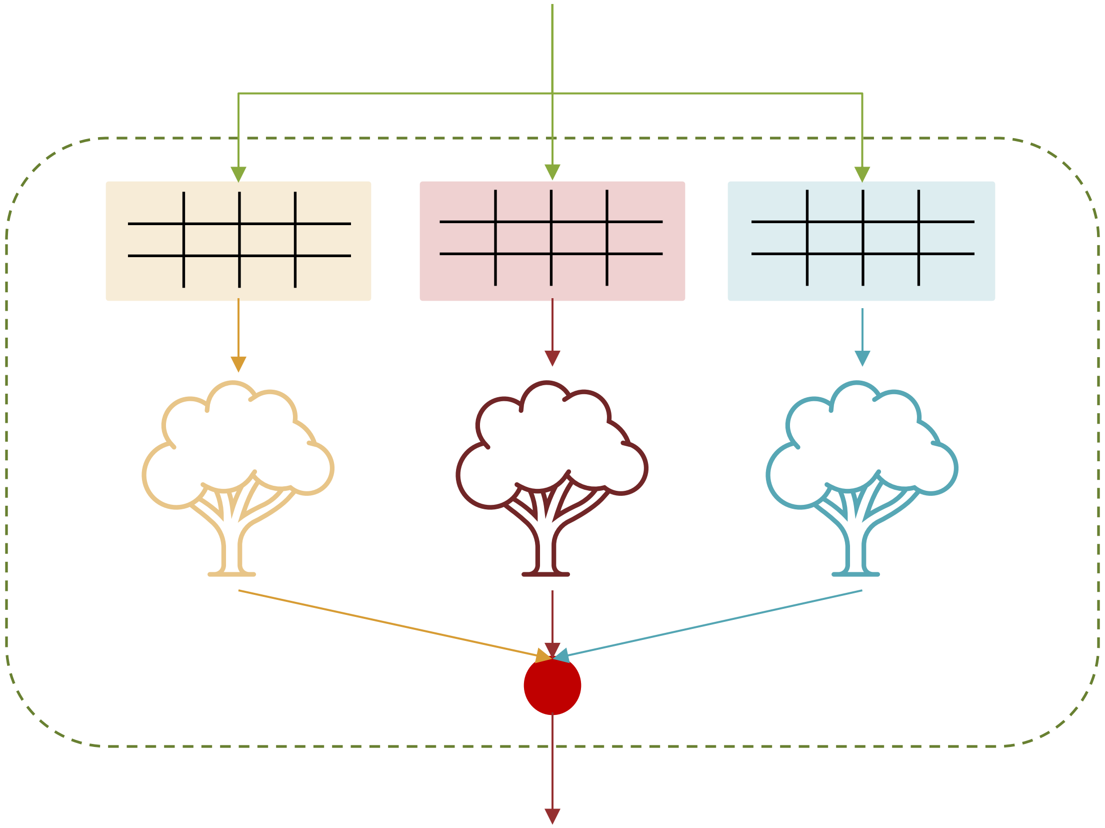
Outline

SECTION 1
Text Classification

SECTION 2
Text Preprocessing

SECTION 3
Text Representation

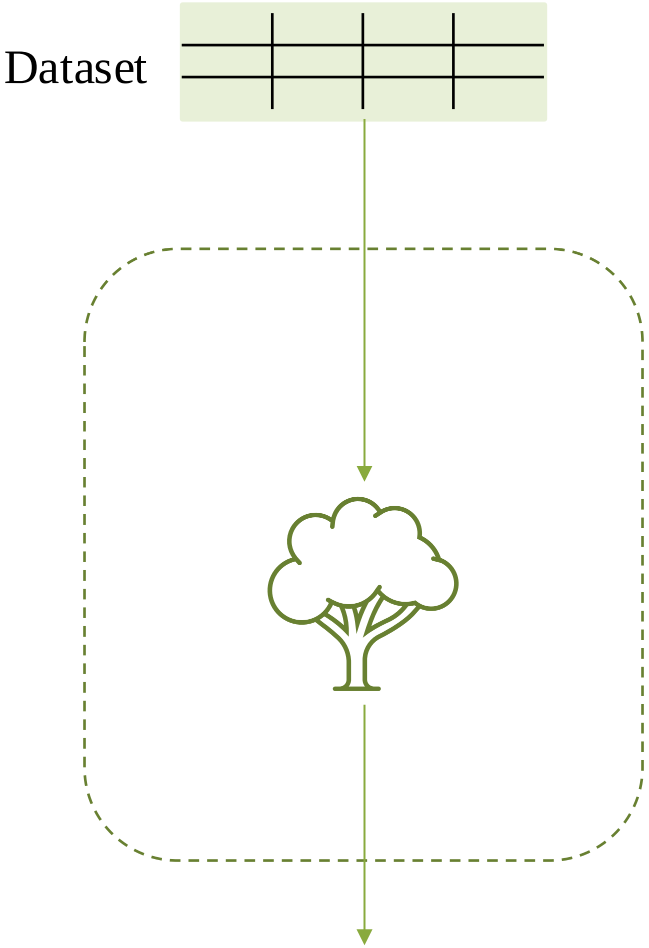
SECTION 4
Classification



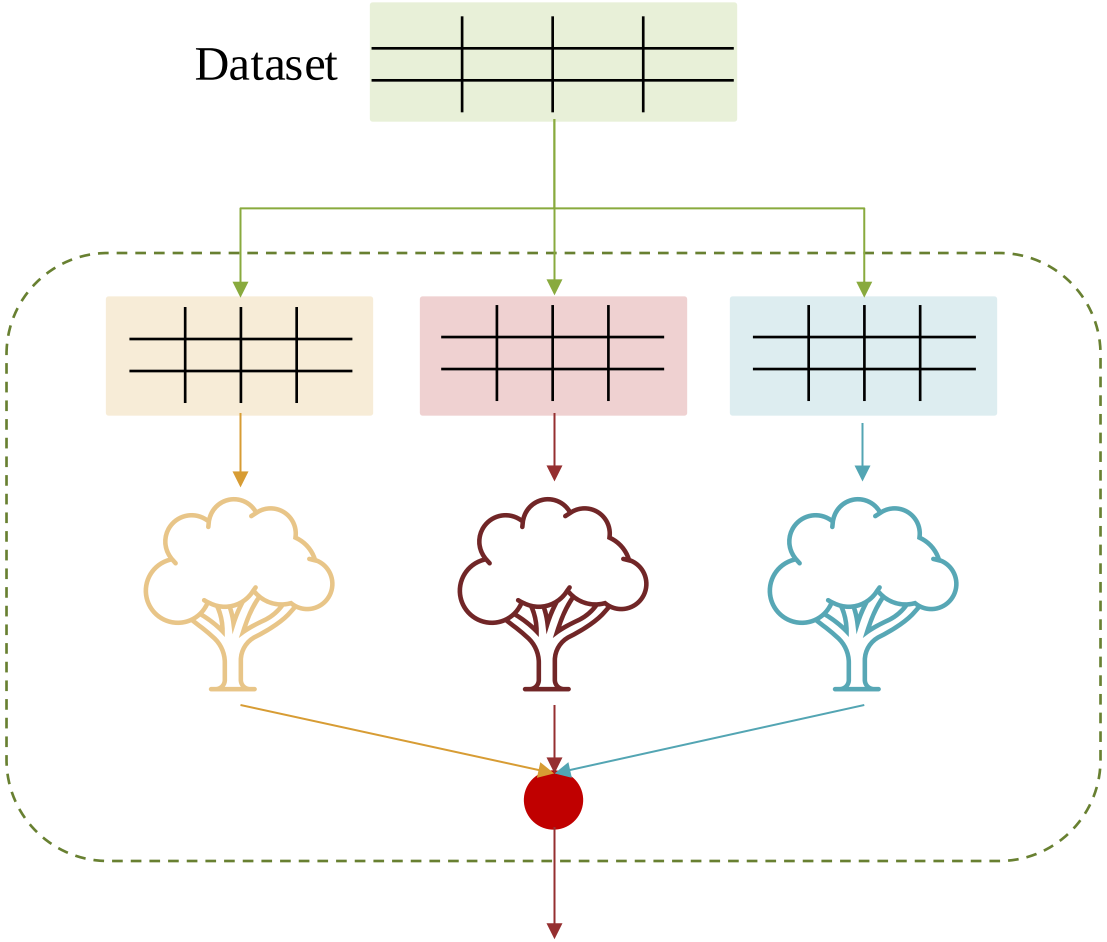
Classification

Decision Tree & Random Forest

Decision Tree



Random Forest



Classification

Decision Tree & Random Forest

Decision Tree

```
dt_classifier = DecisionTreeClassifier(
    criterion='entropy',
    random_state=42
)
dt_classifier.fit(x_train_encoded, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=42)
```

```
y_pred = dt_classifier.predict(x_test_encoded)
```

```
accuracy_score(y_pred, y_test)
```

```
0.7180598971463145
```

Random Forest

```
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(x_train_encoded, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
y_pred = rf_classifier.predict(x_test_encoded)
```

```
accuracy_score(y_pred, y_test)
```

```
0.8420893415347384
```

Classification

AdaBoost & Gradient Boosting

AdaBoost

```python
adb_classifier = AdaBoostClassifier(
    n_estimators=100, random_state=42
)
adb_classifier.fit(x_train_encoded, y_train)
```

```
▾ AdaBoostClassifier
AdaBoostClassifier(n_estimators=100, random_state=42)
```

```python
y_pred = adb_classifier.predict(x_test_encoded)
```

```python
accuracy_score(y_pred, y_test)
```

```
0.8195018654835131
```

Gradient Boosting

```python
gb_classifier = GradientBoostingClassifier(
    n_estimators=100, random_state=42
)
gb_classifier.fit(x_train_encoded, y_train)
```

```
▾ GradientBoostingClassifier
GradientBoostingClassifier(random_state=42)
```

```python
y_pred = gb_classifier.predict(x_test_encoded)
```

```python
accuracy_score(y_pred, y_test)
```

```
0.7968135524856307
```

Classification

XGBoost

```
from xgboost import XGBClassifier

xgb_classifier = XGBClassifier(n_estimators=100)
xgb_classifier.fit(x_train_encoded, y_train)
```

```
▾ XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```
y_pred = xgb_classifier.predict(x_test_encoded)
```

```
accuracy_score(y_pred, y_test)
```

```
0.8490470908540889
```

Summary

Text Classification

- Introduction
- Token-level Text Classification
- Document-level Text Classification
- Sentiment Analysis
- IMDB Dataset

Text Preprocessing

- Duplicate Handling
- Text Cleaning
- EDA
- Tokenization

Text Representation

- Numeric Representation
- One-hot Encoding
- Bag-of-Words (BoW)
- TF-IDF

Classification

- Decision Tree Classifier
- Random Forest Classifier
- Evaluation
- Inference

Thanks!

Any questions?