Text as Data Pipelines

Jared Edgerton

Note on data. This problem set uses a **synthetic** text corpus created for the Week code tutorial (or an instructor-provided file). The goal is to practice end-to-end text pipelines—not to make substantive claims about real-world political texts.

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.**

1. Compare **bag-of-words** representations (e.g., a document-term matrix) to **embedding** representations (Word2Vec / transformer embeddings). Discuss two trade-offs involving interpretability, robustness, and downstream modeling. Give one concrete example of when you would prefer each representation in social science research.
2. 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, vocabulary, embeddings), (ii) how you would document versions and randomness (seeds), and (iii) one way you would prevent data leakage or overfitting when evaluating models.

Applied Exercises

Use the code in the week's code tutorial and the lecture slides to answer the following questions.

3. **Topic model (LDA): tokenization → document-term matrix → topics.** Using the Week Python tutorial (classic LDA section):
   - Load the synthetic corpus (e.g., `data_raw/week_synthetic_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 5–8 sentences, interpret at least **two** topics (what they seem to capture) and explain how preprocessing choices (stopwords, `min_df`, token pattern) affect topic quality.
4. **Word embedding regression: Word2Vec → document vectors → out-of-sample prediction.** Using the Word2Vec regression section of the tutorial:
   - Tokenize the documents (simple tokenization is fine).
   - Train Word2Vec on the corpus and create a document embedding for each document by averaging its word vectors.

- Fit a Ridge regression model to predict the provided outcome variable (e.g., `y_outcome`). Use a train/test split and report out-of-sample **MAE**, **RMSE**, and R^2.
- Create a predicted-vs-actual scatterplot (or another clear diagnostic plot).
- In 5–8 sentences, explain what it means to regress an outcome on embeddings and discuss at least **two** limitations of this approach (e.g., interpretability, domain shift, embedding quality, small corpora).

5. **BERTopic: transformer embeddings → clustering → topic summaries.** Using the BERTopic section of the tutorial:
   - Fit BERTopic and generate a topic summary table (e.g., `get_topic_info()`).
   - Create a plot of topic counts (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 6–10 sentences, compare BERTopic to LDA on this dataset: discuss topic interpretability, sensitivity to preprocessing, and computational cost. Identify one situation where BERTopic is likely to outperform LDA and one where LDA may be preferable.
6. **Challenge Question (Optional — if you finish early):** Use **transformer embeddings** for prediction and compare to Word2Vec regression.
   - Compute sentence/document embeddings using a lightweight model (e.g., `sentence-transformers/all`
   - Fit a Ridge regression (or another simple model) to predict `y_outcome` using the transformer embeddings.
   - Report MAE, RMSE, and R^2, and compare them directly to your Word2Vec regression results.
   - In 4–8 sentences, interpret why the transformer approach does better or worse here, and what that implies for choosing representations in real text-as-data projects.