Natural Language Processing with Disaster Tweets

Problem Statement and Justification

Twitter has become an essential communication platform, especially during emergencies. The widespread use of smartphones enables people to report emergencies in real-time, providing valuable information to disaster relief organizations, news agencies, and governmental bodies. However, distinguishing between genuine emergency tweets and unrelated messages poses a significant challenge.

Organizations that rely on Twitter for real-time disaster monitoring need an automated way to filter relevant tweets accurately. A machine learning model capable of distinguishing real disaster-related tweets from unrelated ones would enhance response times and resource allocation, ensuring efficient emergency management.

Assumptions and Scope

- **The dataset used for training the model consists of labeled tweets, identifying whether they describe a real disaster or not.**
- **Tweets may include textual indicators of disasters, such as figurative language that might lead to misclassification.**
- **The model assumes that tweets are written in English and that linguistic patterns can be used to determine their relevance.**
- **The model focuses on text-based features and does not incorporate external data sources (e.g., images, geolocation).**

Hypothesis (NLP Related):

- **Tweets about real disasters have distinct linguistic patterns, including keywords, urgency markers, and direct mentions of locations or events.**
- **Sentiment analysis and keyword extraction can help differentiate real disaster tweets from metaphorical or unrelated statements.**
- **NLP techniques such as TF-IDF and word embeddings can effectively classify tweets into relevant and non-relevant categories.**

Data Description

The dataset consists of tweets labeled as either related to a disaster (1) or not (0). It contains the following columns:

- **id: A unique identifier for each tweet.**
- **keyword: A keyword extracted from the tweet (may be blank).**
- **location: The location from which the tweet was sent (may be blank).**
- **text: The actual content of the tweet, which serves as the main feature for NLP processing.**
- **target: Present only in the training data, with values:**
  - **1: The tweet is about a real disaster.**
  - **0: The tweet is not related to a disaster.**

In [1]:

```
# libraries
import pandas as pd
```

In [ ]:

```
# loading the data
df = pd.read_csv('train.csv')
```

```
df.tai()
```

Out[ ]:

	id	keyword	location	text	target
7608	10869	NaN	NaN	Two giant cranes holding a bridge collapse int...	1
7609	10870	NaN	NaN	@aria_ahrary @TheTawniest The out of control w...	1
7610	10871	NaN	NaN	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	1
7611	10872	NaN	NaN	Police investigating after an e-bike collided ...	1
7612	10873	NaN	NaN	The Latest: More Homes Razed by Northern Calif...	1

In [5]:

```
df['keyword'].unique()
```

Out[5]:

```
array([nan, 'ablaze', 'accident', 'aftershock', 'airplane%20accident',
       'ambulance', 'annihilated', 'annihilation', 'apocalypse',
       'armageddon', 'army', 'arson', 'arsonist', 'attack', 'attacked',
       'avalanche', 'battle', 'bioterror', 'bioterrorism', 'blaze',
       'blazing', 'bleeding', 'blew%20up', 'blight', 'blizzard', 'blood',
       'bloody', 'blown%20up', 'body%20bag', 'body%20bagging',
       'body%20bags', 'bomb', 'bombed', 'bombing', 'bridge%20collapse',
       'buildings%20burning', 'buildings%20on%20fire', 'burned',
       'burning', 'burning%20buildings', 'bush%20fires', 'casualties',
       'casualty', 'catastrophe', 'catastrophic', 'chemical%20emergency',
       'cliff%20fall', 'collapse', 'collapsed', 'collide', 'collided',
       'collision', 'crash', 'crashed', 'crush', 'crushed', 'curfew',
       'cyclone', 'damage', 'danger', 'dead', 'death', 'deaths', 'debris',
       'deluge', 'deluged', 'demolish', 'demolished', 'demolition',
       'derail', 'derailed', 'derailment', 'desolate', 'desolation',
       'destroy', 'destroyed', 'destruction', 'detonate', 'detonation',
       'devastated', 'devastation', 'disaster', 'displaced', 'drought',
       'drown', 'drowned', 'drowning', 'dust%20storm', 'earthquake',
       'electrocute', 'electrocuted', 'emergency', 'emergency%20plan',
       'emergency%20services', 'engulfed', 'epicentre', 'evacuate',
       'evacuated', 'evacuation', 'explode', 'exploded', 'explosion',
       'eyewitness', 'famine', 'fatal', 'fatalities', 'fatality', 'fear',
       'fire', 'fire%20truck', 'first%20responders', 'flames',
       'flattened', 'flood', 'flooding', 'floods', 'forest%20fire',
       'forest%20fires', 'hail', 'hailstorm', 'harm', 'hazard',
       'hazardous', 'heat%20wave', 'hellfire', 'hijack', 'hijacker',
       'hijacking', 'hostage', 'hostages', 'hurricane', 'injured',
       'injuries', 'injury', 'inundated', 'inundation', 'landslide',
       'lava', 'lightning', 'loud%20bang', 'mass%20murder',
       'mass%20murderer', 'massacre', 'mayhem', 'meltdown', 'military',
       'mudslide', 'natural%20disaster', 'nuclear%20disaster',
       'nuclear%20reactor', 'obliterate', 'obliterated', 'obliteration',
       'oil%20spill', 'outbreak', 'pandemonium', 'panic', 'panicking',
       'police', 'quarantine', 'quarantined', 'radiation%20emergency',
       'rainstorm', 'razed', 'refugees', 'rescue', 'rescued', 'rescuers',
       'riot', 'rioting', 'rubble', 'ruin', 'sandstorm', 'screamed',
       'screaming', 'screams', 'seismic', 'sinkhole', 'sinking', 'siren',
       'sirens', 'smoke', 'snowstorm', 'storm', 'stretcher',
       'structural%20failure', 'suicide%20bomb', 'suicide%20bomber',
       'suicide%20bombing', 'sunk', 'survive', 'survived', 'survivors',
       'terrorism', 'terrorist', 'threat', 'thunder', 'thunderstorm',
       'tornado', 'tragedy', 'trapped', 'trauma', 'traumatised',
       'trouble', 'tsunami', 'twister', 'typhoon', 'upheaval',
       'violent%20storm', 'volcano', 'war%20zone', 'weapon', 'weapons',
       'whirlwind', 'wild%20fires', 'wildfire', 'windstorm', 'wounded',
       'wounds', 'wreck', 'wreckage', 'wrecked'], dtype=object)
```

In [6]:

```
df['location'].unique()
```

Out[6]:

```
array([nan, 'Birmingham', 'Est. September 2012 - Bristol', ...,
       'Vancouver, Canada', 'London ', 'Lincoln'],
      shape=(3342,), dtype=object)
```