Text as Data Pipelines

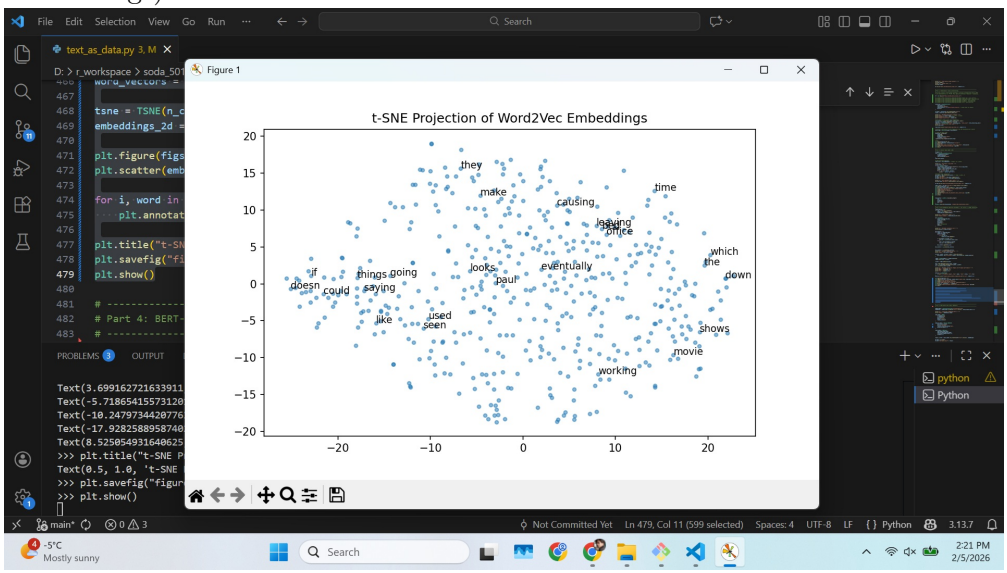
Yasin Shafi

5 February 2026

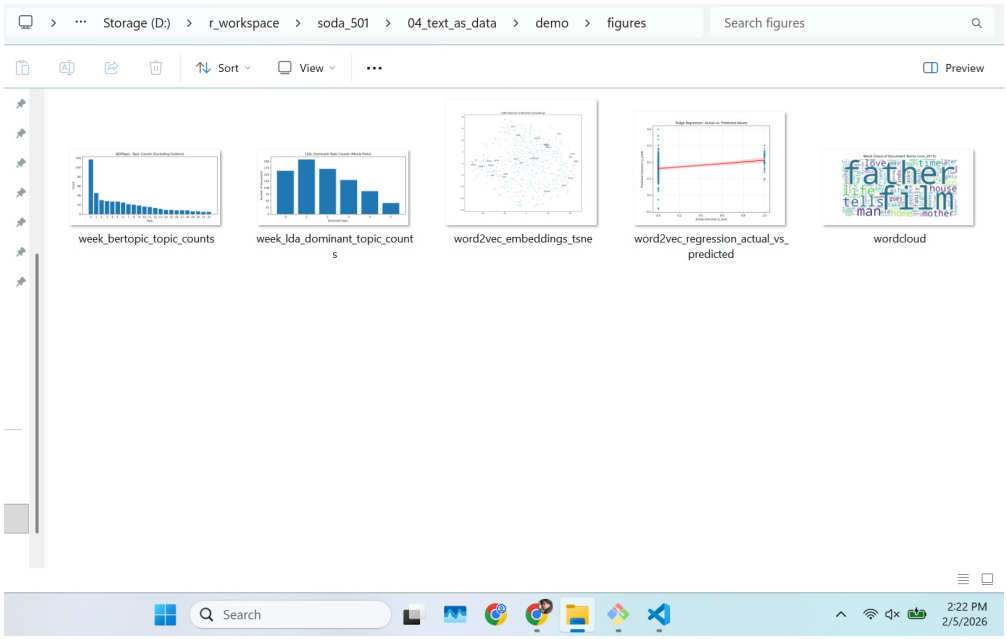
Note on data. This problem set uses a corpus of **movie plot summaries** derived from a public archival dataset (the CMU Movie Summary Corpus, provided by the instructor). Movie plots are used as a neutral, non-political text source that allows us to focus on *pipeline design, representation choices, and modeling decisions* rather than substantive interpretation. You should treat this corpus as a stand-in for real-world social text data.

Start Off: Verifying Your Environment

1. **Environment check (required).** Submit proof that you successfully ran the full tutorial code on your machine. You may submit *one* of the following:
   - A screenshot or text file showing console output that includes the printed representation shapes (document–term matrix, Word2Vec document vectors, and transformer embeddings).



   - A screenshot of your `figures/` directory showing generated plots with timestamps.



- A Git commit (hash or screenshot) that includes at least one generated figure or output file.
- A short log file (e.g., `run_log.txt`) containing printed diagnostics and evaluation metrics.

Conceptual Questions

Please write three to ten sentence explanations for each of the following questions. **You are only required to answer ONE of the two questions below.**

2. Compare **bag-of-words** representations (e.g., a document–term matrix) to **embedding-based** representations (Word2Vec or transformer embeddings). Discuss two trade-offs involving interpretability, robustness, or downstream modeling. Give one concrete example of when you would prefer each representation in social science research.
   Answer: Bag-of-words (BoW) approach treats documents as unordered collections of word frequencies, creating a simple document-term matrix where each row is a document and each column represents a word's count or presence. Embedding-based representations like Word2Vec or transformer models instead map words or documents into dense, continuous vector spaces where semantic similarity is captured by proximity, allowing "government" and "administration" to have similar representations even if they never co-occur in the same document.
   Trade-off: Interpretability versus Semantic Richness
   BoW representations are highly interpretable. We can directly inspect which words drove a classification or topic assignment, making it easy to audit and explain results to non-technical audiences. Embeddings, however, are black boxes where individual dimensions lack clear meaning, though they capture nuanced semantic relationships and context that BoW completely misses.
   Trade-off: Robustness to Corpus Size and Pre-training Bias
   BoW models are transparent about their limitations. If a word is not in the corpus, it simply doesn't exist in the representation. Embeddings, especially pre-trained transformers, inherit biases from their training data (often general web text) that may not align with the domain,

but they perform much better on small datasets by leveraging external knowledge.

Concrete Example

We may prefer BoW when analyzing protest event descriptions to extract structured information like locations, actors, and tactics. The interpretability is crucial for validating that the model is actually identifying "police" and "demonstrators" rather than picking up spurious patterns, and the vocabulary is relatively constrained and domain-specific. Conversely, we may prefer embeddings when measuring ideological similarity across political speeches where the same concept is expressed through varied language. Embeddings would recognize that "tax relief" and "fiscal responsibility" belong to similar ideological spaces even without lexical overlap, which BoW would miss entirely.

Applied Exercises

Response: Find this week's work here: `https://github.com/yasinshafi/soda501/tree/main/04_text_as_data`.

Use the code in the week's Python tutorial and the lecture slides to answer the following questions. **You are only required to answer TWO out of the three questions below.**

4. **Topic model (LDA): tokenization → document–term matrix → topics.**
   Using the Week Python tutorial (classic LDA section) and the movie plot corpus:
   - Load the movie corpus (e.g., `data_raw/week_movie_corpus.csv`) and create a document–term matrix using `CountVectorizer`.
   - Fit an LDA model with a chosen number of topics (e.g., $K = 6$).
   - Report the top 8–12 words for each topic.
   - Assign each document a dominant topic and create a plot showing the number of documents per dominant topic.
   - In 3–8 sentences, interpret at least **two** topics and explain how preprocessing choices (e.g., stopwords, `min_df`, token pattern) affect topic quality.
5. **Word embedding regression: Word2Vec → document vectors → prediction.**
   Using the Word2Vec section of the tutorial and the movie corpus:
   - Tokenize the movie plots (simple tokenization is sufficient).
   - Train a Word2Vec model on the corpus and construct a document-level embedding for each plot by averaging word vectors.
   - Fit a Ridge regression model to predict the provided outcome variable `y_outcome` (a binary indicator derived from movie genres). Use a train/test split.
   - Report out-of-sample **MAE**, **RMSE**, and R^2, and include a predicted-vs-actual diagnostic plot.
   - In 3–8 sentences, explain what it means to regress an outcome on text embeddings and discuss at least **two** limitations of this approach (e.g., interpretability, domain shift, embedding quality).

     Answer: Regressing an outcome on text embeddings means using dense vector representations of documents as predictive features in a regression model. Each document is converted into a fixed-length numerical vector that captures semantic content, and these vectors serve as independent variables to predict a continuous or binary outcome. In this analysis, we averaged Word2Vec word vectors to create document-level embeddings,

then used Ridge regression to predict whether a movie belongs to the "action" genre based solely on its plot summary's semantic representation.

Limitations:

a) Interpretability is severely compromised - while traditional bag-of-words models allow us to identify which specific words drive predictions (e.g., "explosion" predicting action movies), embeddings collapse meaning into opaque numerical dimensions that cannot be easily traced back to interpretable linguistic features, making it difficult to audit model decisions or understand why certain predictions were made.

b) Embedding quality is highly sensitive to the training corpus and can introduce systematic biases - Word2Vec trained on movie plots learns semantic associations specific to that domain, so embeddings may fail to capture relevant distinctions if the training corpus is too small, unrepresentative, or contains systematic gaps in coverage of important concepts. Additionally, averaging word vectors to create document embeddings is a crude aggregation strategy that discards word order, sentence structure, and contextual nuances, potentially losing critical information about how meaning is constructed through narrative flow in plot summaries.

6. **BERTopic: transformer embeddings → clustering → topic summaries.**
   Using the BERTopic section of the tutorial:
   - Fit a BERTopic model on the movie plots and generate a topic summary table.
   - Create a plot showing the number of documents assigned to each topic (excluding outliers if present).
   - Report (i) the number of topics discovered (excluding outliers) and (ii) the share of documents assigned to the outlier topic (Topic $= -1$), if applicable.
   - In 3–10 sentences, compare BERTopic to LDA on this dataset. Discuss topic interpretability, sensitivity to preprocessing, and computational cost. Identify one setting where BERTopic is likely to outperform LDA and one where LDA may be preferable.

   Answer: BERTopic and LDA represent fundamentally different approaches to topic modeling. LDA uses a probabilistic generative model over bag-of-words representations, while BERTopic leverages pre-trained transformer embeddings and density-based clustering to discover topics. On this movie plot dataset, BERTopic produces more semantically coherent topics because it captures contextual meaning through BERT embeddings, whereas LDA relies solely on word co-occurrence patterns that may miss nuanced semantic relationships in narrative text.

   Topic interpretability differs substantially between the two methods. LDA topics are defined by probability distributions over words, making it straightforward to inspect the top words and understand what drives each topic. BERTopic topics emerge from clustering in embedding space and are represented by extracting representative terms post-hoc, which can sometimes produce less transparent topic definitions. However, BERTopic often generates topics that align better with human intuitions about semantic similarity because embeddings capture meaning beyond simple word co-occurrence.

   Sensitivity to preprocessing is a critical distinction. LDA is highly sensitive to preprocessing decisions - choices about stopword removal, minimum document frequency, and vocabulary size fundamentally reshape the document-term matrix and can dramatically alter discovered topics. BERTopic is more robust to preprocessing because transformer embeddings are pre-trained on massive corpora and encode semantic information that survives minimal preprocessing, though this also means BERTopic inherits biases from its pre-training data.

Computational cost favors LDA for large-scale applications. LDA with batch learning can process large corpora efficiently using only sparse count matrices, while BERTopic requires generating dense embeddings for every document (computationally expensive with transformer models) and performing UMAP dimensionality reduction plus HDBSCAN clustering, making it substantially slower and more memory-intensive.

BERTopic would outperform LDA when analyzing short, semantically diverse texts where context matters - for example, analyzing Twitter posts about political events. LDA would be preferable when working with very large corpora (millions of documents) where computational constraints matter, or when analyzing highly structured domain-specific text with consistent terminology where interpretability and auditability are paramount, such as analyzing legal documents or scientific abstracts where stakeholders need to understand exactly which terms define each topic.

7. **In-Class Extension: Multinomial Outcome (Required for Participation).**
In the tutorial, the outcome variable `y_outcome` is binary. Extend this analysis to a **multinomial outcome** using movie genres.
   - Redefine the outcome variable so that each document belongs to one of several genre categories (e.g., Action, Comedy, Drama, Horror, Other).
   - Replace the regression model with an appropriate multinomial model.
   - Evaluate performance using accuracy and a confusion matrix.

   Pseudo code (outline only):

```
# Map genre labels to integers
genre_to_label = {"action": 0, "comedy": 1, "drama": 2, "horror": 3, "other": 4}
y_multiclass = map(true_genre, genre_to_label)

# Split data
X_train, X_test, y_train, y_test = train_test_split(embeddings, y_multiclass)

# Fit multinomial model
fit multinomial_logit(X_train, y_train)

# Predict and evaluate
y_pred = predict(X_test)
compute accuracy
compute confusion_matrix
```

   In 5–7 sentences, discuss how changing the outcome definition affects interpretation and model performance, even though the text representation and pipeline remain the same.

   Answer: Changing from a binary outcome (action vs. non-action) to a multinomial outcome (five genre categories) fundamentally alters what the model learns and how we interpret its performance, even though the text embeddings remain identical. With binary classification, the model only needs to learn a single decision boundary separating action movies from everything else, while multinomial classification requires the model to learn multiple decision boundaries distinguishing between five distinct genre categories simultaneously, making the learning problem substantially more complex. This increased complexity typically results in lower overall accuracy because the model must make finer-grained distinctions - for instance, distinguishing drama from horror based on plot summaries may be much harder than simply identifying action movies.

8. **Challenge Question (Optional — if you finish early).**
   Use transformer embeddings for prediction and compare results to the Word2Vec-based models.
   - Compute document embeddings using a pretrained transformer model.
   - Fit a simple predictive model using these embeddings.
   - Compare performance to the Word2Vec results.
   - In 4–8 sentences, discuss what this comparison suggests about representation choice in text-as-data projects.