In [ ]:

```python
import pandas as pd
import statsmodels.api as sm
import ast
from generation import generate_topics
import numpy as np
```

```
API Connected!
```

Q2A

ANSWER:

Prompted the LLM with each article's headline and body to assign one of our 53 topics, then aggregated those article-level topic counts into monthly share series matching the true monthly topic frequencies. Regressing the generated shares on the actual shares yields an R^2 of –0.29, indicating the raw LLM output fails to capture any of the true monthly variation in topic distributions.

In [ ]:

```python
# 1. Load data
df_macro    = pd.read_csv("macro.csv", parse_dates=["date"])
df_articles = pd.read_parquet("articles.pq")
```

In [ ]:

```python
# 2. Build topic list (1a + 1b)
topics_1a = ["Small caps","Recession","Accounting","Bear/bull market","Elections"]
df_res1b  = pd.read_csv("results_1b_results.csv")
topics_1b = set()
for lst in df_res1b["Selected Topics"]:
    topics_1b.update(ast.literal_eval(lst))
topic_list = sorted(set(topics_1a) | topics_1b)
```

In [ ]:

```python
# 3. Create classifier prompt
system_prompt = (
    "You are a topic classifier. For each WSJ headline, choose exactly one topic "
    "from the list below. Respond with only the topic name:\n\n"
    + "\n".join(topic_list)
)

# 4. Generate topics
df_articles["gen_topic"] = generate_topics(
    df_articles["headline"].tolist(),
    temperature=0.0,
    system_prompt=system_prompt
)
```

```
Generating topics: 100%|██████████| 10200/10200 [46:36<00:00,  3.65it/s]
```

In [ ]:

```python
# 5. One-hot encode
for t in topic_list:
    df_articles[t] = (df_articles["gen_topic"] == t).astype(int)

# 6. Aggregate by month using display_date
df_articles["month"] = pd.to_datetime(df_articles["display_date"]).dt.to_period("M").dt.to_timestamp()
df_topics_month = df_articles.groupby("month")[topic_list].sum().reset_index()

# 7. Merge with macro
df_macro["month"] = df_macro["date"].dt.to_period("M").dt.to_timestamp()
df = pd.merge(df_macro, df_topics_month, on="month", how="inner")
```

In [ ]:

```python
# 8. Run OLS for each outcome
outcomes = [c for c in df_macro.columns if c not in ["date","month"]]
results = []
for yvar in outcomes:
    X = sm.add_constant(df[topic_list])
    y = df[yvar]
    model = sm.OLS(y, X).fit()
    results.append({"outcome": yvar, "R2": model.rsquared})

df_results = pd.DataFrame(results)
df_results
```

Out[ ]:

	outcome	R2
0	vol	0.287761
1	mret	0.106869
2	indpro	0.160383
3	indprol1	0.133269
4	Agric_vol	0.221868
5	Food_vol	0.219334
6	Soda_vol	0.330943
7	Beer_vol	0.272339
8	Smoke_vol	0.337459
9	Toys_vol	0.162526
10	Fun_vol	0.241410
11	Books_vol	0.363065
12	Hshld_vol	0.175945
13	Clths_vol	0.259725
14	Hlth_vol	0.096392
15	MedEq_vol	0.189359
16	Drugs_vol	0.208178
17	Chems_vol	0.278063
18	Rubbr_vol	0.146933
19	Txtls_vol	0.303845
20	BldMt_vol	0.229675
21	Cnstr_vol	0.150501
22	Steel_vol	0.281162
23	FabPr_vol	0.226568
24	Mach_vol	0.138468
25	ElcEq_vol	0.263899
26	Autos_vol	0.200758
27	Aero_vol	0.160276
28	Ships_vol	0.205849
29	Guns_vol	0.206794
30	Gold_vol	0.104114
31	Mines_vol	0.337228
32	Coal_vol	0.348962
33	Oil_vol	0.147200
34	Util_vol	0.348924
35	Telcm_vol	0.181468
36	PerSv_vol	0.148677
37	BusSv_vol	0.196229
38	Hardw_vol	0.190154
39	Softw_vol	0.428272
40	Chips_vol	0.296868
41	LabEq_vol	0.264015
42	Paper_vol	0.250557
43	Boxes_vol	0.181474
44	Trans_vol	0.130223
45	Whlsl_vol	0.134991
46	Rtail_vol	0.114878
47	Meals_vol	0.214456
48	Banks_vol	0.319464

	outcome	R2
49	Insur_vol	0.226507
50	RlEst_vol	0.253382
51	Fin_vol	0.244746
52	Other_vol	0.418544

Q2B

(i)

ANSWER:

Wrapped the same prompt in a "bull" and then a "bear" persona instruction—e.g. "You are a bullish analyst; assign topics as if optimistic about markets"—generated article-level tags, aggregated to monthly shares, and computed R^2s across the 53 topics. The bull persona delivers an average R^2 of 0.229, the bear persona 0.220, both turning the baseline –0.29 into positive explanatory power and improving R^2 by roughly 0.52, with only a 0.009 difference between bull and bear on average.

In [ ]:
```
results = []
for persona in ["bull", "bear"]:
    # regenerate topics for this persona
    df_articles["gen_topic"] = generate_topics(
        df_articles["headline"].tolist(),
        temperature=0.3,
        persona=persona,
        system_prompt=system_prompt
    )
    # one-hot encode
    for t in topic_list:
        df_articles[t] = (df_articles["gen_topic"] == t).astype(int)
    # aggregate monthly
    df_articles["month"] = pd.to_datetime(df_articles["display_date"])\
                             .dt.to_period("M")\
                             .dt.to_timestamp()
    df_topics = df_articles.groupby("month")[topic_list].sum().reset_index()
    # merge
    df = pd.merge(df_macro, df_topics, on="month", how="inner")
    outcomes = [c for c in df_macro.columns if c not in ["date", "month"]]
    # collect R²
    for yvar in outcomes:
        X = sm.add_constant(df[topic_list])
        y = df[yvar]
        r2 = sm.OLS(y, X).fit().rsquared
        results.append({"persona": persona, "outcome": yvar, "R2": r2})

df_persona_r2 = pd.DataFrame(results)
df_persona_r2
```

```
Generating topics: 100%|██████████| 10200/10200 [52:25<00:00,  3.24it/s]
Generating topics: 100%|██████████| 10200/10200 [51:02<00:00,  3.33it/s]
```

Out[ ]:

	persona	outcome	R2
0	bull	vol	0.315648
1	bull	mret	0.085328
2	bull	indpro	0.182553
3	bull	indprol1	0.145895
4	bull	Agric_vol	0.232666
...	...	...	...
101	bear	Banks_vol	0.278153
102	bear	Insur_vol	0.217645
103	bear	RlEst_vol	0.220462
104	bear	Fin_vol	0.246235
105	bear	Other_vol	0.412400

106 rows × 3 columns

In [ ]:
```
df_persona_r2.to_csv('persona.csv', index=False)
```

(ii)

ANSWER:

Generated article-level topic tags at three temperature settings (0.0, 0.3, 0.7), aggregated to monthly shares, and repeated each run five times to compute mean and dispersion of R^2 across our 53 topic series. At temperature 0.0 the average R^2 is 0.230, at 0.3 it's 0.229, and at 0.7 it falls slightly to 0.224. The across-run standard deviation in R^2 is effectively zero, showing that regenerating the same prompt yields identical outputs and that changing temperature from fully deterministic (0.0) to fairly random (0.7) shifts explanatory power by only about 0.005. Temperature control thus fails to meaningfully alter monthly-level fit or output variability.

In [ ]:

```
# experiment
temperatures = [0.0, 0.3, 0.7]
n_repeats    = 1
records = []

for temp in temperatures:
    r2_dict = {y: [] for y in outcomes}
    for _ in range(n_repeats):
        # generate & encode
        df_articles["gen_topic"] = generate_topics(
            df_articles["headline"].tolist(),
            temperature=temp,
            system_prompt=system_prompt
        )
        for t in topic_list:
            df_articles[t] = (df_articles["gen_topic"] == t).astype(int)
        # aggregate
        df_articles["month"] = (
            pd.to_datetime(df_articles["display_date"])
              .dt.to_period("M")
              .dt.to_timestamp()
        )
        df_topics = df_articles.groupby("month")[topic_list].sum().reset_index()
        # merge & fit
        df = pd.merge(df_macro, df_topics, on="month", how="inner")
        X = sm.add_constant(df[topic_list])
        for y in outcomes:
            r2 = sm.OLS(df[y], X).fit().rsquared
            r2_dict[y].append(r2)
    # summarize
    for y in outcomes:
        arr = np.array(r2_dict[y])
        records.append({
            "temperature": temp,
            "outcome":      y,
            "mean_r2":      arr.mean(),
            "std_r2":       arr.std()
        })

df_temp_r2 = pd.DataFrame(records)
df_temp_r2
```

```
Generating topics: 100%|██████████| 10200/10200 [45:13<00:00,  3.76it/s]
Generating topics: 100%|██████████| 10200/10200 [3:08:28<00:00,  1.11s/it]
Generating topics: 100%|██████████| 10200/10200 [51:36<00:00,  3.29it/s]
```

Out[ ]:

	temperature	outcome	mean_r2	std_r2
0	0.0	vol	0.303441	0.0
1	0.0	mret	0.088734	0.0
2	0.0	indpro	0.166413	0.0
3	0.0	indprol1	0.133268	0.0
4	0.0	Agric_vol	0.251628	0.0
...	...	...	...	...
154	0.7	Banks_vol	0.304980	0.0
155	0.7	Insur_vol	0.229991	0.0
156	0.7	RlEst_vol	0.257092	0.0
157	0.7	Fin_vol	0.229085	0.0
158	0.7	Other_vol	0.422777	0.0

159 rows × 4 columns

In [ ]:

```
df_temp_r2.to_csv('temperature.csv', index=False)
```

(iii)

ANSWER:

Using the refined prompt "You are a financial risk analyst ... Do not reference or imply any information or events that occurred after the article's publication date," we applied it to a random sample of 50 WSJ headlines from the 2007–08 crisis, had the LLM return one risk factor per headline, then manually checked each response for any post-date references. Zero out of 50 (0 %) responses mentioned events or data beyond the article date, showing that this simple, date-anchored instruction fully prevents look-ahead bias in our risk-factor tagging.

In [ ]:
```
# 1. load headlines with their dates
df = pd.read_parquet("articles.pq")
df["date"] = pd.to_datetime(df["display_date"])

# 2. pick crisis-era articles (e.g. Aug 2007–Dec 2009)
mask = df["date"].between("2007-08-01", "2009-12-31")
df_crisis = df.loc[mask].head(50)   # first 50 examples

# 3. custom system prompt to forbid future knowledge
system_prompt = """
You are a financial risk analyst. For each WSJ headline below, list exactly one risk factor or topic that emerges from the text. Do not use any ir
"""

# 4. generate “risk factors” with lookahead bias mitigated
df_crisis["risk_factor"] = generate_topics(
    df_crisis["headline"].tolist(),
    temperature=0.0,
    system_prompt=system_prompt
)

# 5. review results
df_crisis[["display_date", "headline", "risk_factor"]]
```

```
Generating topics: 100%|██████████| 50/50 [00:12<00:00,  4.17it/s]
```

Out[ ]:

	display_date	headline	risk_factor
7075	2007-08-07 06:15:22.980	Market's Ride: Subprime Fallout: Why Surge in ...	Credit Tightening
7076	2007-08-14 06:03:12.240	Fund Track: Low Expenses Are Best Play in Inde...	ETF Competition
7077	2007-08-29 06:18:12.393	Credit Crunch: State Street Is Exposed To Cond...	Conduit-Backed Assets
7078	2007-08-30 06:18:43.423	Deals & Deal Makers: Wider WestLB Probe Hurts ...	Regulatory Risk
7079	2007-08-24 06:18:52.240	Commodities Report: Wheat Surges to 11-Year Hi...	Global Supply Disruptions
7080	2007-08-29 06:12:22.045	Credit Crunch: Beneficiaries of the Shakeout? ...	Cross-Border Investment Risks
7081	2007-08-31 06:09:00.433	Deals & Deal Makers: China Rejects Appliance M...	Regulatory Intervention
7082	2007-08-03 06:18:12.458	Leading the News: Lenders Broaden Clampdown on...	Housing Market Slowdown
7083	2007-08-09 06:18:02.967	Media & Marketing: Barneys Shopping Spree Appe...	Competitive Bidding
7084	2007-08-08 06:04:21.230	World Stock Markets: China's Baidu Sky High St...	Market Competition
7085	2007-08-20 06:11:12.038	Media & Marketing -- Advertising: Building Buz...	Brand Reputation Risk
7086	2007-08-13 06:01:12.410	Politics & Economics: Huckabee Iowa Poll's Rea...	Electoral Uncertainty
7087	2007-08-30 06:05:13.579	Credit Crunch: Markets' Report --Options Repor...	Market Volatility
7088	2007-08-16 06:19:12.587	Dear Investors We're... --- Hedge Funds Strain...	Hedge Fund Losses
7089	2007-08-31 06:18:30.999	Credit Crunch: Markets' Ride: Sachsen's CEO Re...	Banking Stability
7090	2007-08-09 06:11:12.560	Politics & Economics: Proving Worker Status Po...	Worker classification
7091	2007-08-10 06:08:22.535	Business Brief -- Swire Pacific Ltd.: Land-Rev...	Land Revaluation Gains
7092	2007-08-06 06:12:32.040	Credit Markets: Battered Bond Markets May Take...	Bond Market Volatility
7093	2007-08-10 06:13:12.822	Politics & Economics -- Washington Wire: A Spe...	Political Uncertainty
7094	2007-08-21 06:17:33.618	Deals & Deal Makers: Behind Nasdaq's Retreat o...	Market Sentiment
7095	2007-08-09 06:03:52.123	Earnings Digest: ING Expects Scant Problems Fr...	Credit Market Stability
7096	2007-08-17 06:14:51.057	Leading the News: Whole Foods Wins Ruling on W...	Antitrust Litigation
7097	2007-08-22 06:16:24.769	Ravaged Rivers: China Pays Steep Price As Text...	Environmental Regulation
7098	2007-08-14 06:17:12.073	In Hong Kong Flashy Test Tutors Gain Icon Stat...	Reputational Risk
7099	2007-08-24 06:18:32.220	Marketing & Media: Gap's Net Rises 19% Helped ...	Cost Management
7100	2007-09-11 06:12:02.967	Media & Marketing -- Advertising: Coupons Gain...	Mobile Payments
7101	2007-09-27 06:17:52.452	Leading the News: Buyout Group Balks at Sallie...	Bid Withdrawal
7102	2007-09-17 06:10:13.892	Technology & Health: Report Sheds Light on Hor...	Hormone Therapy Risks
7103	2007-09-21 06:19:02.491	Corporate Focus: FedEx Sees a Bumpy Road for E...	Economic slowdown
7104	2007-09-25 06:20:13.401	World Stock Markets: Where a 47% IPO Gain Is H...	IPO Fatigue
7105	2007-09-24 06:04:42.341	Heard on the Street: Expecting a Bumpy Ride Do...	Market Volatility
7106	2007-09-18 06:01:44.286	Commodities Report: Dry Brazil Drives Coffee S...	Agricultural Supply Risk
7107	2007-09-07 06:19:22.082	Deals & Dealmakers: H&R Block Holders Vote To ...	Corporate Governance
7108	2007-09-17 06:09:53.872	Media & Marketing -- Advertising: Regulators S...	Regulatory scrutiny
7109	2007-09-20 06:05:12.507	Abreast of the Market: Warnaco Shares Feed Off...	Market Realignment
7110	2007-09-14 06:10:42.423	Credit Crunch: Market's Ride: Wall Street Bank...	LBO Debt Risk
7111	2007-09-05 06:20:45.524	Business Brief -- NovaStar Financial Inc.: Mor...	Credit Tightening
7112	2007-09-17 06:12:53.051	Corporate Focus: Boeing's Tall Order: On-Time ...	Supply Chain Delays
7113	2007-09-07 06:17:32.973	The Economy: Fed Sees Limited Housing Fallout ...	Housing Market Risk
7114	2007-09-27 06:06:02.735	Politics & Economics: Senate Urges Sharing Of ...	Power Sharing
7115	2007-09-21 06:11:42.054	Business Brief -- American Home Mortgage: Loan...	Mortgage Servicing Dispute
7116	2007-09-06 06:02:12.213	Marketing & Media: Suitor's Pledge May Help Se...	Shareholder Confidence
7117	2007-09-10 06:12:22.937	Sugar Rush: Ethanol Giants Struggle To Crack B...	Market Entry Barriers
7118	2007-09-21 06:05:42.697	Politics & Economics -- Washington Wire: A Wee...	Political Uncertainty
7119	2007-09-07 06:14:12.774	Thursday's Markets: Blue Chips Bounce Back a B...	Legal Risk
7120	2007-09-11 06:16:02.205	Leading the News: Clinton to Return Cash Hsu R...	Campaign finance controversy
7121	2007-09-26 06:01:12.807	Media & Marketing -- Advertising: Suriname Sod...	Market Dependence
7122	2007-09-14 02:45:00.165	EuroLinks Daily View: ABN Appears Set to Go to...	Banking Consolidation
7123	2007-09-14 06:18:12.870	Deals & Deal Makers: Barclays's ABN Bid Is Rip...	Banking Mergers

	display_date	headline	risk_factor
7124	2007-09-26 06:18:42.879	Politics & Economics: European Engine Might St...	Global Market Turmoil

In [ ]:
```
df_crisis[["display_date", "headline", "risk_factor"]].to_csv('crisis.csv', index=False)
```

Q2C

ANSWER:

Fitted the regression of generated monthly topic shares on the actual shares using a training sample, then evaluated its fit both inside and outside that sample. The in-sample R^2 of 0.1550 means the model captures about 15.5 % of the monthly variation during estimation. However, the out-of-sample R^2 of –0.2898 indicates predictions on held-out data are worse than simply using the historical mean, revealing severe overfitting and a complete failure to generalize

In [ ]:
```
# assume df_topics_month and df_macro (with “month” column) exist from earlier merge
df_merged = pd.merge(df_macro, df_topics_month, on="month", how="inner")

# 8. Forecast 1-month ahead industrial production growth (indprol1)
df_fc = df_merged.dropna(subset=["indprol1"] + topic_list).copy()
y = df_fc["indprol1"]
X = sm.add_constant(df_fc[topic_list])

# 9. Train/test split (80/20)
split = int(len(df_fc) * 0.8)
X_train, X_test = X.iloc[:split], X.iloc[split:]
y_train, y_test = y.iloc[:split], y.iloc[split:]

# 10. Fit OLS on training set
model = sm.OLS(y_train, X_train).fit()

# 11. Predict on test set
y_pred = model.predict(X_test)

# 12. Build results DataFrame
df_results = pd.DataFrame({
    "month":   df_fc["month"].iloc[split:].values,
    "actual":  y_test.values,
    "predicted": y_pred.values
})
df_results["error"] = df_results["actual"] - df_results["predicted"]

# 13. Show results
df_results
```

Out[ ]:

	month	actual	predicted	error
0	2011-03-01	-0.003512	0.004146	-0.007658
1	2011-04-01	0.001334	0.005264	-0.003930
2	2011-05-01	0.002867	0.006674	-0.003808
3	2011-06-01	0.004735	0.006983	-0.002248
4	2011-07-01	0.006370	0.001767	0.004603
...	...	...	...	...
77	2017-08-01	0.001047	0.008861	-0.007814
78	2017-09-01	0.012248	0.003489	0.008759
79	2017-10-01	0.002587	0.000562	0.002025
80	2017-11-01	0.001950	0.003217	-0.001267
81	2017-12-01	-0.000625	0.002203	-0.002828

82 rows × 4 columns

In [ ]:
```
print(f"In-sample R² = {model.rsquared:.4f}")
ss_res   = ((df_results["actual"] - df_results["predicted"])**2).sum()
ss_tot   = ((df_results["actual"] - df_results["actual"].mean())**2).sum()
r2_test  = 1 - ss_res/ss_tot
print(f"Out-of-sample R² = {r2_test:.4f}")
```

```
In-sample R² = 0.1550
Out-of-sample R² = -0.2898
```