Implications of Concept Extraction with Large Language Models

Owen Finucan and **Jonah Smith** and **Timothy Mooney**
School of Computing and Information
University of Pittsburgh
Pittsburgh, PA, United States
{OTF6, JWS130, TJM182}@pitt.edu

Abstract

Large language models (LLMs) are increasingly used as educational tools, yet their ability to reliably extract instructional concepts from lecture materials remains underexplored. We investigate concept extraction from university slide decks, comparing two LLM-based pipelines against expert human annotations derived from a shared codebook. The first model, Model 1, was a text-only pipeline that fed slide text directly into the LLM. In contrast, Model 2 was a multimodal pipeline that used both slide text and slide summaries derived from screenshots. Using per-slide F1 against human gold labels, Model 1 achieves a moderate performance (average F1 $\approx$ 0.647), whereas Model 2 performs poorly (average F1 $\approx$ 0.440). A stepwise regression of Model 1 identified part-of-speech entropy (positive), verbosity penalty (negative), and slide compression potential (marginal) as robust predictors of concept extraction performance. We suggest ways to expand these minimalist pipelines to improve performance.

1 Introduction

Large language models (LLMs) are increasingly being deployed as educational tools (Kasneci et al., 2023; Wang et al., 2024). In particular, LLMs have shown promise in lecture summarization, study material generation, and content retrieval (Wang et al., 2024; Song et al., 2023). In higher education, slide decks are among the most common ways for instructors to transmit information to students (León and Escudero-Vilaplana, 2021; McCrory, 2022). These slide decks cover the core concepts students are expected to master to succeed in a given course. While experts can extract concepts at a high rate, students actively learning the material are unlikely to replicate expert performance in identifying and organizing them (Chen and Chen, 2010; Hassani et al., 2022). Moreover, because experts have already mastered the material, manually pulling concepts from slides is a nontrivial and inefficient use of their time. This motivates the use of LLMs to extract concepts from slide decks automatically. We question whether LLMs can generate concepts with high success rates and what factors make some slides easier for LLMs to parse than others (Krishnaswamy, 2024; Norouzi et al., 2025).

Many elements of a slide might influence how an LLM performs at concept extraction. These include the density of terminology, the presence of mathematical symbols, the use of abbreviations and examples, and the degree of structural organization (e.g., numbering, bullets, headers, footers) (Hassani et al., 2022; Song et al., 2023). Without a principled study, it is difficult to determine whether performance on a given slide deck is due to the model, the prompt(s) used, or the slide deck's structure.

In this work, we investigate the performance of two LLM-based pipelines for concept extraction from lecture slide decks (Krishnaswamy, 2024; Norouzi et al., 2025). We collected gold-standard concept annotations from high-skilled human annotators using a shared codebook and compared them with the concepts produced by the LLM pipelines. We then analyze how slide-level characteristics influence model performance as measured by F1 scores.

2 Task & Problem Definition

The goal of this research is to analyze how LLMs perform at concept extraction from slide decks. We define a *concept* as the smallest standalone unit of knowledge. Given various input formats (e.g., `.txt` and `.pptx` files), we prompt an LLM to extract concepts. The output is a `.txt` file containing the set of predicted concepts for each slide. These predictions are then evaluated against gold-standard human labels using F1 scores.

A key component of this research is understanding which linguistic or structural properties of a slide influence extraction performance. In particular, we will focus on two baseline control variables in our regression analysis: *vocabulary entropy* and *average line length*. We define vocabulary entropy as the Shannon entropy of the slide's token distribution. This captures the variance in terminology throughout a slide deck. Average line length is defined as the mean number of characters per non-empty line. This allows for analysis of verbosity and sentence compactness. Together, these two variables provide a baseline measure of linguistic diversity and textual density against which all other metrics are evaluated.

3 Structure & Setup

3.1 Codebook

The codebook is a list of rules that **must** be followed when performing annotations. It is defined as follows:

- **Grounding Rule** - Concepts must appear directly in the slide text.
- **Granularity Rule** - Use the most meaningful level of specificity.
- **Definition Rule** - Include terms that are meaningfully defined.
- **Emphasis Rule** - Include terms that are heavily emphasized or highlighted.
- **Abbreviation Rule** - Include both the full form and its acronyms as separate concepts.
- **Unified Terms Rule** - Include multi-word expressions when they function as a single unified idea.
- **Generic Terms Rule** - Exclude vague or context-free terms.

Both human annotators and LLM pipelines were provided with a copy of the codebook and were instructed to follow it as closely as possible.

3.2 Annotations

What was provided to a given annotator depended on the annotator type. High-skilled human annotators (who were knowledgeable in the topics being discussed) and all LLMs were given a standard `.txt` file containing the slide deck text, along with the codebook. In addition, human annotators and image-based models were provided with the slideshow itself. Human annotators were given direct access to the original `.pptx` file, while image-capable LLMs were provided with `.jpg` screenshots of the slides. Models that were not image-based received only the slide-deck text and the codebook.

Human annotators performed their annotations in Doccano and exported the results as a `.jsonl` file, whereas LLMs were instructed to output a `.txt` file once concept extraction was complete. The human-produced annotations serve as the gold labels and are used to score all LLM outputs.

3.3 Role of the LLM

Although LLMs were always prompted with the full codebook, certain rules required additional emphasis. Concepts were required to be explicitly found within the given slide deck; models were instructed not to infer or hallucinate concepts that were not supported by the slide text or images. A third party should be able to map each predicted concept verbatim to the text found on the slide. This grounding constraint helps prevent the extraction of non-concepts.

Prompts further emphasized evidence-based extraction and strict adherence to the codebook. LLMs were reminded to remove any terms that are not explicitly defined, emphasized, or instantiated via an example, and to exclude course titles, structural labels, or institution-specific language from the concept set. Only core instructional concepts were to be retained.

To standardize evaluation, LLMs were also prompted to produce a fixed output format: a `.txt` file with one concept per line, no numbering or commentary, all lowercase (unless an acronym), and with abbreviations to capture all possible concepts.

4 Models

4.1 Metrics

We tested a variety of slide-level features in an effort to capture various lexical, structural, syntactic, and semantic properties of slides. The full list of explored metrics can be found in Appendix A.

The dataset consisted of seven computer science (CS) courses at the University of Pittsburgh. Model 1 was evaluated from text from CS-0441, CS-0447, CS-0449, CS-1502, CS-1541, CS-1550, and CS-1622. As both text content and slide decks (in `.pptx`) were only available for CS-0007, CS-0449, and CS-1622, these were the only courses that Model 2 interacted with.

4.2 Model 1

The first model was a simplistic, naive approach to the concept extraction. The basis of the model, as shown in Figure 1, was extracting the text and then directly piping the text into the LLM.

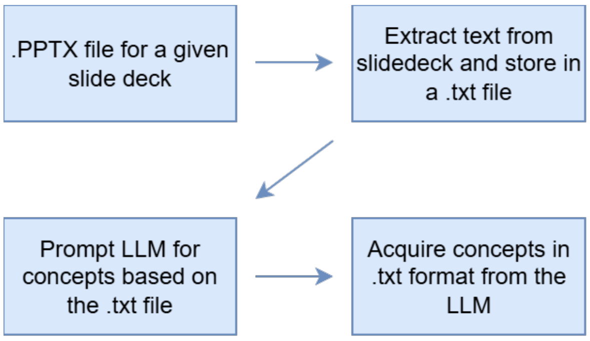

.PPTX file for a given slide deck
Extract text from slidedeck and store in a .txt file
Prompt LLM for concepts based on the .txt file
Acquire concepts in .txt format from the LLM

Figure 1: Concept-extraction pipeline for Model 1

4.2.1 Implementation Specifics

For Model 1, the pipeline operated exclusively on the raw slide text extracted from each deck. No screenshots or visual inputs were used at any stage. The per-slide concept annotations were generated using `gpt-5` with a temperature of 1.0. Prompting details are included in Appendix C.

The prompt was constructed around the codebook, emphasizing the strict grounding of all concepts in the extracted text. The model was instructed to avoid including any course-specific identifiers (institutional course numbers, instructor names, or structural labels) and to refrain from inferring concepts not explicitly stated in the slide content. The prompt also enforced the required output format: one concept per line, no numbering, lowercase unless an acronym, and canonical word forms only.

As Model 1 only utilizes text, no batching was required. All that was passed into the LLM was the text along with slide numbers. The model was first instructed to generate an initial concept list and then to re-evaluate it, removing any terms that did not meet the definition of an instructional concept according to the codebook.

4.2.2 Results

As shown in Table 3, Model 1 achieves moderate performance with concept extraction. Performance varies dramatically between courses (with a net range of 0.221). By F1 score, CS-1502 proved to be the most challenging course, while CS-0447 was the easiest. It is unsurprising that CS-1502 also has the highest variance, suggesting that this course includes slides that are particularly difficult for our naive model. In contrast, courses such as CS-0447, CS-0449, and CS-1622 show that it is possible to achieve a relatively high average F1 score while keeping the variance across slide decks low. We show the results of Model 1 in Figure

Class	Avg F1 Score	F1 Score Variance	Aggregate F1
CS-0007	0.750	0.0101	0.7650
CS-0441	0.572	0.0305	0.7356
CS-0447	0.721	0.0094	0.7562
CS-0449	0.709	0.0090	0.7528
CS-1502	0.667	0.0301	0.7601
CS-1541	0.667	0.0113	0.7272
CS-1550	0.660	0.0389	0.7745
CS-1622	0.702	0.0121	0.7592

Table 1: Model 1 F1 Scores

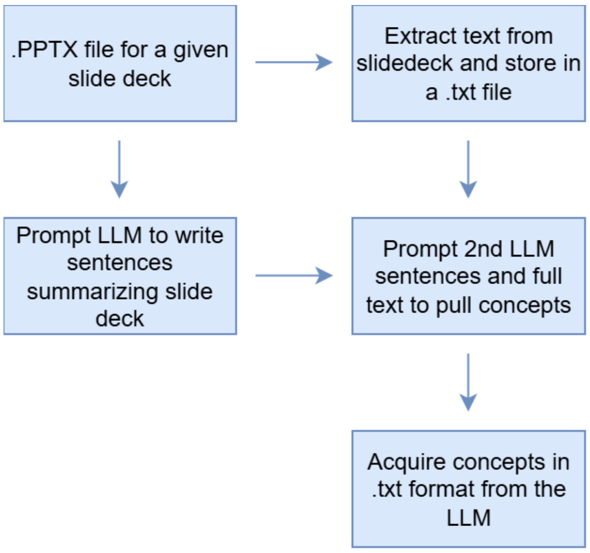
To better understand which slide-level properties (the density of terminology, the presence of mathematical symbols, abbreviations or examples, and organization of the slides) drive the results we found, we fit a linear regression model predicting F1 from our candidate metrics while controlling for vocabulary entropy and average line length (Table 4).

From Table 4, we see that the regression identifies three variables as particularly important: POS entropy, verbosity penalty, and slide compression potential. POS entropy is a strong positive predictor of F1 ($\beta = 0.162$, $t = 4.586$, $p < 10^{-5}$), indicating that slides with a broader range of part-of-speech patterns tend to yield better concept extraction (higher F1). The verbosity penalty is negatively associated with F1 ($\beta = -0.007$, $t = -3.332$, $p = 0.001$), meaning that slides with longer, denser lines perform worse than slides with shorter, more concise lines. Finally, slide compression potential has a negative but only marginally significant effect ($\beta = -0.338$, $t = -1.664$, $p = 0.098$), suggesting that high redundancy may hurt extraction performance, but the evidence is weaker.

As mentioned, we evaluated a larger set of metrics (see Appendix A), but none remained significant after including vocabulary entropy, average line length, and the three variables above in the model. In other words, their apparent effects on F1 were either redundant with POS entropy, verbosity, and compression potential or statistically negligible after controlling for these core dimensions of slide quality.

4.3 Model 2

The second model involved including visuals of the slides themselves in the pipeline. The model, as shown in Figure 2, started with the `.pptx` file. Two transformations were then applied. On one hand, the text was extracted from the slide deck. On the other hand, an LLM was prompted to generate sentences based on the slide deck. More specific information will be provided on this in the subsequent section. These outputs are then provided to a second LLM, and this LLM is instructed to output the list of concepts.



.PPTX file for a given slide deck
Extract text from slidedeck and store in a .txt file
Prompt LLM to write sentences summarizing slide deck
Prompt 2nd LLM sentences and full text to pull concepts
Acquire concepts in .txt format from the LLM

Figure 2: Concept-extraction pipeline for Model 2

4.3.1 Implementation Specifics

As with Model 1, the text extraction involved no use of LLMs. Screenshots of each slide (in `.jpg`) were taken and provided to the LLM.

For generating the per-slide annotations, `gpt-4.1-mini` was used with a temperature of 0.2. The full prompt can be found in Appendix E. Generally, the prompt included many elements from the codebook. Emphasis was placed on not including any course-specific information (institution-specific course numbers, course labels, or instructor names). To avoid rate limits, screenshots were provided in batches of 3 slide images per request. The model was also provided with the slide numbers for which it was generating the sentences. It was prompted to output a `.json` of the sentences.

For the LLM that took in both the slide deck text and the sentence summaries, `gpt-4.1-mini` was used with a temperature of 0.1. The full prompt can be found in Appendix E. The prompt was based on the codebook. It contained elements to try to guide the output ("do not include near-synonyms", "only the canonical or dictionary form"). It contained lots of specific information about the output (one concept per line, no numbers, concepts should only be four to five words). The prompt told the LLM to place significantly more emphasis on the text than on the screenshot-generated sentences (90/10). It was also told to create its initial list of concepts, then to go back through and weed out any terms that might not be concepts.

4.3.2 Results

Compared to Model 1, Model 2 produces much less output. Of the courses in the dataset, only three provided both the course text (`.txt`) and the `.pptx` file. This limited our results with this model to those three slide deck sets.

As shown in Table 5, we achieved poor results with Model 2. We visualize the results within Figure **??**. Though we see a wide range (0.217) of F1 scores, the variance remains relatively low (0.016 ± 0.002). This shows that for a given course slide deck set, Model 2 performs equally as well from slide deck to slide deck. This does not mean quality, but it does mean consistency.

Class	Avg F1 Score	F1 Score Variance	Aggregate F1
CS-0007	0.325	0.0138	0.4202
CS-0449	0.453	0.0179	0.5070
CS-1622	0.542	0.0164	0.5975

Table 2: Model 2 F1 Scores

We found no significant predictors in the Model 2 pipeline using the stepwise approach. We theorize this is due to a lack of data (i.e., full sets of `.pptx` and `.txt` files). Given sufficient data, we would anticipate a trend emerging, but with our limited data, we interpret the null result as a lack of statistical power rather than evidence that the metrics are irrelevant for this pipeline.

4.4 Stepwise Regression Methodology

Motivation While each model produces per-class F1 scores and an overall average F1 score, these outputs alone do not reveal *why* performance varies across slides or classes. To systematically improve the model, we require a principled way to diagnose which measurable slide-level properties meaningfully influence F1. The goal of this analysis is to identify such explanatory variables and use them to guide future model development. A stepwise regression framework provides a structured, statistically grounded approach for isolating these effects.

Method We began by theorizing which slide characteristics might plausibly relate to F1 performance. Candidate explanatory variables included linguistic structure metrics, redundancy measures, line-length distributions, entropy-based features, and several text-quality indicators. This candidate set was intentionally broad: the stepwise algorithm would later determine which variables were truly essential.

Using the per-class average F1 as the dependent variable, we applied a stepwise regression procedure. Before proceeding with the general procedure, we tested each predictor's significance in a single-variable regression. If it was not significant at the $\alpha = 0.05$ level, we removed it from the predictor set. Starting from an intercept-only model, the algorithm iteratively added or removed predictors to minimize AIC. This prevents overfitting, reduces redundant variables, and yields a minimal set of statistically meaningful features. Only predictors with statistically significant contributions ($p < .05$) were retained.

4.4.1 Stepwise Algorithm Outline

1. **Input:** Candidate predictors $\{A, B, C\}$, dependent variable = per-class average F1.
2. **Step 1:** For each predictor individually:
   (a) Fit a single-variable regression: F1 $\sim$ predictor.
   (b) If $p \geq 0.05$, remove the predictor from the candidate set.
3. **Step 2:** Initialize model M_0 with intercept only.
4. **Step 3:** Iteratively add or remove predictors to minimize AIC:
   (a) Among remaining predictors, add the one that yields the largest AIC reduction (e.g., A).
   (b) Refit model (e.g., F1 $\sim A$).
   (c) Check whether adding any remaining predictors (e.g., B or C) further lowers AIC.
   (d) Remove any predictors whose $p \geq 0.05$ if doing so improves AIC.
5. **Output:** Final model with minimal AIC and all predictors statistically significant ($p < 0.05$).

Usefulness The outcome of a stepwise regression serves as an objective diagnostic tool. It highlights which aspects of the slide text meaningfully influence LLM extraction performance—exactly the information needed to refine the upstream preprocessing or prompting pipeline.

For example, suppose a linguistic feature such as entropy is a consistently strong predictor of F1. In that case, then systematically increasing that feature in the input (without distorting the content) may serve as a model amplifier. Thus, the stepwise method acts as a feedback mechanism for iterative model improvement, grounded in statistical evidence rather than intuition.

4.4.2 Significant Metrics Identified by the Stepwise Model

After restricting the candidate feature set to individually significant variables ($p < .05$) and running AIC-based stepwise selection, the algorithm converged on three explanatory variables:

1. POS Entropy This metric is computed as the Shannon entropy of the combined token stream consisting of all words and punctuation marks:

$$H_{\text{pos}} = -\sum_{t \in T} p(t)\, \log p(t),$$

where $T = \{\text{punctuation tokens}\} \cup \{\text{word tokens}\}$. A higher value reflects greater unpredictability and diversity in surface-level linguistic structure.

- High H_{pos} indicates richer syntactic and punctuation-driven variation, which provides clearer structural cues for the LLM.
- Slides with greater token-type diversity tend to support more stable concept extraction, resulting in higher F1 scores.

2. Verbosity Penalty Verbosity penalty is defined as the mean character count per nonempty line:

$$V = \frac{1}{N} \sum_{i=1}^{N} |\ell_i|,$$

where N is the number of nonempty lines and $|\ell_i|$ denotes the character length of line i.

- Larger V reflects verbose or overly dense lines, which obscure conceptual boundaries and make key terms harder to isolate.
- Lower verbosity (smaller V) corresponds to clearer, more segmentable input for the model and is associated with higher F1 performance.

3. Slide Compression Potential Compression potential is defined as:

$$C = \frac{1}{1+R},$$

where R is the redundancy count:

$$R = \#\{\, w \in \text{unique words } : \text{ frequency}(w) > 3 \,\}.$$

Thus, C decreases as repeated filler or function-word overuse increases.

- A larger C indicates lower redundancy and greater information density—i.e., more unique content per token.
- Slides with higher compression potential exhibit cleaner signal, reduced noise, and yield more accurate extracted concepts.

Summary These three predictors—POS entropy, verbosity penalty, and compression potential—were the only variables whose joint inclusion minimized AIC. They correspond to three complementary dimensions of slide quality:

- **Linguistic richness** (POS entropy)
- **Clarity and conciseness** (verbosity penalty)
- **Information density** (compression potential)

All other candidate variables became redundant or insignificant once these were included. It is important to note that compression potential is not significant in the multiple-regression model at $\alpha = 0.05$ (but it is at $\alpha = 0.10$), so it may be optimal to move forward without this predictor.

4.4.3 Using Stepwise Regression to Improve Future Models

This pipeline provides a systematic way to uncover the sources of explainable F1 variability. Once statistically meaningful features are identified, they can be incorporated into future versions of the model to reduce variance and improve accuracy.

The iterative procedure is as follows:

1. Hypothesize potential explanatory variables for F1 variation (optionally assisted by an LLM).
2. Run stepwise regression using all hypothesized variables.
3. Retain only the predictors selected by the stepwise model.
4. Modify the pipeline to address or optimize for those factors directly.
5. Evaluate the new model's average F1 score and F1 variance.

Why reducing F1 variance improves performance Reducing variance across classes or slides alone may/may not improve average performance. However, if we identify structural properties that hinder extraction and then systematically correct them, we shift more inputs toward the "optimal" region of the feature space, improving both mean F1 and its stability.

Example: Increasing POS Entropy Our analysis found that higher POS entropy corresponds to higher F1. This suggests an experimental model amplifier:

1. Take in raw slide text.
2. Use an LLM to enrich POS diversity (e.g., clarify punctuation, adjust phrasing without altering meaning).
3. Run the standard concept extraction model.

Each proposed adjustment can be evaluated empirically as the process is iterative and modular; any ineffective modification can be easily reverted.

5 Limitations

5.1 Models

Throughout, the primary models used were OpenAI models (gpt-4.1-mini, gpt-5). Some exploration was conducted with Llama models (gemma3:4b), but minimal statistical analysis was performed. Overall, there should be more exploration with various model types. Number of parameters, size of the model, and date of release are just a few variables that should be explored further.

5.2 Prompts

As discussed, we used the prompts as shown in Appendix C, D, and E. These were prompts we iteratively generated to maximize the F1 score. As with all LLMs, further exploration with prompts is warranted.

5.3 Data Sets

All of the data sets used were based on computer science (CS) courses at the University of Pittsburgh. These courses ranged from an introductory programming course (CS-0007) to upper-level courses such as Intro to Compiler Design (CS-1622). This process should be repeated across a breadth of courses. Additionally, more datasets should be acquired and used in pipelines similar to Model 2.

6 Future Development

Several directions remain open for advancing the reliability, consistency, and overall performance of our concept extraction framework.

Model Stability via Repeated Inference. Because LLM-based outputs can vary across runs, an immediate extension is to rerun each model multiple times and summarize performance using averaged F1 scores and confidence intervals. This would help identify whether performance differences between models are meaningful or simply artifacts of generation variance.

Evaluating Model 2 with GPT-5. Our current implementation of Model 2 uses an earlier-generation LLM. Re-evaluating Model 2 with GPT-5 may adjust the performance gap that was observed between text-only and image+text pipelines. A more capable vision-language model could handle slide images with greater fidelity, potentially increasing aggregate F1 and reducing noise introduced by irrelevant visual content.

Leveraging Predictor-Driven Input Adjustments. An important avenue for future investigation is to test whether the stepwise pipeline can be used for actively shaping the model's inputs and prompts in ways that push key predictor variables toward regions associated with stronger performance. This includes modifying the prompt structure, altering the formatting or organization of slide text, or inserting intermediate transformation stages that rewrite or normalize content before extraction. If these controlled adjustments can reliably shift the input distribution toward forms that historically correspond to higher F1 scores, the pipeline could serve as a mechanism for boosting model accuracy through targeted input engineering rather than solely through model-level improvements. Evaluating this hypothesis would clarify whether strategically manipulating inputs offers a scalable path to substantial performance gains.

7 Conclusion

Performing concept extraction with LLMs has shown promise. As illustrated, for specific courses we achieved acceptable average F1 scores ($F1 > 0.700$); however, we also found that certain slide decks perform poorly. Models akin to our Model 1 have shown potential, and our results illustrate that POS entropy and verbosity are the two most important factors.

We identify that image-based models, while promising, often yield results inferior to those of a straightforward text-based model (for CS-0449, $F1 = 0.709$ for just text vs $F1 = 0.435$ for image and text). We see comparable variance in F1 scores with both models. This indicates that the model is just as confident with its results while being nowhere near as accurate. We suspect this is due to the images adding extraneous information, leading to incorrect concept marking. Implementing a more thorough image analysis method could lead to substantially better results.

In addition to what has been previously discussed, future works could test slide re-writing or formatting interventions that explicitly increase POS entropy and reduce verbosity, as suggested by our regression analysis. Reformatting slide decks could have a dramatic impact on the results of the content extraction.

Acknowledgments

We want to thank Dr. Emma Jordan (our CS-1980 instructor), Dr. Daniel Mosse (project sponsor), Dr. Nathan Ong (project co-sponsor), and Raja Krishnaswamy (general project research) for their help on this project.

References

Ying-ning Chen and Berlin Chen. 2010. Automatic key term extraction from spoken course lectures. In *IEEE Workshop on Spoken Language Technology*, pages 265–270.

Hossein Hassani and 1 others. 2022. A new method for keyphrase extraction from scientific video lectures. *Information Processing & Management*, 59(6):103077.

Enkelejda Kasneci, Kevin Sessler, and 1 others. 2023. Chatgpt for good? on opportunities and challenges of large language models in education. *TechTrends*, 67(6):128–133.

Raja Krishnaswamy. 2024. A framework for concept extraction from course presentations. Master's thesis, University of Pittsburgh.

Sara P. León and Vicente Escudero-Vilaplana. 2021. Impact of the provision of powerpoint slides on learning. *Computers & Education*, 171:104234.

Marion McCrory. 2022. Rethinking powerpoint for active learning. Technical report, Ulster University.

Ebrahim Norouzi, Sven Hertling, and Harald Sack. 2025. Conexion: Concept extraction with large language models. *arXiv preprint arXiv:2504.12915*.

Mi Song and 1 others. 2023. A survey on recent advances in keyphrase extraction: Methods and evaluation. In *Findings of EACL*.

Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S. Yu, and Qingsong Wen. 2024. Large language models for education: A survey and outlook. *arXiv preprint arXiv:2403.18105*.

A Predictors Tested

Note on Predictor Origin: Each predictor is labeled as either **[AI]** or **[Human]** to indicate its source. **[AI]** denotes features that were proposed or refined through interaction with a language model, while **[Human]** denotes features that were originally conceived and drafted independently by the author.

Concept Density (proxy) [AI]: Concentration of meaningful terms relative to the total size of the text.

Entropy–Line Length Interaction Term [AI]: Distinguishes whether entropy arises from structured variety or simply long lines.

Local Entropy Slope [AI]: Change in conceptual richness from one line to the next.

Line Compactness [AI]: Average number of tokens per line; measures the tightness of expressions.

Fragment Index [AI]: Variance in line lengths or token counts, indicating inconsistent conceptual chunking.

Information-per-Line [Human]: Amount of unique conceptual information contained in each line.

Verbosity Penalty [AI]: Penalizes long, rambling lines that blur key ideas.

Compositionality Score [AI]: Measures how clearly concepts are expressed in simple, compositional structures.

Unique Term Burstiness [AI]: Variance in line-level entropy; inconsistently distributed unique words.

Vocabulary Entropy [Human]: Variation in how unique terms appear throughout the entire slide.

Headword Concentration [AI]: The degree to which core conceptual nouns dominate the slide.

Lexical Novelty Score [AI]: Fraction of words appearing only once.

Definition Frequency [AI]: Count of definitional patterns ("X is…", "We call Y…", "Z, also known as…").

Enumerative Structure Index [AI]: Presence of lists, numbering, bulleting, or a hierarchical structure.

Line-Break Purity [AI]: Whether lines end at natural syntactic boundaries.

Blank-Line Segment Clarity [AI]: How well blank lines separate conceptual blocks.

Discourse Mode Classification [AI]: Purpose of the slide (definitions, examples, proofs, summary, explanations).

Definition-to-Example Ratio [Human]: Balance between definitional content and examples.

Referential Ambiguity [AI]: Frequency of ambiguous pronouns (this, that, it).

Abstraction Depth [AI]: Density of abstract or theorem-like language.

Symbol Density [AI]: Proportion of mathematical or notation symbols.

Greek-Letter Ratio [AI]: Frequency of Greek letters, signaling formal notations.

Formal-Structure Token Ratio [AI]: Occurrence of structured automata-like tokens (ϵ, δ, σ, $\sum$, q_{accept}, q_{reject}, q_n).

POS Entropy [AI]: Diversity in part-of-speech patterns.

Syntactic Complexity Score [AI]: Number of clauses or syntactic embeddings.

Parse Tree Depth Variance [AI]: Variation in syntactic depth across lines.

Semantic Jumpiness [AI]: Variance in semantic similarity between consecutive lines.

Interline Semantic Difference [AI]: Average conceptual shift across adjacent lines.

Slide Topical Focus Score [AI]: Topic consistency across the slide.

Concept-Coherence Score [AI]: Semantic similarity of key noun phrases.

Definition Boundary Clarity [AI]: Clarity of conceptual definition boundaries.

Slide Compression Potential [AI]: How well the slide compresses without losing its meaning.

Redundancy Score [Human]: Extent of repeated or filler content.

Information Flow Sharpness [AI]: Sharpness of new information appearing across lines.

B Data

Class	Avg F1 Score	F1 Score Variance	Aggregate F1
CS-0007	0.750	0.0101	0.7650
CS-0441	0.572	0.0305	0.7356
CS-0447	0.721	0.0094	0.7562
CS-0449	0.709	0.0090	0.7528
CS-1502	0.667	0.0301	0.7601
CS-1541	0.667	0.0113	0.7272
CS-1550	0.660	0.0389	0.7745
CS-1622	0.702	0.0121	0.7592

Table 3: Model 1 F1 Scores

Variable	Coef.	SE	t	p
Intercept	-0.287	0.185	-0.155	0.877
POS Entropy	0.162	0.035	4.586	9.24e-6
Verbosity Penalty	-0.007	0.002	-3.332	0.001
Compression Potential	-0.338	0.203	-1.664	0.098

Table 4: Regression coefficients for Model 1 predicting F1 (Coefficient, standard error, t value, p value).

Class	Avg F1 Score	F1 Score Variance	Aggregate F1
CS-0007	0.325	0.0138	0.4202
CS-0449	0.453	0.0179	0.5070
CS-1622	0.542	0.0164	0.5975

Table 5: Model 2 F1 Scores

C Model 1 Prompt

Below is the full system prompt used for Model 1. This model was responsible for extracting only *explicitly defined instructional concepts* directly from the slide text, with no inferred or contextual additions.

C.1 Prompt Generation Process

The process of designing the Model 1 prompt involved several structured stages, each aimed at progressively improving alignment between the model's concept annotations and the human gold-label set. The overall goal was to translate the subjective process of concept identification into a reproducible, rule-based annotation procedure that could be implemented algorithmically.

1–2. Codebook Generation and Inclusion / Output Requirements The first step involved writing a comprehensive *concept extraction codebook* that specified explicit algorithmic rules governing inclusion and exclusion criteria for concepts. The purpose of this codebook was to transform what is typically a subjective linguistic task—deciding which words or phrases constitute instructional concepts—into a clear and rule-based process. Each rule was designed to be interpretable and enforceable, ensuring that two independent annotators (human or model) could, in principle, arrive at consistent results.

In conjunction with the codebook, explicit *output requirements* were established to standardize the model's format and ensure parsability. These requirements included:

- Specifying the output file type and structure (plain text, one concept per line);
- Enforcing stylistic conventions (lowercase except for acronyms);
- Providing an explicit, verbatim example of the desired output format.

Including the example output directly in the prompt significantly improved the model's adherence to the requested formatting and reduced hallucinated commentary or numbering.

3. Filtering Stage After integrating only the codebook and output requirements, the model tended to produce a much larger number of candidate concepts than were found in the human gold-label set. This indicated over-generation—an excess of marginal or loosely related terms. To address this, a secondary *filtering layer* was added to the prompt. This stage used more qualitative and subjective filters, emphasizing criteria such as conceptual salience, explicitness, and definitional grounding. The filtering rules instructed the model to discard terms that lacked direct textual justification or that merely repeated near-synonyms of existing concepts.

4. Explicit Example Inclusion The final improvement involved adding a full verbatim example of a slide deck alongside its corresponding human-annotated gold-label concepts. Providing this paired example proved essential in guiding the model toward a better conceptual understanding of what constituted a "valid concept." The explicit, human-annotated example served as a grounding reference, improving the model's consistency and precision when applied to unseen slides.

Things Tried but Didn't Work

Several alternate approaches were explored but did not yield improvements in performance or alignment with the gold-label set:

- **Reinforcing meta-instructions ("follow the rules strictly"):** Adding redundant self-reminders and reinforcement cues (e.g., "double-check that you follow every rule exactly") had negligible effect and occasionally caused repetition or boilerplate disclaimers in the output.
- **Overly fine-grained rule sets:** Expanding the codebook to more than ten rules reduced coherence. The model often latched onto low-priority conditions while ignoring higher-level inclusion criteria. From empirical testing, it appears that the usefulness of the code book is to extract the first layer of candidate concepts.

Future iterations could test hybrid strategies combining structured rule-based guidance with light in-context demonstrations to balance precision and generalization.

Role

The annotator functions as an expert academic extractor focusing strictly on concepts that are textually supported within the provided slides. No inference beyond the text is permitted; all outputs must be evidence-based and minimal.

Input Variables

- `{slides text}` — the complete textual content of the PowerPoint slide deck.

Full System Prompt

```
### ROLE ###
You are an expert academic annotator specializing in *concept extraction* from university lecture
slides.

Your goal is to identify and list only the key **concepts** that are *explicitly* defined, emphasized,
or used in examples within the provided slide text.

Do NOT infer concepts beyond the slide text. Your task is purely textual and evidence-based.

---

### CODEBOOK (Condensed from “Concept Annotation Codebook – Refined, Oct 2025”) ###

1. **Slide-Deck Grounding Rule** — Only annotate concepts that are justified or defined in the given
slide text. Do not infer from outside knowledge.

2. **Granularity Rule** — Annotate at the most specific level that adds distinct meaning.
   - If splitting a phrase adds no new meaning, keep it as one concept.
   - If the parts are meaningful alone, include both the whole and its parts.

3. **Definition & Emphasis Rule** — Label any word or phrase that is:
   - Clearly *defined or described*,
   - *Emphasized* as a key term, or
   - Used as the *subject* of an example or explanation.

4. **Abbreviation Rule** — Always include abbreviations *and* their expanded forms on separate lines.

5. **Non-Conceptual Modifiers Rule** — Do **not** include syntax, keywords, or modifiers unless the
slide explicitly defines or discusses them as concepts (e.g., “static”, “visibility”).

6. **Adjoinment Rule** — Include multi-word terms when they represent a unified idea (e.g., “function
header”, “process control block”).

7. **Generic Term Rule** — Do not label generic or context-free terms (e.g., “thing”, “object”) unless
defined specifically.

---

### FILTERING STEP ###
After identifying possible concepts, remove any that are **not explicitly**:
- defined,
- emphasized, or
- used in an example or explanation.

Only keep the *most important instructional concepts*.

---

### OUTPUT REQUIREMENTS ###
- Output **each concept on a new line**.
- Do **not** number them.
- Do **not** include reasoning, commentary, or explanations.
- Include abbreviations *and* their full forms as separate lines.
- Keep consistent capitalization (use lowercase unless the concept is an acronym).
- Do not include REPEATED COURSE NAME as a concept.

Example Output:

function
function name
function body
function header
parameters
return type

EXAMPLE ANNOTATION:

###provided full cs0007/Lecture 7.txt (available for viewing in the github repository)###

(ENDING CONCEPTS)
```

```
Function
Functions
Parameter
Parameters
function body
return
Returns
function header
function name
header
return type
scope
passing-in
pass-by-value
pass-by-reference
void
variable
variables
Primitive
Public
```

D Model 2 Screenshot Prompt

Below is the system prompt:

```
You are a precise annotator. Always respond in json only. Return a single, well-formed JSON object
that matches the requested schema. Do not include any extra commentary or text.
```

Note: "deck name", "slide indices", and "concept gold represent the slide deck name, the indices that are currently being fed into the LLM, and a list of gold, human-labeled concepts, respectively. "Below is the full prompt used:

```
You are an expert slide-deck annotator. Your goal is to look at screenshots of the images and create
sentences to describe ALL of the key concepts in the slide. This includes looking at the text and
processing meaning in addition to looking at any visuals on the slideshow.Again, the goal of this is
to get at least 1 sentence per key concept.You should not write anything about the course number, ie
CS-1234 or the professor. Assume the reader of your sentences is taking the class for the first time
and is just looking for brief summaries of the concepts from the lecture. You should ensure that all
key concepts from the slidedeck are included in at least one sentence. Multiple sentences can be about
the same topic, just ensure that every concept is mentioned somewhere in that chunk of slide. Here is
an example for the type of sentences that could be useful. These are by no means the only sentences
that you could derive from these slides, but these are a start. The slidedeck name is {deck name}. The
slides in this batch are {slide_indices}. After you create the sentences, we will run another model
that pulls the concepts from the sentences. Here were some examples from the gold label concepts from
this slidedeck {gold_concepts}. For the output, output the result strictly as JSON (with no extra
text) in the following schema:

{
    "deck": "<deck name>",
    "batch slides": [<slide numbers>],
    "per_slide\": { "<slide number>": "["sentence 1", "sentence 2", ...]", ... },
    "sentences: ["slide-tagged sentences flattened in order"],
}
```

E Model 2 Concept Generation Prompt

Below is the system prompt used:

```
You are a helpful assistant that follows the user instructions exactly.
```

Note: Here, "deck name", "slides text", and "sentences" are the slide deck name or title, the full text from the slide deck, and the full list of sentences summarizing the slide deck, respectively.

```
You are an expert academic annotator specializing in concept extraction from university lecture
slides. Identify and list only the key concepts that are explicitly defined, emphasized, or used in
examples within the provided datasets. These datasets consist of a full transcript of the PowerPoint
along with sentences summarizing the PowerPoint. Place a 90 percent weight on the text and 10 percent
on the summaries. If unsure, exclude the term (prioritize precision over recall). Again, your goal is
to produce the most concise and minimal list of explicitly defined concepts. Do not include
near-synonyms, plural variants, or implementation details unless the slide clearly defines them as
separate terms. If you are uncertain, omit the term. If multiple variants of a term appear (e.g.,
“override”, “overriding”), keep only the canonical or dictionary form.
```

Slide-Deck Grounding Rule — Do not infer concepts not textually supported, but include any concept that the slide defines, titles, or repeatedly references. You should be able to point to a particular slide and say this is where this concept came from.

Granularity Rule — Annotate at the most specific meaningful level. If a concept phrase (“superclass constructor”) is only an instance or elaboration of a broader concept (“constructor”), include only the broader term unless the slide defines the specific variant."

Definition & Emphasis Rule — Label words or phrases that are: explicitly defined (“X is a . . . ”), visually emphasized (bold/italic/title-case), or used as key examples.

Explicit evidence includes patterns such as “X is a Y”, “X means . . . ”, bold headings, or definition bullets.

Abbreviation Rule — Include both abbreviation and full form only if both appear.

Normalization Rule — Output each concept once, in singular form. Use lowercase unless the concept is an acronym. Do not repeat plural or synonymous variants (e.g., “method/methods”, “class/classes”).

Implementation Filter — Exclude code-implementation details (field, property, variable, getter, setter, overloaded constructor, superclass constructor) unless explicitly defined as conceptual terms.

Relation Boundary Rule — Include relational concepts only if the term itself is introduced or defined (e.g., “Inheritance = relationship between superclass and subclass”). Do not list relational variants (“parent class”, “child class”, “subtype”, “inter-class relationship”) unless defined verbatim.

Keyword Filter — Exclude language keywords or visibility specifiers (public, private, static, final) unless defined as concepts.

Semantic Boundary Rule — Ignore meta-instructional phrases or general advice (e.g., “modeling a problem”, “find the nouns”).

Core Concept Retention Rule — Always include foundational terms that appear in definitions, titles, or repeated headers (e.g., “program”, “memory”, “class”, “inheritance”). If a slide introduces a section or example with such a term, treat it as a core concept even if not fully defined.

Precision Emphasis Rule — When uncertain, omit rather than guess.

Self-Check Step — After extraction, remove any concept not clearly supported by definition, emphasis, or example.

IMPORTANT OUTPUT FORMAT: Return ONLY the concept phrases, one per line, with NO bullets, NO hyphens, NO numbers, and NO extra commentary before or after the list. Concepts are to be no more than 4–5 words. Output the concepts as plain text, one concept per line.

=====================================

FILTERING STEP:
After identifying possible concepts, remove any that are **not explicitly** defined, emphasized, or used in an example or explination.

OUTPUT REQUIREMENTS
- Extract all concepts from the sentences
- Do **not** number them.
- Do **not** include reasoning, commentary, or explanations.
- Include abbreviations *and* their full forms as separate lines.
- Keep consistent capitalization (use lowercase unless the concept is an acronym).
- Do not include REPEATED COURSE NAME as a concept.

Self-Check Step (Precision Pass):
Before finalizing, review your list and remove any term that:
is not directly defined, emphasized, or exemplified; duplicates or pluralizes another item; or is a code keyword, modifier, or descriptive relationship rather than a conceptual noun phrase.

You will be annotating {deck name}
And here is the text from the powerpoint:
{slides text}

And here is the sentences summarizing it:
{sentences}

After going pulling the concepts, go back through the concepts list and compare what you have to the sentences and powerpoint to ensure that it meets the criteria for a concept.