thesis step

1. データ収集（data_collection.py）

- 文字列型に統一して混合型エラーを防止。
- 欠損・重複はまだ触らず “Raw” として保存。

2. クリーニング & トークン化（text_cleaning.py）

処理手順

2-1	Unicode **NFKC** 正規化 → 英数を半角・小文字化
2-2	https://… URL, メール, ハッシュタグ, HTMLタグ を正規表現で除去
2-3	記号・絵文字・数字（日/英）を一括置換
2-4	**Janome** を -o wakati モードでトークン化（語単位を維持）
2-5	品詞フィルタ：名詞, 形容詞 のみ残す
2-6	追加フィルタ：・1文字ひらがな／し を除外・独自 stop_words.py リストに含まれる語を除外
2-7	残ったトークンをスペース結合→ clean_joined 列

3. ベクトル化（vectorize.py 内で呼び出し）

4. K-means クラスタリング

1. **シルエット法**（cluster_eval.py）
- k = 2〜10 を試行 → 最高スコアの k を採用
2. **クラスタ学習**（clustering.py）
- KMeans(n_clusters=k).fit(X)
- ラベルを cluster 列として付与
- models/<slug>_kmeans_k{k}.pkl に保存

5. クラスタ内フレーズ抽出（topic_inspection.py）

- 各クラスタで出現頻度 or TF-IDF 上位 n-gram を 10 個抽出
- _NEG_KEYWORDS を用いて苦情系ワードを強調
- 実装：summarise_cluster(df, cid) が (phrases, sample_reviews) を返却

6. ルールベース感情分析（sentiment_analysis.py）

キーワード一致で Negative 判定

7. LDA トピックモデル（lda_modeling.py）

- 上位語を手動でテーマ命名
- 各レビューに **支配的トピック** を付与：dominant_topic = lda.transform(Xc).argmax(axis=1)
- モデル保存：models/<slug>_lda.pkl

8. ハイブリッド解析

分析	手法	成果物
クラスタ × トピック クロス集計	`pd.crosstab(cluster, dominant_topic, normalize="index")`	Heatmap (sns.heatmap)
トピック出現率比較	8 トピックの % を両コーパスで計算	折れ線 / Δバーチャート
代表コメント抽出	各トピック内で Negative keyword ヒット数が多い順に抽出	引用用テキスト

9. 結果統合スクリプト（analysis_summary.py）

1. 高 & 多コメント両データの トピック出現率 を計算
2. Δ(High − Commented) を算出
3. **CSV** (`topic_summary.csv`)、 **Markdown** 表、 **Δバーチャート** (`delta_chart.png`) を自動生成