ARI_425_HW_1

James Ryan

February 2026

1 Data

Given the assignment parameters, I had some lofty goals about the dataset. The Congressional one looked really interesting and I spent much more time than I should have trying to make it work. Given my short timeframe, I should have simply gotten a canned corpus, which is eventually exactly what I did. The Reuters dataset is apparently an NLP standard: 21,000 or so articles pulled from the Reuters newsfeed in 1987. So standard, in fact, that nltk has a pull for it, which I also didn't discover until after I spent some time trying to learn SGML for the ancient copy I found on the net. Go me.

Reuters has a lot of good points. Its size, as stated 21,000 documents, for one. It is also categorized into 135 categories so there's a lot of potential for testing with a fairly small footprint for easy processing with the moderate CPU I'm using on my elderly MacBook Pro. Excellent. Some of the pertinent features, both general and used in these experiments, are summarized in Table1.

Characteristic	**Description**
Dataset Name	Reuters-21578
Source	Reuters Newswire (1987)
Number of Documents	21,578 news articles
Docs after Pre-processing	9,160 news articles
Language	English
Number of Categories	135 total topic categories
Categories After Pre-processing	65 topic categories
Document Length	Short to medium-length newswire articles
Average number of tokens per article	71

Table 1: Key characteristics of the Reuters dataset.

2 Methodology

Starting with an eye toward self-deprecation, and not at all because of the code quality in any way (you rush a miracle man, you get crummy miracles), we'll call this first part the Clunky method. What I turn in with this assignment is a LOT cleaner than what i was actually testing with, full of false starts and somewhat NSFW commentary.

2.1 Step One: Bag of Words (BoW)

After my somewhat rocky dataset start, it was time for pre-processing. I leveraged my code from HW0, which already tackled tokenization, stopwords, stemming, lowercasing and rare words. I used all but the last one in this project. Rare words are a bit of a problem when processing small documents because with short text samples, word rarity, in my opinion, should not be attempted at the document level because no words really appear a lot in one or two paragraphs, as many of these stories are. If removing rare words it should be done by pruning at the corpus level, something which I would try but frankly I'm out of time.

After pre-processing, it became time to tackle the actual project. My pre-processing spotted me a BoW representation of each doc, so I was reasonably ahead of the game there. I was faced with a bit of a challenge in that the documents in Reuters can have more than one category associated with them. In the end I simplified this by eliminating multi-category documents, which still left around 10,000 or so examples for analysis. From there, I took a dictionary of all of the docs and added keys for the full text (in case I needed it for something), topic, and BoW transformations. I concatenated all of the unique words from all of the documents into a common vocabulary of 396,412 unique words. I also grouped individual documents with the same topic by concatenating the individual word counts.

2.2 Step Two: Probabilities

From there it was time to compute probabilities. I used Laplace Smoothing, so the conditional word probabilities became:

$$P(w \mid c) = \frac{N_{w,c} + 1}{\sum_{w' \in V} N_{w',c} + |V|} \tag{1}$$

God, I love LaTeX. The document editor. And ChatGPT, which is much better at spitting out formulas than I am. But I digress.

This made the document probabilities:

$$P(c \mid d) \propto P(c) \prod_{i=1}^{n} P(w_i \mid c)^{\text{count}(w_i,d)} \tag{2}$$

This actually wasn't too bad to put in place. I wrote some display methods and got the results in Table2. ChatGPT to the rescue again for that table. I really don't know how I used to author without it any more. It saves an incredible amount of time.

Topic	**Top 10 Words**
earn	vs, mln, ct, net, dlr, shr, loss, lt, profit, said
acq	said, lt, share, dlr, compani, mln, inc, pct, offer, corp
interest	rate, pct, bank, said, cut, interest, market, prime, day, billion
crude	oil, said, price, crude, dlr, barrel, mln, opec, bpd, product
trade	trade, said, japan, billion, would, dlr, year, export, import, japanes
money-supply	billion, dlr, pct, bank, mln, said, money, week, rose, januari
money-fx	said, bank, market, currenc, rate, dollar, exchang, mln, stg, pct
coffee	coffe, said, export, quota, produc, ico, price, brazil, meet, market
sugar	sugar, said, tonn, mln, year, price, trader, ec, export, white
cpi	pct, price, februari, said, inflat, januari, rise, year, consum, rose

Table 2: Top 10 words per Reuters topic based on the Clunky Method.

2.3 Step Three: Latent Dirichlet Allocation (LDA)

On to LDA. I had a bit of a crisis when I looked at this section (this is when I posted on the Discord) but it really wasn't too bad after I investigated it. (Topics vs. categories) vs. (categories vs. topics) aside, you said there was a canned method so I looked for an implementation and found one for the Python library **gensim**. 11 lines of code later I had results. Awesome. It did help that I already had the groundwork from the Clunky implementation, but yeah. That section took like half an hour.

3 Results

. Now it was time for the moment of truth. I ran LDA and held my breath. It came out with Table3.

LDA Topic	**Top 10 Words**
0	pct, year, januari, februari, said, rose, rise, billion, decemb, increas
1	said, compani, would, offer, lt, share, analyst, bid, merger, dlr
2	mln, billion, stg, year, profit, tonn, sugar, trade, china, dlr
3	said, oil, price, dlr, tonn, product, crude, mln, barrel, ga
4	bank, rate, billion, said, pct, market, dollar, mark, currenc, dlr
5	said, trade, would, countri, export, japan, market, meet, govern, offici
6	dlr, said, mln, compani, lt, sale, earn, inc, corp, year
7	lt, said, ltd, pct, compani, stake, unit, corp, firm, group
8	vs, mln, ct, net, loss, shr, dlr, lt, profit, qtr
9	share, said, stock, lt, pct, dlr, offer, compani, common, inc

Table 3: Top 10 words per LDA topic for the Reuters corpus.

I started comparing the two and frankly I'm still dumbfounded. It's almost like I might have a future in NLP or something. As seen in the overall matching in Table4, similarity reached 90% in 'earn,' which is just incredibly gratifying. Overall similarity across the board was 66% which I think is pretty respectable.

Only one topic is below a 50% match. I can think of some things to try (like the aforementioned pruning), and I may delay this to get some of them done, but I am very happy as things stand.

LDA Topic	**Best Matching Topic**	**% Overlap**
0	cpi	70.0%
1	acq	60.0%
2	trade	40.0%
3	crude	80.0%
4	money-fx	70.0%
5	trade	50.0%
6	acq	70.0%
7	acq	50.0%
8	earn	90.0%
9	acq	80.0%
	Overall Match	66.0%

Table 4: Mapping of LDA topics to best matching topics from the Clunky Method with percentage overlap.

4 Discussion

4.1 The Dataset

I wish I had some grandiose realization about the importance of Reuters, but to be honest it was a means to an end. However, with it's ready deployment, categorized documents, and size, it really proved to be ideal for this project. It was complex enough to make this assignment challenging (I did not want to do something with a limited number of categories) and agile enough to be processed quickly. I would like to see about doing some multivariate analysis with category lists rather than singles; perhaps another day. If I get some time I would also really like to do the Congressional dataset. That shows a lot of promise to have some really good content. It would be very interesting to see how stated categories map to actual bill contents. However, I just ran out of cycles.

4.2 Learning

Step one: Really find out when the project is due. I do apologize that I messed this one up quite so badly. It's really not like me. That being said, I guess everyone has a bad day.

I did learn quite a bit this time. Aside from LDA, which frankly blindsided me a bit (I don't remember this being discussed in class but I may have missed a lecture), there was a lot here. Leveraging the stuff from last time was a definite plus; I hate feeling like homework goes into a black hole. However, it's also really nice to see a project build into something bigger. I hope you continue with that.

I also learned a lot in just the implementation of the corpus and probabilities. SGML aside, which I hope to never see again, seeing the facility with which nltk deals with the same (probable) structure really makes for an interesting dichotomy. Knowing the math, as much as it feels like pulling teeth, really does make for a better understanding of how to implement these things too.

And before I forget, it's late, I'm tired and I had ChatGPT write the overlap match section code in its entirety. God, I love LLMs so much.

In case you need it, the GitHub can be found at https://github.com/jjryan111/ARI425