# Student Cluster Analysis via LDA — Project Documentation (Latest) 25 Aug 2025

**Overview**

This project clusters students by themes of their **elective courses**. The pipeline:

1. Cleans/normalizes course text (titles + descriptions)
2. Generates candidate keywords (n-grams, RAKE, YAKE)
3. Enforces a **controlled vocabulary** (keep/remove map + lemmatization)
4. Aggregates to a **student × keyword** count matrix
5. Collapses highly correlated words to the more generic form
6. Fits **LDA** (topic modeling) to reveal latent “interest clusters”
7. Exports auditable CSVs for downstream analysis

**Code files (source of truth)**

1. **NLP\_LDA\_coursenames\_colab.ipynb**  
   Title-based NLP preprocessing, tokenization, n-grams, TF-IDF inspection, RAKE & YAKE (per-title) candidate generation. Writes intermediate CSVs to inform curation.
2. **LDA final.ipynb**  
   Canonical keyword selection (with **description-based YAKE fallback**), lemmatization, student aggregation, sparse feature building, **LDA** model selection, topic inspection, and exports.

**Data inputs**

* **Enrollments (Excel)**  
  /content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/raw data/Project - Student Course Clustering/Student Course Elective Enrollments Graduates 2016-2025.xlsx
* **Course descriptions (CSV)**  
  /content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Course Clustering/Up to 20213 Course Descriptions.csv
* **Keyword maps (curated CSVs)**
  + /content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/keyword-yake5-map.csv
  + /content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/keywords\_keep\_map.csv

The **keep map** is the source of truth for vocabulary (column keep tag == 1 means “approved”).

**CSV artifacts (what gets written)**

**From NLP\_LDA\_coursenames\_colab.ipynb**

* tfidf\_scores ngrams 1,2,3.csv — global n-gram salience (inspection)
* /.../Student Cluster Analysis GDrive/idf\_ngram\_COUNT.csv — n-gram inventory
* /.../Student Cluster Analysis GDrive/ngrams\_rake\_yake\_keywords.csv — per-course title candidates (n-grams + RAKE + YAKE)

Optional (commented in notebook): course names uni-bi-trigrams.csv, unique\_subset.csv.

**From LDA final.ipynb**

* /.../Student Cluster Analysis GDrive/stem\_lemmatized\_kept\_keywords.csv — **canonical course keyword** (+ lemma) after title path and **new description fallback**
* /.../Student Cluster Analysis GDrive/keyword correlation\_matrix.csv — keyword–keyword correlation from student × keyword matrix
* /content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA-20250714T173604Z-1-001/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/df\_keywords\_final.csv

**Final exports** (add if not already):

* /.../Student Cluster Analysis GDrive/lda\_topic\_term\_weights.csv — topic × term weights
* /.../Student Cluster Analysis GDrive/lda\_student\_topic\_distribution.csv — student × topic probabilities (+ dominant topic/prob)
* /.../Student Cluster Analysis GDrive/lda\_topic\_labels\_top10.csv — compact labels (top-10 terms/topic)
* /.../Student Cluster Analysis GDrive/fallback\_na\_resolution\_log.csv — audit of NA→filled decisions (esp. OOV cases)

**Processing details (what & why)**

**1) Title-based preprocessing & candidates *(in NLP\_LDA\_coursenames\_colab.ipynb)***

* **Tokenization** with spaCy; lowercase; drop punctuation/spaces
* **n-grams (1–3)** to preserve meaningful phrases (e.g., “time series”)
* **TF-IDF** (for inspection only) to calibrate pruning and sanity-check candidates
* **RAKE + YAKE (top-k)** on course **titles** to propose high-precision candidates
* Export: ngrams\_rake\_yake\_keywords.csv for curation/inspection

**Why:** LDA works best on compact, semantically consistent vocabularies; this stage proposes candidates, not the final tokens.

**2) Canonical keyword selection (with description-based fallback) *(in LDA final.ipynb)***

**Inputs used here**

* df\_keywords with columns: keyword (may be NA), course\_description, lemma\_root (may be NA), plus course identifiers
* tagged\_keywords (keep map) with keep tag and unique yake\_top5\_keywords

**Helper**

* get\_root\_one(token) → single lemma string (noun-leaning, POS-aware)

**A. Primary (title path)**

* Use keep/remove map + lemmatization over **title** candidates.
* If a canonical keyword exists → keep it.

**B. NEW fallback (description YAKE, ranked & widened)**  
When keyword is NA after the title path:

1. Compute **YAKE(desc)** **top-5** unigrams → store in **desc\_yake\_top5** (always a list/None)
2. If no overlap with keep map, compute **YAKE(desc) top-20** → store in **desc\_yake\_top20**
3. **Scan in order**:
   * First, the **top-5** list: pick the **first** candidate that appears in keep\_set (rank preserved)
   * If none, scan the **top-20** list for the first keep-set match
4. If still none, **OOV fallback**: choose the **top YAKE(desc)** candidate (ensures not NA)
5. If YAKE returns nothing at all, extract a **simple token** from description (simple\_token\_fallback) as last resort
6. In some cases, there is no description at all for coursenames—Potential to use YAKED combined\_name\_x (the course name from the raw df), making sure it is in keep\_map
   * If not in keep\_map—take any keyword from combined\_name\_x

**C. Lemma guarantee**

* Compute lemma\_root **only where** it was NA **and** a keyword now exists
* Use a **safe** wrapper: if the lemmatizer returns empty/None, fall back to the **keyword string**
* Final guard: if keyword is present, ensure lemma\_root is non-null (lowercased keyword as a last fill)

**Why:** This preserves vocabulary control **when possible** (keep-set overlap), widens search to **top-20** for recall, and still assigns a token when no overlap exists (OOV fallback), ensuring every course yields a usable lemma\_root.

**Columns introduced / used in this step**

* desc\_yake\_top5 — list[str] or None
* desc\_yake\_top20 — list[str] or None
* keyword — final canonical keyword after fallback logic (lowercased)
* lemma\_root — guaranteed non-null whenever keyword exists

**3) Student-level aggregation**

* Group by **student id (uni)** to construct:
  + courses: list of courses taken (traceability)
  + keywords: list of canonical keywords (post-fallback)
* Build sparse **student × keyword** counts using CountVectorizer configured for **pre-tokenized** input (tokenizer=lambda x: x, lowercase=False)
* Quick sanity: number of zero-keyword students, matrix sparsity

**4) Keyword correlation diagnostics**

* Compute keyword–keyword correlation from the student × keyword matrix
* Export: keyword correlation\_matrix.csv (helps spot near-duplicates/redundancy)

**5) LDA modeling & model selection**

* Use **counts** (not TF-IDF) with sklearn.decomposition.LatentDirichletAllocation(learning\_method="online", random\_state=42)
* **RandomizedSearchCV** over:
  + n\_components ∈ {1,2,3,4,5,6,7,8,9,10,15,20,25,30}
  + learning\_decay ∈ {0.5, 0.7}
  + batch\_size ∈ {128, 256, 2000}
* Optional **GridSearchCV** refinement
* Select **best estimator** via CV log-likelihood; report params, score, full-data perplexity
* **Topic inspection:** top terms from components\_, document–topic matrix via .transform(X), dominant topic/prob per student
* (Optional) pyLDAvis for interactive exploration

**Recommended exports**

* lda\_topic\_term\_weights.csv
* lda\_student\_topic\_distribution.csv
* lda\_topic\_labels\_top10.csv
* fallback\_na\_resolution\_log.csv (see below)

**Audit logging (recommended)**

Because you now allow **OOV fallback**, log those cases to aid future vocabulary updates:

* **fallback\_na\_resolution\_log.csv** (suggested columns)
  + course\_id / course\_code
  + desc\_yake\_top5 (pipe-joined)
  + desc\_yake\_top20 (pipe-joined)
  + chosen\_keyword
  + in\_keep\_set (True/False)
  + decision\_reason ∈ {top5\_overlap, top20\_overlap, oov\_top\_yake, simple\_token}

Use this to:

* Spot frequent **OOV** keywords worth adding to keywords\_keep\_map.csv
* Review courses whose descriptions yield poor candidates

**How to run (minimal sequence)**

1. **NLP\_LDA\_coursenames\_colab.ipynb**
   * Mount Drive; install deps; run all cells
   * Confirm exports:
     + tfidf\_scores ngrams 1,2,3.csv
     + .../idf\_ngram\_COUNT.csv
     + .../ngrams\_rake\_yake\_keywords.csv
2. **(Optional) Update keep map**
   * Inspect TF-IDF and candidates; refresh keywords\_keep\_map.csv if needed
3. **LDA final.ipynb**
   * Ensure the **description fallback section** creates desc\_yake\_top5 **and** desc\_yake\_top20
   * Confirm exports:
     + .../stem\_lemmatized\_kept\_keywords.csv
     + .../keyword correlation\_matrix.csv
     + *(Recommended)* topic and audit CSVs listed above

**One-look checklist (re-runs & hand-offs)**

* Paths valid; Drive mounted
* keywords\_keep\_map.csv current (keep tags correct)
* Title candidate exports created
* **desc\_yake\_top5**/**desc\_yake\_top20** present and are **lists/None**
* stem\_lemmatized\_kept\_keywords.csv refreshed (spot-check changed rows)
* Student × keyword matrix rebuilt; sparsity reasonable
* LDA refit; topic exports written
* fallback\_na\_resolution\_log.csv reviewed (promote good OOVs into keep map)

**Environment & reproducibility**

* **Libraries:** spaCy, NLTK (stopwords, wordnet), **yake**, **rake-nltk**, scikit-learn, pyLDAvis, NumPy <2 (pinned in preprocessing)
* **Random seeds:** random\_state=42 where applicable
* **Paths:** assume **Colab + Drive** under /content/drive/...; otherwise define a BASE\_DIR config

**Known limitations & next steps**

* **OOV fallback** improves recall but may introduce drift; regularly review fallback\_na\_resolution\_log.csv and update the keep map with high-value OOV tokens.
* Consider tuning doc\_topic\_prior / topic\_word\_prior for sparser topics, or **guided LDA** (seed words) to lock specific themes.
* Title/description quality varies by year; occasionally re-run YAKE with adjusted top or n if you see under-capture in certain domains.

**Changelog (vs. earlier versions)**

* **Fallback logic expanded:** scan **YAKE(desc) top-5** then **top-20** for keep-set overlap; if none, **OOV fallback** to top candidate; last-ditch simple token extraction.
* **Guarantee non-null lemma\_root:** safe lemmatization + final guard.
* Introduced **desc\_yake\_top5** and **desc\_yake\_top20** columns (lists).
* Added **audit log** recommendation to track overlap vs. OOV decisions.

# OLD 22 Aug 2025:

**Student Cluster Analysis via LDA — Project Documentation**

**Code files (source of truth)**

1. **NLP\_LDA\_coursenames\_colab.ipynb**  
   Focus: NLP preprocessing on course names → n-grams, TF-IDF scoring, RAKE/YAKE keyword extraction, and exports to support downstream modeling.
2. **LDA final.ipynb**  
   Focus: join/clean keywords per student, lemmatization + keep/remove logic, sparse bag-of-words construction, LDA model selection (Grid/Randomized search), topic inspection & document–topic summaries.

**Data inputs (read by the notebooks)**

* **Student enrollments (Excel):**  
  /content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/raw data/Project - Student Course Clustering/Student Course Elective Enrollments Graduates 2016-2025.xlsx
* **Course descriptions (CSV):**  
  /content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Course Clustering/Up to 20213 Course Descriptions.csv
* **Keyword maps from manual/YAKE work (CSVs):**
  + /content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/keyword-yake5-map.csv
  + /content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/keywords\_keep\_map.csv

These “maps” carry your human-in-the-loop decisions about which detected keywords to keep/normalize; the notebooks treat them as authoritative.

**Artifacts generated (CSV exports)**

**From NLP\_LDA\_coursenames\_colab.ipynb**

* **tfidf\_scores ngrams 1,2,3.csv** (local working directory)  
  Top n-grams (1–3) by summed TF-IDF score across course names. Used to sense-check which phrases dominate and to inform keep/remove decisions.
* **/content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/idf\_ngram\_COUNT.csv**  
  n-gram inventory with counts/IDF to quantify coverage beyond raw TF-IDF sums.
* **/content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/ngrams\_rake\_yake\_keywords.csv**  
  Course-level table combining:
  + Cleaned course name
  + Uni/bi/tri-grams (from spaCy tokenization)
  + RAKE keywords
  + **YAKE top-5 keywords** (your preferred default because it’s high-precision and simple to use)

**Commented-out exports (not currently written):**

* course names uni-bi-trigrams.csv
* unique\_subset.csv  
  If you want these on disk, just uncomment those to\_csv lines.

**From LDA final.ipynb**

* **/content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/stem\_lemmatized\_kept\_keywords.csv**  
  The **per-course** canonical keyword after applying the keep/remove map and your lemmatization normalization logic (see Processing details below).
* **/content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/keyword correlation\_matrix.csv**  
  Correlation matrix across keywords (from the student-by-keyword sparse matrix) for quick “which keywords travel together” checks prior to topic modeling.

Tip: If you also want **document–topic** and **topic–term** tables as CSV, see the “Recommended extra exports” section at the end — the notebooks compute them but do not currently save them.

**Processing details (what & why)**

**1) Text normalization & n-grams (in NLP\_LDA\_coursenames\_colab.ipynb)**

* **Tokenization**: via **spaCy** (en\_core\_web\_sm), lowercased; punctuation/spaces removed.
* **n-grams**: explicit **uni/bi/tri-grams** constructed from token stream to preserve meaningful phrases in course titles (e.g., “machine learning”, “time series analysis”).  
  *Rationale:* LDA on bag-of-words benefits when semantically dense phrases survive preprocessing; otherwise topics fragment across synonyms/single tokens.
* **TF-IDF scoring**: TfidfVectorizer(ngram\_range=(1,3)) to identify important n-grams globally; exported to tfidf\_scores ngrams 1,2,3.csv for manual inspection.  
  *Rationale:* you used TF-IDF **not** for modeling (LDA expects counts), but to **guide vocabulary pruning** and validate the YAKE/RAKE output.
* **RAKE** (rake\_nltk) & **YAKE**:
  + RAKE to get multi-word candidates.
  + **YAKE** (top-5 per course) as the main candidate source — you preferred its precision and the easy “top-k” interface.
  + Combined outputs exported to ngrams\_rake\_yake\_keywords.csv.  
    *Rationale:* Yields high-signal candidate features while keeping feature space compact for later LDA.

**2) Keep/remove mapping & lemmatization (in LDA final.ipynb)**

* **Stopwords**: NLTK English stopwords downloaded and used in auxiliary filtering; main control is via your **keywords\_keep\_map** file (human-curated).
* **Lemmatization**: NLTK WordNetLemmatizer applied with a POS-aware pass (n/v/a backoff).
  + Heuristic to **prefer nouns**; if ambiguous, you fall back to frequency/shorter string (as per your comments).
* **“Keep” logic**:
  + Join **YAKE candidates** with **keywords\_keep\_map**; if a given course had no acceptable keyword after keep/remove rules, you fallback to the first YAKE candidate.
  + The resulting **canonical keyword** per course is written to stem\_lemmatized\_kept\_keywords.csv.  
    *Rationale:* LDA works best with consistent, semantically stable tokens; your mapping enforces that consistency across messy free-text course names.

**3) Student-level aggregation (in LDA final.ipynb)**

* Group by **student ID (uni)** to build:
  + courses: list of all enrolled courses (for traceability)
  + keywords: list of **canonical** keywords for those courses
* Construct a sparse **student × keyword** matrix with CountVectorizer(tokenizer=lambda x: x, lowercase=False) on the **pre-tokenized** keywords lists.
  + You explicitly skip lowercase here because normalization already happened upstream.
* Quick diagnostics:
  + Count students with no keywords (sanity check).
  + Calculate sparsity (nnz / (n\_docs \* n\_terms)) to understand matrix shape/density.

**4) Keyword correlation check (in LDA final.ipynb)**

* Convert a slice of the sparse matrix to a DataFrame and compute a **keyword–keyword correlation matrix**; export to keyword correlation\_matrix.csv.  
  *Rationale:* sanity-check co-movement; helps spot overly synonymous features you might want to merge before LDA.

**5) LDA modeling & model selection (in LDA final.ipynb)**

* **Vectorizer**: Count (not TF-IDF), because **sklearn LDA** assumes counts.
* **Model**: sklearn.decomposition.LatentDirichletAllocation(learning\_method='online', random\_state=42)
* **Hyperparameter search**:
  + **RandomizedSearchCV** with
    - n\_components ∈ {1,2,3,4,5,6,7,8,9,10,15,20,25,30}
    - learning\_decay ∈ {0.5, 0.7} (per your note that ~0.7 often works well)
    - batch\_size ∈ {128, 256, 2000}
  + A **GridSearchCV** block is also present over a narrower grid.
* **Model selection**: choose **best\_lda\_model** using .best\_estimator\_, with logs printed for:
  + best\_params\_ (n\_components, learning\_decay, batch\_size)
  + best\_score\_ (CV log-likelihood)
  + Perplexity on full X
* **Topic inspection**:
  + **Top terms per topic** from best\_lda\_model.components\_ (weights), printed for quick validation.
  + **Document–topic** distribution via best\_lda\_model.transform(X), then:
    - Hard assignment (dominant\_topic) and its probability (dominant\_prob) per student.
    - Topic counts across students (value\_counts()).
  + **pyLDAvis** prepared (TSNE mode) for interactive exploration (not exported).

**Note:** In the current notebooks, topic tables and doc–topic tables are computed and displayed but not written to CSV. See “Recommended extra exports” below to persist them.

**Reproducibility & environment notes**

* Uses **spaCy** (en\_core\_web\_sm), **NLTK** (stopwords, wordnet), **yake**, **rake-nltk**, **scikit-learn**, **pyLDAvis**.
* You pin **NumPy <2** in the first notebook to avoid API breaks.
* Random seeds set where relevant (random\_state=42).
* Paths assume **Google Drive** mounted under /content/drive/.... If running outside Colab, update paths or introduce a BASE\_DIR config cell.

**What each artifact is for (at a glance)**

| **File** | **Purpose** |
| --- | --- |
| tfidf\_scores ngrams 1,2,3.csv | Quick view of globally important n-grams to guide curation. |
| idf\_ngram\_COUNT.csv | Inventory with counts/IDF for coverage checks and pruning. |
| ngrams\_rake\_yake\_keywords.csv | Course-level features: n-grams + RAKE + **YAKE top-5** (your default candidates). |
| stem\_lemmatized\_kept\_keywords.csv | Canonical **kept** keywords per course after your keep/remove + lemmatization pipeline. |
| keyword correlation\_matrix.csv | Keyword co-movement to spot redundancy & cluster structure pre-LDA. |

**How to run (minimal sequence)**

1. Open **NLP\_LDA\_coursenames\_colab.ipynb**
   * Ensure Drive is mounted; install deps.
   * Load enrollments Excel; generate n-grams and RAKE/YAKE.
   * **Exports written:**
     + tfidf\_scores ngrams 1,2,3.csv
     + .../idf\_ngram\_COUNT.csv
     + .../ngrams\_rake\_yake\_keywords.csv
2. Curate/refresh **keywords\_keep\_map.csv** (outside notebook)
   * Optionally inspect TF-IDF and ngrams\_rake\_yake\_keywords.csv to update decisions.
3. Open **LDA final.ipynb**
   * Load keyword-yake5-map.csv and keywords\_keep\_map.csv.
   * Run lemmatization + keep logic → **stem\_lemmatized\_kept\_keywords.csv**.
   * Build student×keyword matrix; export **keyword correlation\_matrix.csv**.
   * Run LDA random/grid search; inspect best model; review topics and doc–topic assignments.

**Recommended extra exports (add once and done)**

Drop these at the end of **LDA final.ipynb** so downstream analysis or dashboards don’t depend on notebook state:

# 1) Topic → top terms (weights)

TOP\_N = 15

terms = vectorizer.get\_feature\_names\_out()

topicnames = [f"Topic{i}" for i in range(best\_lda\_model.n\_components)]

df\_topic\_keywords = pd.DataFrame(

best\_lda\_model.components\_, index=topicnames, columns=terms

).T # terms as rows

df\_topic\_keywords.sort\_values(by=topicnames, ascending=False).head()

df\_topic\_keywords.to\_csv(

"/content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/lda\_topic\_term\_weights.csv"

)

# 2) Student → topic distribution (+ dominant topic)

lda\_output = best\_lda\_model.transform(X)

docnames = df\_student["uni"].tolist()

df\_document\_topic = pd.DataFrame(lda\_output, columns=topicnames, index=docnames)

df\_document\_topic["dominant\_topic"] = df\_document\_topic[topicnames].values.argmax(axis=1)

df\_document\_topic["dominant\_prob"] = df\_document\_topic[topicnames].max(axis=1)

df\_document\_topic.to\_csv(

"/content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/lda\_student\_topic\_distribution.csv"

)

And if you want a lighter, human-readable topic label file:

# 3) Compact topic labels (top-10 terms per topic)

rows = []

for t in range(best\_lda\_model.n\_components):

top\_idx = np.argsort(best\_lda\_model.components\_[t])[-10:][::-1]

rows.append({

"topic": f"Topic{t}",

"top\_terms": ", ".join(terms[top\_idx])

})

pd.DataFrame(rows).to\_csv(

"/content/drive/MyDrive/CBS/CBS-Github-DND/Student Cluster Analysis and LDA/Student Cluster Analysis GDrive/lda\_topic\_labels\_top10.csv",

index=False

)

**Known limitations / notes for the next person**

* **Exports for topics & doc–topic matrices** are not currently saved — add the cells above if you need persistent outputs (recommended).
* **Two exports in the first notebook are commented out**; uncomment if those intermediate files help audit the pipeline.
* **Vectorizer uses pre-tokenized lists**; any change upstream (e.g., changing lemma rules or keep map) **must** be re-run before LDA.
* **Model quality**: You’re using default priors and online learning — great for scale, but consider a quick pass with additional priors (doc\_topic\_prior, topic\_word\_prior) if you want tighter/sparser topics, and/or try a small **guided LDA** (seed words) if specific themes are desired.

**One-look checklist (when re-running)**

* Drive mounted and paths valid
* keywords\_keep\_map.csv up to date
* Run **NLP notebook** → confirm 3 exports appear
* Run **LDA notebook** → confirm 2 exports appear
* (Optional) Write extra CSVs for topic terms & doc–topic distributions
* Review topic labels (face validity) and dominant topic distribution