CS Job-Market Sentiment Across Market Shifts

A Sentiment Analysis and Topic-Trend Study of Reddit Job-Market Discussion

Anahita Memar
Nifemi Lawal

EECS 767: Information Retrieval
Professor Bo Luo

Department of Electrical Engineering and Computer Science
The University of Kansas

Spring 2026

Contents

Abstract

This project studies how online discussion about the computer science job market changed across the pre-COVID, COVID, and post-COVID periods. We built an end-to-end sentiment analysis pipeline using Sentiment140 as labeled training data and Reddit submissions as the application corpus. Our system trains sentiment classifiers, processes monthly mini-subreddit Reddit submission files, preprocesses post text, predicts sentiment, assigns posts to time periods, tags posts into topic buckets, tests project hypotheses, and generates final tables and plots.

The final analysis covers approximately 1.54 million Reddit posts across 108 months from four subreddits: `cscareerquestions`, `csMajors`, `recruitinghell`, and `jobs`. About 807,000 posts received at least one topic tag. The overall positive sentiment rate was 48.26%. The main finding is a steady decline in positive sentiment from the pre-COVID period to the post-COVID period. Positive sentiment was 52.03% before COVID, 49.00% during COVID, and 46.51% after COVID. Recruiting pipeline discussion dominated the topic distribution, while layoffs, compensation, and work mode were among the most negative areas. AI and LLM discussion increased sharply in recent years, showing that generative AI became a growing part of job-market anxiety and discussion.

1. Introduction

The computer science job market changed sharply after 2020. Remote work became more common, hiring patterns shifted, large technology companies slowed hiring, and major layoff waves became a visible concern for students and early-career developers. More recently, generative AI tools such as ChatGPT, Copilot, and other large language models added new uncertainty about skills, interviews, automation, and the future of software engineering work.

A large amount of discussion about these changes appears in online communities. Reddit is especially useful for this project because career subreddits contain long-running conversations about internships, new graduate roles, interviews, layoffs, salaries, remote work, and company-specific hiring patterns. These posts are unstructured text, which makes them a natural fit for information retrieval and text mining methods.

We studied whether sentiment in CS job-market discussion changed across three market periods:

- **Pre-COVID:** 2017 to 2019
- **COVID:** 2020 to 2021
- **Post-COVID and layoff era:** 2022 to 2025

We also studied which topics were most common and which topics were most negative. The main topics are recruiting pipeline, layoffs and market contraction, compensation, work mode, companies, and AI/LLMs.

2. Research Questions and Hypotheses

We built the project around four testable hypotheses.

H1. Recruiting sentiment becomes more negative in the post-COVID era compared to the pre-COVID era.

H2. Posts mentioning layoffs or hiring freezes increase after 2022 and co-occur with lower sentiment.

H3. User-level sentiment differs by topic, producing measurable cross-topic correlations.

H4. Increases in AI-related discussion are associated with shifts in overall job-market sentiment over time.

These hypotheses connect directly to our project goal. H1 tests whether the broad recruiting mood became worse. H2 tests whether layoff-related discussion is both more common and more negative. H3 checks whether the same users show different sentiment patterns across different job-market topics. H4 studies whether AI discussion grew during the same period as sentiment shifted.

3. Data Sources

3.1. Sentiment140 Training Data

The labeled training data comes from Sentiment140, a dataset of 1.6 million tweets with binary sentiment labels. The raw labels are 0 for negative and 4 for positive. In our code, label 4 is remapped to 1, giving a clean binary classification task:

$$0 = \text{negative}, \qquad 1 = \text{positive}$$

Sentiment140 is useful because it provides a large labeled corpus for short-text sentiment classification. The main limitation is platform mismatch. Twitter text and Reddit text have different writing styles, lengths, slang, and context. This limitation is discussed later, but the dataset still gives a practical supervised signal for building a reusable sentiment classifier.

3.2. Reddit Application Corpus

The application corpus consists of Reddit submissions from four job-market-related subreddits:

- `cscareerquestions`
- `csMajors`
- `recruitinghell`
- `jobs`

We selected these subreddits because they capture different parts of the job-market conversation. `cscareerquestions` focuses on CS recruiting and software engineering career advice. `csMajors`

captures student and early-career concerns. `recruitinghell` captures frustration with hiring processes. `jobs` provides a broader employment discussion baseline.

The final dataset contained approximately 1.54 million posts across 108 months. The Reddit application data came from monthly mini-subreddit submission files for the four selected subreddits. About 807,000 posts were tagged with at least one project topic.

3.3. External Layoff Context

Our project proposal included the Layoffs.fyi-based Kaggle layoff dataset as an external grounding signal. The goal was to compare job-market sentiment with layoff activity over time. In the final implementation, we analyzed layoffs mainly through Reddit topic frequency and topic sentiment. Full direct alignment with external monthly layoff counts remains a future improvement.

4. System Overview

We implemented a reproducible end-to-end pipeline. The main workflow is:

1. Load and clean Sentiment140.
2. Train sentiment classifiers.
3. Save the TF-IDF vectorizer and trained models.
4. Process monthly mini-subreddit Reddit files for the four target subreddits.
5. Select important Reddit fields.
6. Build one final text field from title and body.
7. Clean Reddit text using the same cleaning logic as training.
8. Predict sentiment on Reddit posts.
9. Convert timestamps into month and era labels.
10. Tag posts into topic buckets using keyword rules.
11. Compare sentiment by topic.
12. Measure topic trends over time.
13. Run hypothesis tests.
14. Generate final tables, plots, and written summaries.

The codebase is organized into four main source areas:

- `src/utils/`: shared preprocessing and topic-bucket definitions
- `src/training/`: supervised sentiment model training
- `src/pipeline/`: Reddit ingestion, cleaning, and prediction
- `src/analysis/`: time grouping, topic tagging, and trend analysis
- `src/reporting/`: hypothesis testing and final report outputs

This structure keeps the workflow modular. Each script owns one stage of the pipeline and writes an output file that becomes the input for the next stage.

5. Text Preprocessing

The shared preprocessing module is `src/utils/preprocessing.py`. It contains the cleaning logic used for both Sentiment140 and Reddit text. The cleaning function performs the following steps:

- Lowercase all text.
- Remove URLs.
- Remove user mentions.
- Remove non-letter characters.
- Normalize repeated whitespace.
- Drop rows that become empty after cleaning.

This design matters because the sentiment model is trained on cleaned text, so Reddit inference text must be cleaned in the same way. Otherwise, the TF-IDF vectorizer would see a different input distribution at prediction time.

The preprocessing is intentionally simple. It avoids heavy linguistic processing such as stemming or lemmatization. This keeps the pipeline easier to inspect, easier to reproduce, and more consistent across tweets and Reddit posts.

6. Sentiment Model Training

The training script is `src/training/train_sentiment.py`. It loads the Sentiment140 CSV, applies preprocessing, splits the data into training and testing sets, trains two classifiers, evaluates them, and saves the model artifacts.

6.1. Feature Representation

The project uses TF-IDF features. TF-IDF represents each document as a weighted vector of terms. Terms that appear often in a document receive higher weight, while terms that appear in many documents receive lower weight. This matches the vector space model ideas from information retrieval.

The vectorizer uses:

- maximum features: 50,000
- sublinear term frequency
- unigram text features

Using 50,000 features keeps the feature space large enough to capture useful vocabulary while still staying practical for a course project.

6.2. Models Compared

Two supervised classifiers were trained:

- Logistic Regression
- Multinomial Naive Bayes

The data split was 80% training and 20% testing, with stratification so that the positive and negative label balance stayed consistent across the split.

Table 1: Sentiment140 model performance.

Model	**Accuracy**	**F1 Score**
Logistic Regression	0.7983	0.8010
Naive Bayes	0.7767	0.7745

Logistic Regression performed better on both accuracy and F1 score, so we selected it as the main sentiment model for Reddit prediction. The trained artifacts are saved as:

- `src/models/tfidf_vectorizer.joblib`
- `src/models/logistic_regression.joblib`
- `src/models/naive_bayes.joblib`

Only the TF-IDF vectorizer and Logistic Regression model are required for the final Reddit prediction step.

7. Reddit Processing Pipeline

7.1. Processing Monthly Subreddit Files

The Reddit filtering script is `src/pipeline/filter_reddit_submissions.py`. It reads the monthly mini-subreddit Reddit files line by line and supports compressed `.zst` files through the `zstandard` library. Each line is parsed as a JSON record.

The script keeps only records that satisfy the project selection rules:

- The subreddit is one of the four target subreddits.
- The timestamp falls between 2017 and 2025.

The script also standardizes the Reddit records into one downstream schema. Important output fields include subreddit, author, timestamp, title, body, score, comments, permalink, and URL.

7.2. Selecting Fields

The next script, `src/pipeline/select_reddit_fields.py`, keeps only the fields needed downstream:

- `id`
- `subreddit`
- `author`
- `created_utc`
- `title`
- `body`

This step reduces unnecessary metadata and makes later scripts easier to reason about.

7.3. Building Final Text

The script `src/pipeline/build_reddit_final_text.py` combines title and body into one field called `final_text`. The rules are:

- If both title and body exist, combine them.
- If only title exists, use the title.
- If only body exists, use the body.

- If both are empty, drop the row.

This creates a single text field for both sentiment prediction and topic tagging.

7.4. Cleaning Reddit Text

The script `src/pipeline/preprocess_reddit_text.py` applies the shared `clean_text()` function to `final_text`. The output column is `cleaned_text`. Empty cleaned rows are removed.

This is one of the more important design choices in the project because the same preprocessing function is used in both model training and Reddit prediction.

7.5. Predicting Reddit Sentiment

The script `src/pipeline/predict_reddit_sentiment.py` loads:

- `tfidf_vectorizer.joblib`
- `logistic_regression.joblib`

It transforms `cleaned_text` into TF-IDF features, predicts binary sentiment labels, and writes the following sentiment columns:

- `predicted_label`
- `predicted_sentiment`
- `prob_negative`
- `prob_positive`

The output file becomes the main sentiment-labeled Reddit dataset.

8. Time Grouping

The script `src/analysis/group_reddit_by_time.py` converts `created_utc` timestamps into readable UTC datetimes and creates these columns:

- `created_datetime_utc`
- `year`
- `month`

- `year_month`
- `time_period`

The `time_period` mapping is:

Table 2: Time period mapping.

Period	Years
Pre-COVID	2017 to 2019
COVID	2020 to 2021
Post-COVID	2022 and later

These fields make it possible to compare sentiment at both the month level and the era level.

9. Topic Tagging

The topic-bucket definitions are stored in `src/utils/topic_buckets.py`. Each bucket has a description and a keyword list. The six topic buckets are:

Table 3: Topic buckets used for downstream analysis.

Topic Bucket	Meaning
`recruiting_pipeline`	Recruiting, applications, interviews, offers, rejections, online assessments, LeetCode, internships, and new graduate roles.
`layoffs_market`	Layoffs, hiring freezes, severance, recession, headcount reduction, and job-market contraction.
`compensation`	Salary, total compensation, equity, RSUs, negotiation, bonuses, and pay.
`work_mode`	Remote work, work from home, hybrid work, return to office, and on-site work.
`companies`	Specific companies such as Amazon, Google, Meta, Microsoft, Apple, OpenAI, Nvidia, Netflix, Tesla, and others.
`ai_llm`	ChatGPT, GPT, LLMs, AI, Copilot, generative AI, machine learning, automation, and prompt engineering.

The tagging script is `src/analysis/tag_reddit_topics.py`. It cleans each keyword using the same cleaning function and builds regular expressions that match whole words or whole phrases. This avoids many false matches that would happen with simple substring matching.

A post can match multiple topics. For example, a post about "new grad offers after layoffs" can match both `recruiting_pipeline` and `layoffs_market`. This is the correct behavior because real job-market posts often discuss multiple themes.

The tagging step adds:

- one binary column for each topic
- one matched-keyword column for each topic
- `topic_list`
- `num_topics`
- `has_any_topic`

10. Analysis and Reporting Scripts

10.1. Sentiment by Topic

The script `src/analysis/compare_sentiment_by_topic.py` computes positive and negative sentiment rates for each topic. It also creates ranked topic tables from most positive to least positive and from most negative to least negative.

This script outputs:

- `topic_sentiment_summary.csv`
- `topic_sentiment_ranked_positive.csv`
- `topic_sentiment_ranked_negative.csv`
- `topic_post_long_format.csv`

10.2. Topic Trends Over Time

The script `src/analysis/measure_topic_trends_over_time.py` computes monthly topic counts and topic shares. It also builds wide-format tables for plotting and long-format tables for easier analysis.

This step is important because a topic can become more important over time even if its sentiment does not change. For example, AI-related posts were relatively small before the rise of ChatGPT, but later became much more common.

10.3. Hypothesis Testing

The script `src/reporting/test_project_hypotheses.py` runs structured tests for H1 through H4.

H1 compares recruiting sentiment in the pre-COVID and post-COVID periods.

H2 compares layoff-topic frequency after 2022 against earlier periods and compares sentiment for layoff-related posts against non-layoff posts.

H3 builds user-topic sentiment profiles for users with at least three posts and computes topic-level correlations.

H4 computes monthly AI-topic share and correlates it with overall monthly sentiment.

The script uses a two-proportion z-test implemented directly in Python for the proportion comparisons. It also computes correlations using pandas.

10.4. Final Report Outputs

The script `src/reporting/create_final_report_outputs.py` creates final CSV tables, plots, and written summaries. This script produces report-ready artifacts such as:

- overall summary tables
- sentiment by time period
- sentiment by subreddit
- sentiment by topic
- monthly sentiment tables
- monthly topic trend tables
- hypothesis overview tables
- final plots
- final written summaries

This made the final analysis reproducible instead of a one-time set of manual calculations.

11. Results

11.1. Dataset Overview

The final analysis used approximately 1.54 million Reddit posts. The dataset covered 108 months and four subreddits. About 807,000 posts were tagged with at least one topic bucket.

Table 4: Dataset overview.

Metric	Value
Posts analyzed	1.54 million
Months observed	108
Subreddits covered	4
Topic-tagged posts	807,000
Overall positive sentiment rate	48.26%

11.2. Monthly Sentiment Trend

The monthly sentiment plot shows the overall positive sentiment rate from 2017 through 2025. Sentiment was generally higher before and during the early COVID period, then dropped sharply around late 2021. After 2023, the rate stayed mostly below the earlier pre-COVID levels.

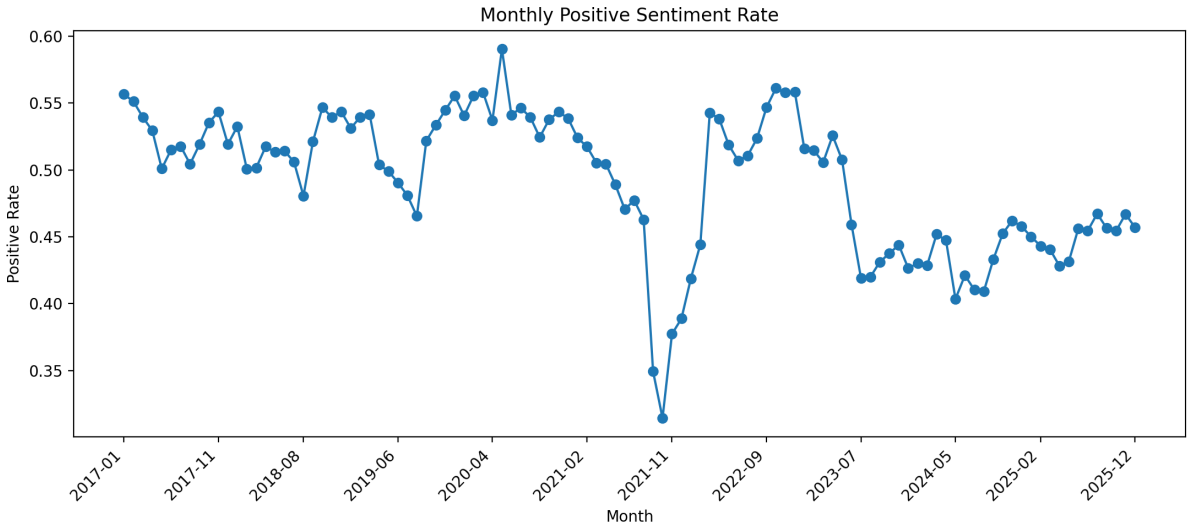

Figure 1: Monthly positive sentiment rate from 2017 to 2025.

11.3. Era-Level Sentiment

Era-level sentiment supports the main direction of H1. Positive sentiment declined from the pre-COVID period to the post-COVID period.

Table 5: Positive sentiment by time period.

Time Period	Posts	Positive Rate
Pre-COVID	337,918	52.03%
COVID	332,373	49.00%
Post-COVID	865,522	46.51%

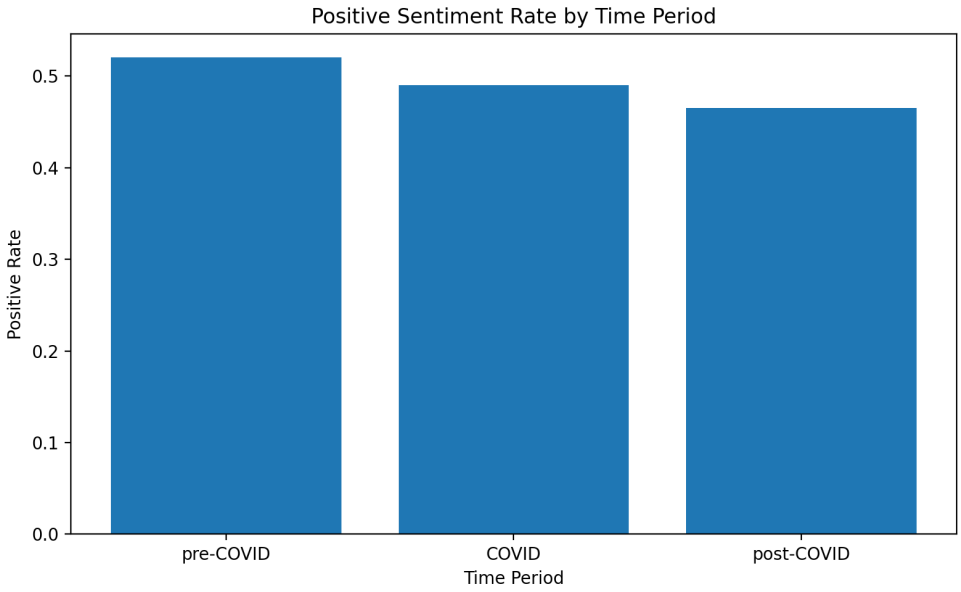

Figure 2: Positive sentiment rate by time period.

11.4. Subreddit-Level Sentiment

Sentiment differed strongly by subreddit. The CS-focused communities were more positive, while broader job-market and recruiting-frustration communities were more negative.

Table 6: Positive sentiment by subreddit.

Subreddit	**Positive Rate**
`csmajors`	59.07%
`cscareerquestions`	56.48%
`jobs`	42.14%
`recruitinghell`	39.55%

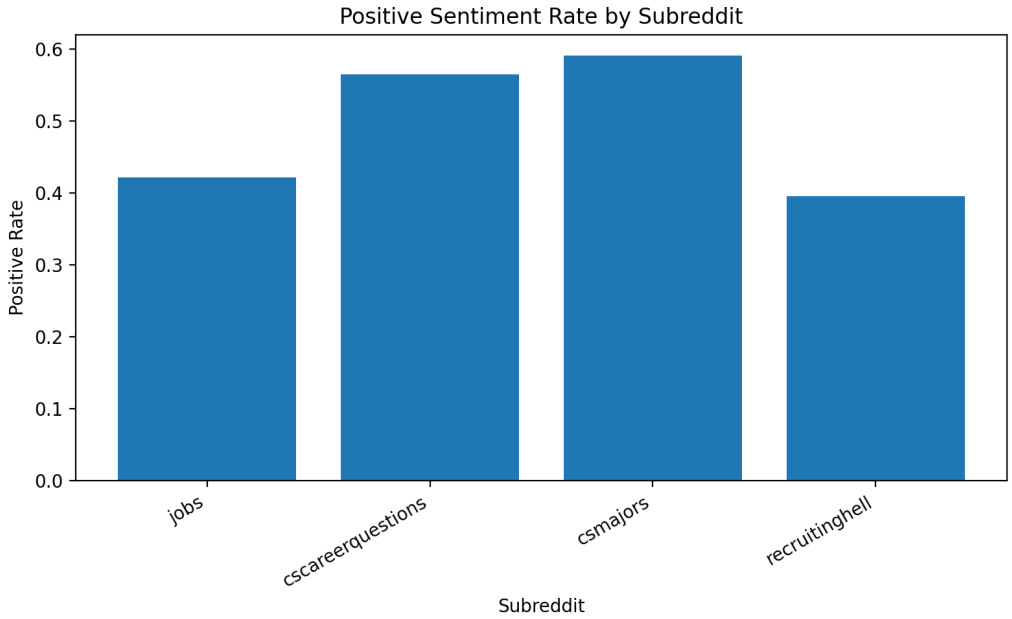

Figure 3: Positive sentiment rate by subreddit.

11.5. Topic Trends Over Time

Recruiting pipeline was the dominant topic across the whole period. Compensation was consistently the second largest topic. Work mode increased after remote and hybrid work became more common. Layoffs and market discussion increased sharply after 2022. AI and LLM discussion grew rapidly in the later years, especially after generative AI became widely used.

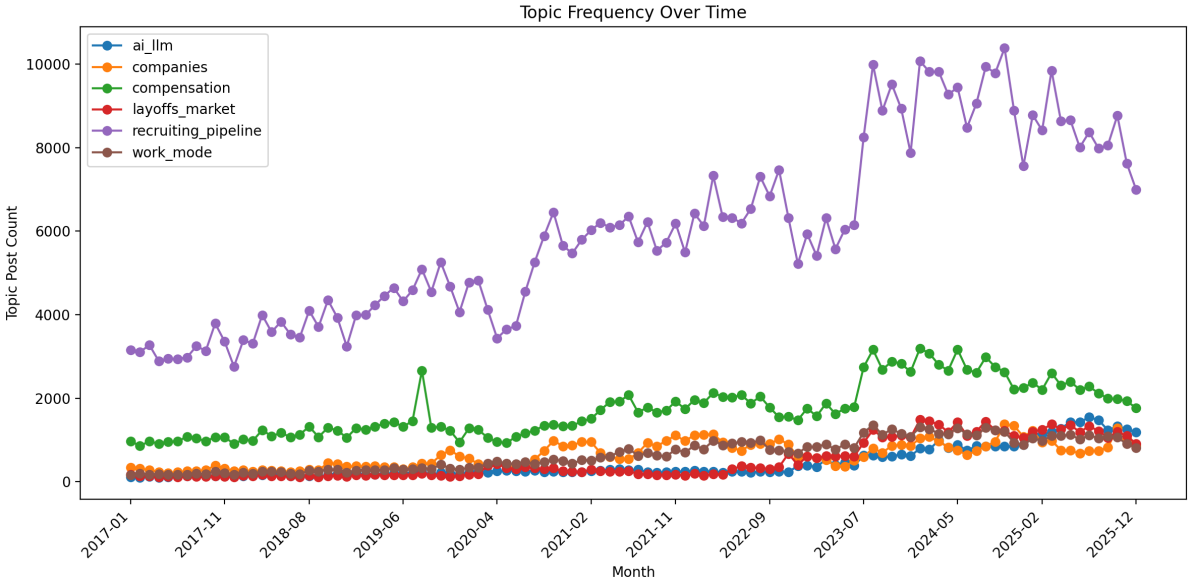

Figure 4: Monthly topic frequency over time.

The combined trend plot is useful for showing scale, but recruiting dominates the graph. The small-multiples version separates the six topic buckets so that lower-volume topics are easier to compare.

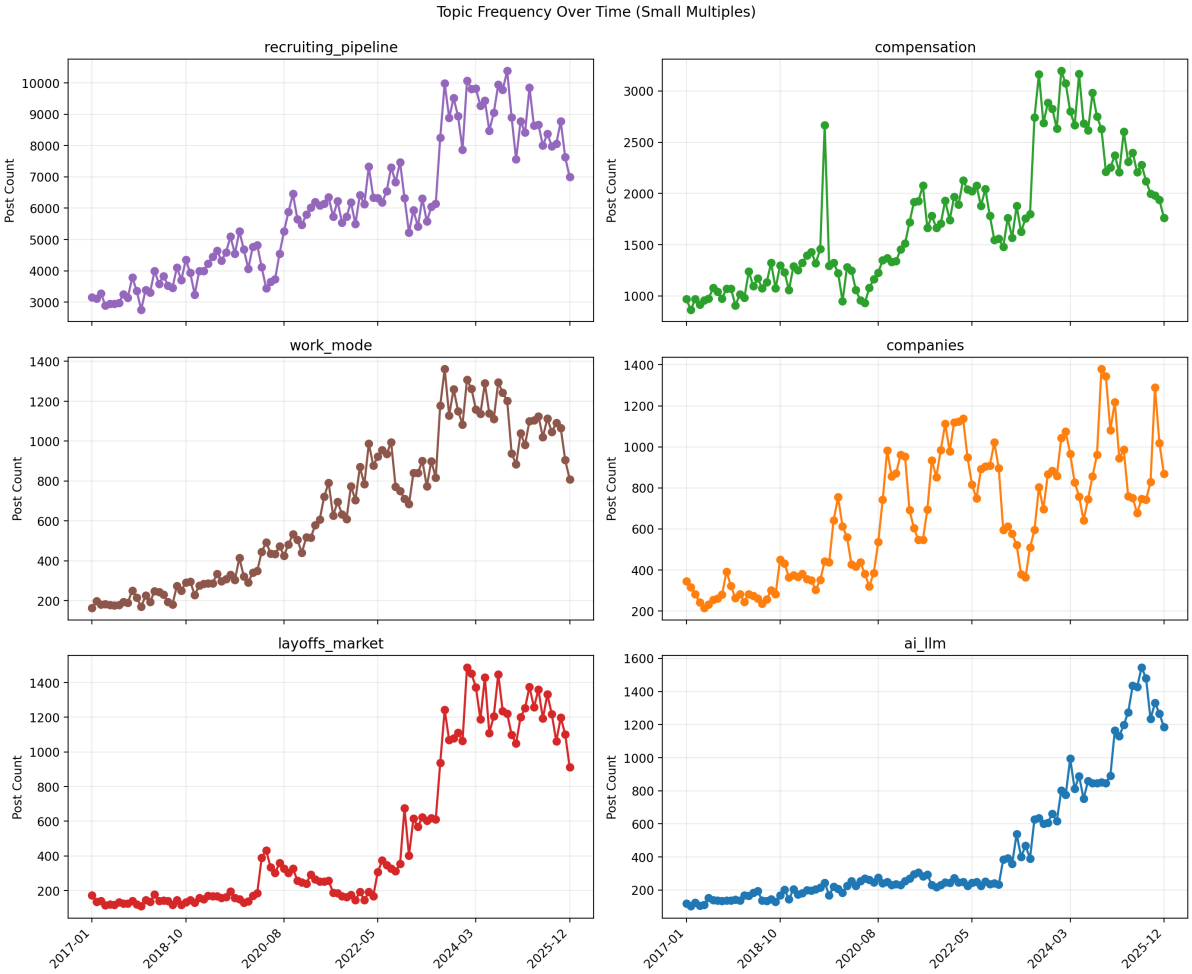

Figure 5: Monthly topic frequency over time, shown as small multiples.

The indexed trend plot normalizes each topic to its first nonzero month. This makes it easier to compare growth patterns instead of raw volume. The AI/LLM line shows the strongest late-period growth.

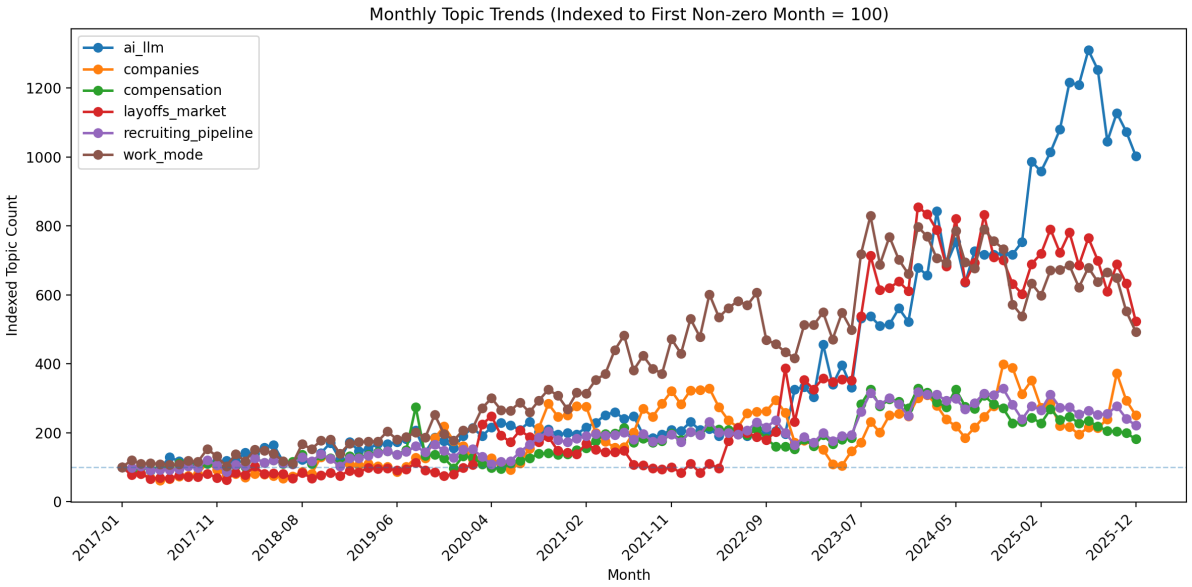

Figure 6: Monthly topic trends indexed to each topic's first nonzero month.

11.6. Topic-Level Sentiment

Topic-level sentiment shows that company-related posts were the most positive, followed by AI/LLM and recruiting pipeline. Layoffs and market discussion was the most negative topic, followed by compensation and work mode.

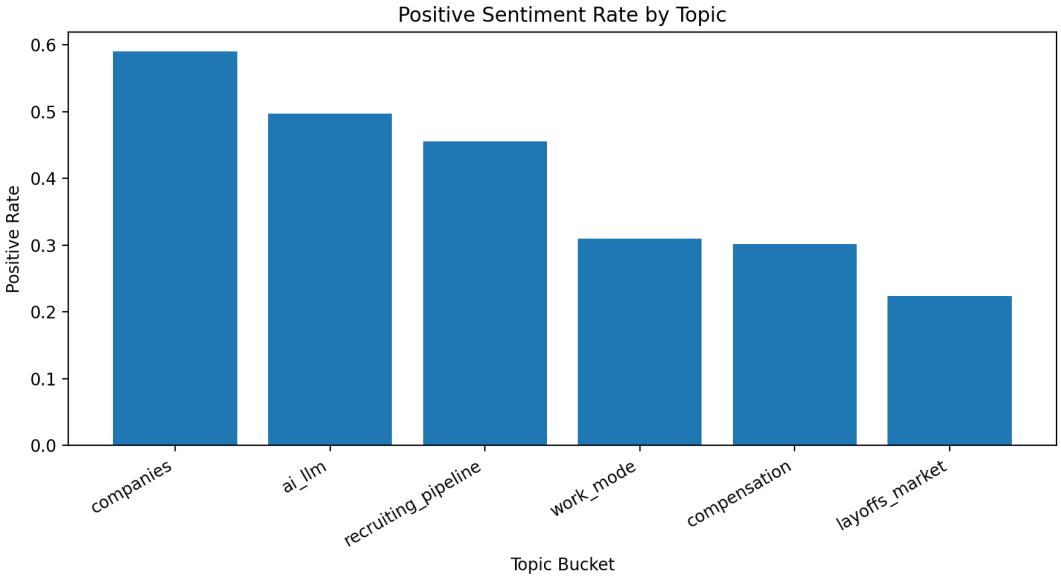

Figure 7: Positive sentiment rate by topic bucket.

Table 7: Approximate positive sentiment ranking by topic.

Topic	Positive Rate, Approx.
Companies	59%
AI/LLM	50%
Recruiting pipeline	46%
Work mode	31%
Compensation	30%
Layoffs and market	22%

11.7. Topic Frequency

Recruiting pipeline was by far the most common topic. Compensation was the second most common. Work mode, companies, layoffs and market, and AI/LLM appeared less often, but they were still important for explaining specific shifts in sentiment.

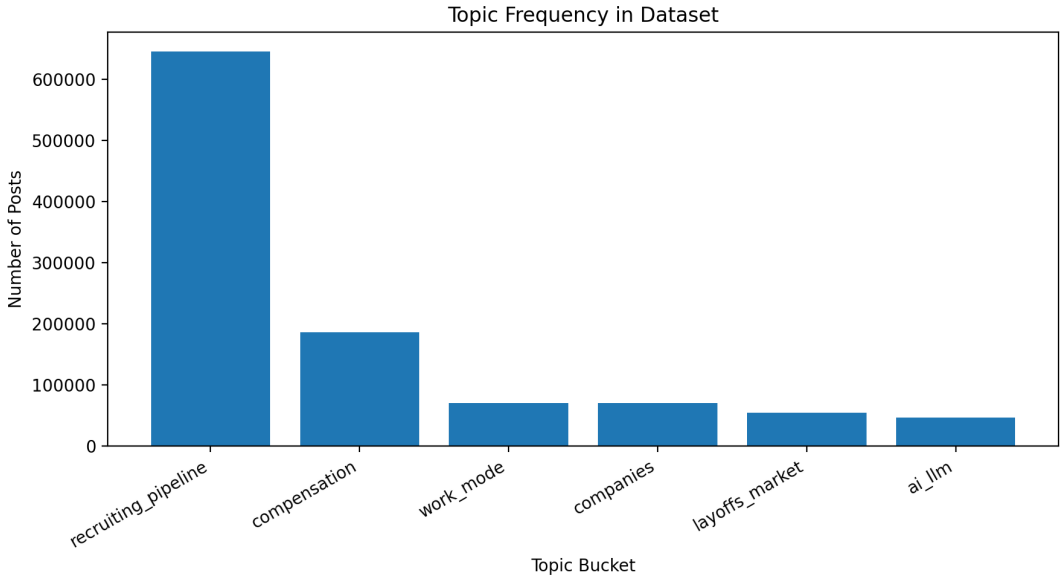

Figure 8: Topic frequency in the dataset.

12. Hypothesis Evaluation

12.1. H1: Recruiting Sentiment Became More Negative Post-COVID

H1 is directionally supported. Overall era-level sentiment declined from 52.03% positive in the pre-COVID period to 46.51% positive in the post-COVID period. We also found that recruiting pipeline was the dominant topic, making it central to the overall mood of the dataset.

The evidence suggests that the CS recruiting conversation became more negative after COVID,

especially in the 2022 and later period.

12.2. H2: Layoff Discussion Increased and Co-Occurred With Lower Sentiment

H2 is supported. Layoff and market-related discussion rose after 2022 and was one of the most negative topic groups. The topic-level sentiment plot showed layoffs and market as the lowest-positive topic, with an approximate positive sentiment rate near 22%.

This means layoff-related posts were not only more visible after the market shifted, but also strongly negative in sentiment.

12.3. H3: User-Level Sentiment Differs by Topic

H3 is partially supported. Our code implements a user-level topic sentiment analysis by filtering to users with at least three posts, calculating average sentiment by topic per user, and computing a topic correlation matrix.

This finding is less central than the time-based findings because user-level overlap across topics can be limited. A user must post enough times and across enough topics to produce useful correlations. The analysis is still valuable because it shows that the pipeline can support user-level behavioral comparisons.

12.4. H4: AI Discussion Is Associated With Sentiment Shifts

H4 is supported as a trend association. AI and LLM discussion increased sharply in recent years. This rise happened during the same broader period where overall positive sentiment was lower than the pre-COVID period.

The project does not prove that AI caused sentiment to decline. It shows that AI became a growing part of the discussion during a period of job-market uncertainty. This distinction matters because correlation does not imply causation.

Table 8: Hypothesis support summary.

Hyp.	**Status**	**Reason**
H1	Directionally supported	Positive sentiment declined from pre-COVID to post-COVID.
H2	Supported	Layoff-topic discussion rose after 2022 and was strongly negative.
H3	Partially supported	User-topic correlation analysis is implemented, but it is less impactful than the time-based results.
H4	Supported as association	AI/LLM discussion surged in recent years during the same broad period of lower sentiment.

13. Discussion

The strongest conclusion from our analysis is that the online CS job-market conversation became more negative over time. The decline is visible at the era level and is reinforced by topic-level results. The post-COVID period had the lowest positive sentiment rate, and the topics most connected to market stress, especially layoffs and compensation, were among the most negative.

The project also shows the importance of topic-aware sentiment analysis. Looking only at the overall positive rate hides important differences between types of discussion. Company posts were relatively positive, while layoffs and compensation were much more negative. This means that job-market sentiment is not one single mood. It is a mixture of different concerns, and each topic contributes differently.

Subreddit differences also matter. A dataset with more `recruitinghell` posts will naturally look more negative than a dataset with more `csMajors` posts. For that reason, subreddit-level reporting is necessary for interpreting the results fairly.

The AI/LLM trend is also important. AI was not just another topic. It grew sharply in the later part of the dataset. This suggests that generative AI became part of how students and developers discuss career uncertainty, job preparation, and future skill demands.

14. Limitations

14.1. Platform Mismatch

The sentiment model was trained on Sentiment140, which contains tweets. It was then applied to Reddit posts. This creates a platform mismatch. Reddit posts are often longer, more detailed, and more context-dependent than tweets. Sarcasm, nuanced complaints, and mixed emotions may not be classified perfectly.

14.2. Binary Sentiment

The model predicts only negative or positive sentiment. It does not include neutral sentiment. This is a limitation because many job-market posts are informational rather than clearly positive or negative. A three-class or continuous sentiment model could capture more nuance.

14.3. Keyword-Based Topic Tagging

Topic tagging uses keyword rules. This makes the system transparent and easy to debug, but it can miss posts that discuss a topic without using the expected keywords. It can also misclassify posts when a keyword appears in an unexpected context.

14.4. Correlation Does Not Prove Causation

The project finds associations between time periods, topics, and sentiment. It does not prove that layoffs, AI, or remote work directly caused sentiment changes. External economic data and stronger causal methods would be needed for that.

14.5. Reddit Is Not the Entire Market

Reddit discussion reflects active online communities, not the whole CS job market. People who post on Reddit may be more anxious, more frustrated, or more job-search-focused than the average CS student or developer.

15. Future Work

Several improvements could make the project stronger.

First, the model could be improved with Reddit-native labeled data or a transformer-based sentiment classifier. This would reduce the mismatch between Twitter training data and Reddit application data.

Second, topic detection could move from keyword rules to semantic classification. A semantic model could detect layoffs, AI, compensation, or work-mode discussion even when exact keywords are not present.

Third, Reddit comments could be added. Submissions show what topics users start, but comments show how communities respond. Comment-level analysis would give a richer picture of sentiment and discussion dynamics.

Fourth, the external layoff dataset could be directly merged into the monthly trend analysis. This would allow clearer comparison between actual layoff counts and Reddit sentiment.

Fifth, subreddit-specific models or controls could be added. This would help separate true market mood changes from changes caused by subreddit composition.

16. Conclusion

This project built a complete sentiment analysis and topic-trend pipeline for studying CS job-market discussion across major market shifts. The system trains a sentiment model on Sentiment140, applies it to Reddit submissions, groups posts by time period, tags posts into meaningful job-market topics, tests project hypotheses, and generates final plots and tables.

The main result is that CS job-market sentiment became more negative from the pre-COVID period to the post-COVID period. Positive sentiment fell from 52.03% before COVID to 46.51% after COVID. Recruiting pipeline was the dominant topic, while layoffs, compensation, and work mode were among the most negative topics. AI and LLM discussion grew sharply in recent years,

showing that generative AI became an important part of job-market discussion.

The project demonstrates how information retrieval and text mining methods can turn large-scale unstructured social media text into structured evidence about market sentiment. While the results should be interpreted with caution, the pipeline is reusable, transparent, and strong enough to support further analysis with larger datasets or improved sentiment models.

A. Appendix A: Code Structure

Table 9: Main project files and purposes.

File	Purpose
`src/utils/preprocessing.py`	Shared text cleaning functions and Sentiment140 loading.
`src/utils/topic_buckets.py`	Defines the topic buckets and keyword lists used for tagging Reddit posts.
`src/training/train_sentiment.py`	Trains Logistic Regression and Naive Bayes on Sentiment140, compares model performance, and saves the TF-IDF vectorizer and trained models.
`src/pipeline/filter_reddit_submissions.py`	Reads monthly mini-subreddit Reddit files, applies the project filters, and writes the filtered Reddit dataset.
`src/pipeline/select_reddit_fields.py`	Keeps the core Reddit fields needed for analysis, including post ID, subreddit, author, timestamp, title, and body.
`src/pipeline/build_reddit_final_text.py`	Combines title and body into one final text field for sentiment prediction and topic tagging.
`src/pipeline/preprocess_reddit_text.py`	Cleans Reddit text using the same shared preprocessing function used during Sentiment140 training.
`src/pipeline/predict_reddit_sentiment.py`	Loads the saved TF-IDF vectorizer and Logistic Regression model, then predicts Reddit sentiment labels and probabilities.
`src/analysis/group_reddit_by_time.py`	Converts timestamps into year, month, year-month, and study-period fields.
`src/analysis/tag_reddit_topics.py`	Tags posts into topic buckets using cleaned keyword rules and regex matching.
`src/analysis/compare_sentiment_by_topic.py`	Computes positive and negative sentiment rates for each topic bucket.
`src/analysis/measure_topic_trends_over_time.py`	Measures monthly topic counts and topic shares over time.
`src/reporting/test_project_hypotheses.py`	Runs the H1 through H4 hypothesis checks and writes structured hypothesis output files.
`src/reporting/create_final_report_outputs.py`	Creates final report tables, plots, and written summaries.

B. Appendix B: Reproducible Command Sequence

The project can be rerun from the repository root by following the numbered workflow in Table 10. The Reddit input shown in Step 2 should point to the monthly mini-subreddit files used for the

study.

Table 10: Reproducible project execution workflow.

Step	Stage	Command
1	Train sentiment model	`pythonsrc/training/train_sentiment.py`
2	Process Reddit input files	`pythonsrc/pipeline/filter_reddit_submissions.py--input"src/data/reddit/monthly_subreddit_files/"`
3	Select Reddit fields	`pythonsrc/pipeline/select_reddit_fields.py`
4	Build final text field	`pythonsrc/pipeline/build_reddit_final_text.py`
5	Preprocess Reddit text	`pythonsrc/pipeline/preprocess_reddit_text.py`
6	Predict Reddit sentiment	`pythonsrc/pipeline/predict_reddit_sentiment.py`
7	Add time groups	`pythonsrc/analysis/group_reddit_by_time.py`
8	Define topic buckets	`pythonsrc/utils/topic_buckets.py`
9	Tag Reddit topics	`pythonsrc/analysis/tag_reddit_topics.py`
10	Compare topic sentiment	`pythonsrc/analysis/compare_sentiment_by_topic.py`
11	Measure topic trends	`pythonsrc/analysis/measure_topic_trends_over_time.py`
12	Test hypotheses	`pythonsrc/reporting/test_project_hypotheses.py`
13	Create final outputs	`pythonsrc/reporting/create_final_report_outputs.py`

Replication note. For the full final study, the Reddit input path should point to the full set of monthly mini-subreddit files for `cscareerquestions`, `csMajors`, `recruitinghell`, and `jobs`.

C. Appendix C: Code Sources and External Libraries

All project-specific pipeline code was written for this EECS 767 project. The full source code is available in the project repository. The project also uses third-party Python libraries for standard data processing, machine learning, compressed-file reading, model saving, and plotting.

- **Source code repository:** https://github.com/nifemi-l/cs-job-market-analysis
- **Project code:** preprocessing, training orchestration, Reddit filtering, text construction, topic tagging, hypothesis testing, final table generation, and plotting scripts.
- **External libraries:** pandas, NumPy, scikit-learn, matplotlib, joblib, and zstandard.
- **External datasets:** Sentiment140, Reddit submissions archive, and Layoffs.fyi-based layoff data.

References

[1] A. Go, R. Bhayani, and L. Huang. *Twitter Sentiment Classification using Distant Supervision*. Sentiment140 dataset. https://www.kaggle.com/datasets/kazanova/sentiment140/data

[2] J. Baumgartner, S. Zannettou, B. Keegan, M. Squire, and J. Blackburn. *The Pushshift Reddit Dataset*. Proceedings of the International AAAI Conference on Web and Social Media, 2020. https://ojs.aaai.org/index.php/ICWSM/article/view/7347

[3] Academic Torrents. *Reddit comments/submissions archive*. https://academictorrents.com/details/3d426c47c767d40f82c7ef0f47c3acacedd2bf44

[4] Kaggle. *Layoffs 2022 dataset, sourced from Layoffs.fyi*. https://www.kaggle.com/datasets/swaptr/layoffs-2022

[5] F. Pedregosa et al. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 2011. https://scikit-learn.org/

[6] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. https://nlp.stanford.edu/IR-book/