Text as Data Pipelines

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.
3. A text-as-data pipeline is more than a model. Describe a practical pipeline architecture for a research project that uses topic models and embeddings. In your answer, address: (i) where you would cache intermediate artifacts (e.g., tokenized text, vocabularies, embeddings), (ii) how you would document versions and randomness (e.g., seeds), and (iii) one way you would prevent data leakage or overfitting when evaluating models.

Applied Exercises

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).
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.
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.

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.