Annotation Tools Advanced

Feature Engineering

Jin Zhao

Brandeis University

March 16, 2025

Today's Agenda

1. Argilla for RLHF data collection
2. Custom annotation interfaces
3. From annotations to features
4. Feature engineering for NLP
5. Representation learning
6. Handling rare classes

Project: Draft 2 guidelines due
Assignment: HW 2 due

Argilla: For RLHF and Feedback

Designed specifically for LLM workflows

Key features:

- Native preference annotation support
- Hugging Face ecosystem integration
- Built-in weak supervision
- Collaborative feedback collection
- Active learning capabilities

Best for:

- RLHF preference data
- LLM evaluation and red-teaming
- Human-AI collaborative annotation
- Feedback collection at scale

Argilla vs. Label Studio

Feature	Label Studio	Argilla
General annotation	✓	✓
Multi-modal	✓✓	✓
Preference annotation	Limited	✓✓
RLHF workflows	Limited	✓✓
Hugging Face integration	✓	✓✓
Active learning	✓	✓
Self-hosted	✓	✓

Choose based on your task:

- General NLP annotation → Label Studio
- RLHF/preference data → Argilla
- Multi-modal → Label Studio

Custom Annotation Interfaces

When built-in tools aren't enough:

Options:

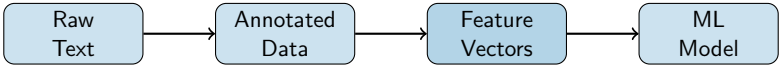
1. **Extend existing tools:** Custom Label Studio templates
2. **Streamlit/Gradio:** Quick web interfaces
3. **Custom web app:** Full control, more effort
4. **Jupyter notebooks:** Interactive annotation

When to build custom:

- Unique annotation type
- Complex workflows
- Integration with specific systems
- Special visualization needs

From Annotations to Features

ML models need features, not raw annotations



Feature engineering: Converting text + annotations to numbers

Basic Text Features

Bag of Words (BoW):

- Each word is a feature
- Count or binary occurrence
- Simple but effective baseline

Example:
"I love this movie" $\rightarrow$ [I:1, love:1, this:1, movie:1]

TF-IDF:

- Weight by term frequency $\times$ inverse document frequency
- Down-weights common words
- Up-weights distinctive words

Feature Matrix

Representing features as a matrix:

	charming	delightfully	hollow	picture	uplifting	...
"A pleasant..."	0	0	0	1	1	...
"A charming..."	1	1	0	0	0	...
"A hollow..."	0	0	1	0	0	...

Sparse representation: Most entries are 0

In Python: Use `scipy.sparse` or `DictVectorizer`

Additional Features

Beyond bag of words:

- **N-grams:** Word pairs/triples ("not good" vs. "good")
- **Character n-grams:** Subword patterns
- **Length features:** Word count, sentence count
- **Lexical features:** Punctuation, capitalization
- **Metadata:** Time, author, source

Feature binning:

- Convert continuous to categorical
- Example: length $\rightarrow$ short/medium/long
- Reduces dimensionality

Scikit-Learn DictVectorizer

Convert feature dictionaries to matrices:

```
from sklearn.feature_extraction import DictVectorizer
```

Input:

- List of dictionaries
- Each dict: {feature: value}

Output:

- Sparse matrix
- Each row = one instance
- Each column = one feature

Handles: Both numeric and categorical features

Feature Weights in Classification

Models learn weights for features:

Feature	Positive weight	Negative weight
“charming”	3.0	0.5
“delightful”	2.0	0.2
“hollow”	-0.4	2.4
“terrible”	-1.5	3.5

Prediction: Weighted sum of feature values

Learning algorithms: Naive Bayes, Logistic Regression, SVM
These are covered in COSI 114b and 134

Handling Rare Classes

Problem: Some labels have few examples

Options:

1. **Merge classes:** Combine rare labels into "Other"
2. **Remove:** Drop if not essential to task
3. **Oversample:** Duplicate rare examples
4. **Undersample:** Reduce frequent examples
5. **Weighted loss:** Penalize errors on rare classes more
6. **Better models:** SVM, SetFit handle imbalance better

Key question: Is the rare class important for your task?

Filtering Vocabulary

Zipf's Law: Most words are rare

Problem:

- Most frequent words may be uninformative ("the", "a")
- Least frequent words have too few examples to learn from
- Full vocabulary = slow model training

Solutions:

- Remove stopwords
- Set minimum document frequency (ignore very rare)
- Set maximum document frequency (ignore very common)
- Limit total vocabulary size

Modern Representations

Beyond bag of words:

Word embeddings:

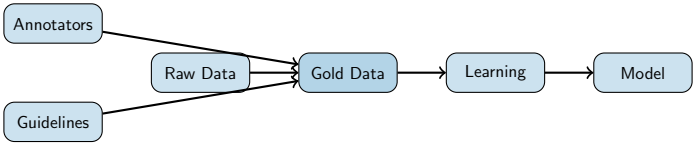
- Word2Vec, GloVe
- Dense vectors (300 dimensions)
- Capture semantic similarity

Contextual embeddings:

- BERT, RoBERTa, GPT
- Different representation per context
- State-of-the-art performance

For this course: Focus on annotations, use pre-built models

Blueprint: Annotation to Model



Key insight: Annotation quality affects model quality
"Garbage in, garbage out"

Next Class: Human-AI Collaboration

Lecture 17 (Mar 18): Human-AI Collaborative Annotation

Topics:

- Human-in-the-loop paradigm
- LLM pre-annotation + human correction
- Active learning with LLMs
- Efficiency gains from hybrid approaches
- When to trust LLM annotations

Project: Continue annotation work

Key Takeaways

1. **Argilla** is designed for RLHF and preference data
2. **Feature engineering** converts text to numbers for ML
3. **Bag of words** is simple but effective baseline
4. **Rare classes** need special handling
5. **Vocabulary filtering** improves efficiency
6. **Modern embeddings** (BERT) often outperform BoW

Questions?

Office Hours: Wednesdays 1-3pm, Volen 109

jinzhao@brandeis.edu