Text as Data Pipelines

Jared Edgerton

Why Text Is Data

Large volumes of social data appear as text:

- News articles
- Speeches and statements
- Social media posts
- Reports and documents

Text requires structured transformation before analysis.

Text as a Pipeline

Text analysis is a pipeline:

1. Collection
2. Cleaning
3. Representation
4. Modeling
5. Validation

Each stage shapes inference.

Raw Text Is Not Analysis-Ready

Raw text contains noise:

- Formatting and markup
- Boilerplate and duplicates
- Irrelevant tokens

Preprocessing decisions are theoretical decisions.

Tokenization (Pseudocode)

```
# Input: raw documents
documents = load_corpus()

# Output: list of token lists
tokenized_docs = []

for doc in documents:
    tokens = tokenize(doc)          # word / subword / character
    tokens = normalize(tokens)      # lowercasing, stemming, etc.
    tokens = remove_stopwords(tokens)
    tokenized_docs.append(tokens)
```

Bag-of-Words Construction

```
# Build vocabulary
vocab = build_vocabulary(tokenized_docs)

# Initialize document-term matrix
DTM = zeros(len(documents), len(vocab))

for i, tokens in enumerate(tokenized_docs):
    for token in tokens:
        DTM[i, vocab[token]] += 1
```

BoW prioritizes transparency over semantics.

TF-IDF Weighting

```
# Compute document frequencies
df = count_documents_containing_each_term(DTM)

# Compute IDF
idf = log(N_documents / df)

# Apply TF-IDF weighting
TFIDF = DTM * idf
```

TF-IDF downweights ubiquitous terms.

Embeddings (Conceptual)

```
# Load pretrained embedding model
model = load_embedding_model()

embeddings = []

for doc in documents:
    vec = model.encode(doc)
    embeddings.append(vec)
```

Embeddings shift interpretation from words to geometry.

Topic Models (Conceptual)

```
# Input: document-term matrix
initialize K topics

repeat until convergence:
    update topic-word distributions
    update document-topic distributions

return topics, document_assignments
```

Topics summarize patterns; they are not ground truth.

Transformer Workflow (Lightweight)

```
# Tokenize with model-specific tokenizer
tokens = tokenizer(documents)

# Encode context-aware representations
hidden_states = transformer(tokens)

# Pool representations (e.g., CLS token)
doc_embeddings = pool(hidden_states)

# Optional: fine-tune for downstream task
predictions = classifier(doc_embeddings)
```

Transformers model context explicitly.

Validation and Stability Checks

```
# Sensitivity to preprocessing
for variant in preprocessing_variants:
    results = run_pipeline(variant)
    compare(results)

# Stability across random seeds
for seed in seeds:
    set_seed(seed)
    results = run_pipeline()
```

Text models can fail silently.

Error and Bias Propagation

Bias enters through:

- Corpus selection
- Tokenization choices
- Pretrained models
- Labeling procedures

Upstream choices dominate downstream results.

Documentation Requirements

Every text pipeline should record:

- Corpus source
- Preprocessing steps
- Representation choice
- Model parameters
- Validation checks

Auditable pipelines are reproducible pipelines.

What We Emphasize in Practice

- Pipelines over single models
- Simplicity before complexity
- Validation over accuracy
- Transparency in preprocessing

Discussion

- Where does meaning get lost?
- Which pipeline decisions matter most?
- How should uncertainty be communicated?