CMSC 398z
Effective use of AI Coding Assistants and Agents

Bill Pugh and Derek Willis

Nov 7th, 2025

Plan for today

Tips at using Claude

Using Gen AI for understand code

Reviewing work on Congressional Review

Not planning on any coding in class today

But there will be discussion and requests for presentations from student teams

Reading for next week is important

Material that the AI & CS education committee will be discussing Nov 17th

All CS faculty will be invited to that meeting

- I'd love to invite you all but don't think I can swing that
- But I can present your thoughts on the the material

There will be a survey, working on questions for everyone to answer, but will also be free response questions on each reading

- Survey posted by Tuesday

Readings next week

- What do professional software developers need to know to succeed in an age of Artificial Intelligence? FSE 2025
  - Notebook LLM summary - with audio and presentation summary

- UW's messaging about AI & CS Education
  - Web page on "Why an Allen School degree continues to be a great choice for students"
  - News story "Coding is dead: UW computer science program rethinks curriculum for the AI era"

- Rajeev Kaul's post - Software Engineers Should Be Excited, Not Anxious, About AI
  - Rajeev is a UMD CS alumnus

GenAI in teaching and learning.
3 statements:

"Generative AI is changing the way I teach/learn, and represents an enormous opportunity to rethink how we do teaching and learning at Cornell…"

"Generative AI is the single biggest threat to teaching and learning at Cornell [or 'in discipline/college X'] in many decades…

"Generative AI as it currently operates is a bland tool that produces mediocre results and has had little effect on learning [or teaching] in my field.'

Using Claude CLI well

You need a pro account to use it for anything more than a toy example

You can end a claude session (ctrl-D) and resume it with claude -c

You can resume any previous claude session from the past month

You can use ctrl-G to edit prompt in IDE

claude -h will list command line options

claude doctor will check your installation, see if updates are available

claude recently moved from an npm implementation to native

claude update will check for updates

Don't overstuff your context

Using /context will tell you how much of your context window is used up

- If you get close to your limit, claude with auto compact your context
- Having it produce a summary document is better

If you fix one issue, before moving on to an issue that isn't closely related, have it write a summary, and then use /clear or end the session and start a new one.

You can also ask it to read all of the uncommitted changes in your workspace, or all the changes since a particular revision.

Understanding usage and context

You can use /usage to get a live plot of how much of your claude usage you have and when it will reset

- You can keep this running this in an extra window if you want

Creating your own / commands

Create the .claude/commands/ directory in your home directory or project directory

Create a markdown file in that directory. For example, catchup.md

Put the instructions you want to have executed in that file

Use /*name* to invoke the command (e.g., /catchup)

Claude conversions

stored in ~/.claude/projects - one folder per project

Nice tool: https://github.com/ZeroSumQuant/claude-conversation-extractor

can use uv tool install claude-conversation-extractor, can extract to markdown, html or pdf

claude-extract --list

claude-extract --recent 5

claude-extract --search "python error"

Claude overload

Sometimes, claude just gets overloaded

Most of the time, the issue resolves in 10-15 minutes

Miscellaneous notes

I had to add the following to CLAUDE.md

- When the user asks a questions, don't continually start your response with "Good Question!" or some variation. It gets tiring.

Claude is great for answering questions about data. I downloaded many years worth of data, several hundred thousand documents

Your thoughts on using claude

Moving on to congressional-record

Well contained issues

- Issue 55 - Cannot scrape January 3 records from 2012-2025
- Issue 51 - application crashing when downloading specific days

Issues with speakers

- Issue 36 - speech tagged as title
- Missing speaker_bioguide for speech

Bigger issue

Issue 10 - Things that are not getting parsed but could be if desirable

Issue 55 - Cannot scrape January 3 records from 2012-2025

The scraper and parser are working great overall, but we've run into the issue that it cannot extract any records on the date January 3 (the official first session of each year) from the past 14 years. This is likely because January 3 typically includes two Volumes as opposed to the norm of one.

Great catch. Would you be interested in writing a patch yourself and sending in a PR? It might be quicker for me to help you do it (eg pointing to where in the code the issues are likely to be) rather than wait for me to handle it all on my own.

My analysis

The downloader doesn't have a directory of what files can be downloaded. It just has a pattern it uses to download zip files for a day.

On some days, no congressional record is published

The downloader can also have problems downloading the zip file even if one exists

Issue report mentioned something special about Jan 3rd: two different congresses meet on that day.

Understanding what is available

There is a web page that tells you what zip files can be downloaded

On January 3rd, two different volumes are published

Every Jan 3rd, not just at the transition between the old congress and a new one

A new volume of the congressional record starts each Jan 3rd

Congress doesn't always meet on Jan 3rd, so on some Jan 3rds, neither volume is available

I asked Claude to parse the web page, but it is all Javascript, and being able to get that data via python requires an API Key

Solving Issue 55

So on Jan 3rd, we can just calculate the names for the two zip files and try to download them

Optional extensions:

- It might be better for the top level organization of the output directory to be by volume rather than by year
  - But this would really mess up existing workflows
- If the user had a GovInfo API key, we could use that to make downloads more efficient

Issue 51 code crashing on specific days

Code crashes with AttributeError: 'NoneType' object has no attribute 'string

Occurs on several specific days:

- 1999-01-16
- 2001-01-06
- 2019-05-14
- 2021-10-25

My analysis

Two different problems:

- app crashes, doesn't specify date on which parsing failed, doesn't continue
- the problem that caused the code to crash
    - type="parsed" is missing from congmember elements,

Fixing the parsing problem

Is the parsed field missing because of a mistake in our code, or it is a problem in what gets downloaded in the zip files?

It is a problem in what gets downloaded.

congmember has other name elements (authority-lnf, authority-fnf and authority-other)

Parser fix

Try to find name with fallback hierarchy

1. Prefer type="parsed" (name as it appears in Congressional Record)
2. Fall back to type="authority-lnf" (Last, First format)
3. Fall back to type="authority-fnf" (First Last format)
4. Fall back to type="authority-other" (other official name)
5. Emit warning error log entry, skip entry

Log any entry that isn't parsed

Fixing the rude crash

Crashing and not telling you the file or at least the date that caused the crash is just rude

Might want to change the code to log a crash while parsing a file and continue onto the next day

Issue 36 - speech tagged as title

The parser mistakenly assigned a speech as a title.

To replicate this behavior, parse CREC-1997-01-28-pt1-PgS771-3.

A bug fix and a new test are both required.

The bug fix should address what is (I think) an issue with the regular expression currently used to find titles.

The test should assert there aren't extensive lower or normally-cased strings in things tagged as titles.

Let's look at the HTML - using ␣ `for space`

<body><pre>
[Congressional␣Record␣Volume␣143,␣Number␣8␣(Tuesday,␣January␣28,␣1997)]
[Senate]
[Page␣S771]
From␣the␣Congressional␣Record␣Online␣through␣the␣Government␣Publishing␣Office␣
[<a␣href="https://www.gpo.gov">www.gpo.gov]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ADDITIONAL␣STATEMENTS ***- this is the Title***
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣______
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
␣␣LAWRENCE␣B.␣LINDSEY'S␣DEPARTURE␣FROM␣THE␣BOARD␣OF␣GOVERNORS␣OF␣THE
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣FEDERAL␣RESERVE ***- this is the doc title***

␣␣<bullet>␣Mr.␣ABRAHAM.␣Mr.␣President,␣I␣rise␣today␣to␣note␣the

Look at the regular expression used for matching speaker

```
   re_speakers = (
        r"^(\s{1,2}|<bullet>)(?P<name>(("
        + speaker_list
        + ")|(((Mr)|(Ms)|(Mrs)|(Miss))\. (([-A-Z'])(\s)?)+( of [A-Z][a-z]+)?)|(((The
((VICE|ACTING|Acting) )?(PRESIDENT|SPEAKER|CHAIR(MAN)?)( pro tempore)?)|(The PRESIDING
OFFICER)|(The CLERK)|(The CHIEF JUSTICE)|(The VICE PRESIDENT)|(Mr\. Counsel [A-Z]+))(
\([A-Za-z.\- ]+\))?)))\.")
else:
    re_speakers = r"^(\s{1,2}|<bullet>)(?P<name>((((Mr)|(Ms)|(Mrs)|(Miss))\.
(([-A-Z\'])(\s)?)+( of [A-Z][a-z]+)?)|((The ((VICE|ACTING|Acting)
)?(PRESIDENT|SPEAKER|CHAIR(MAN)?)( pro tempore)?)|(The PRESIDING OFFICER)|(The
CLERK)|(The CHIEF JUSTICE)|(The VICE PRESIDENT)|(Mr\. Counsel [A-Z]+))( \([A-Za-z.\-
]+\))?))\."
```

Understanding what the code does

It looks for a kind

```
kind: str = Field(
        ...,
        description=(
            "Type of content item: 'speech', 'recorder', 'clerk', 'linebreak', "
            "'excerpt', 'rollcall', 'metacharacters', 'empty_line', 'title', or 'Unknown'"
        ),
    )
```

Most kinds will start a new content_item; metecharacters and empty line are the exception

The pattern for speaker

Pattern looks for a line that

- starts with 1 or 2 spaces or a <bullet>, followed by one of:
- The name of one of the known speakers
- A pattern that should match any name
- A procedural name (e.g., The PRESIDING OFFICER)

What is wrong with this?

Repetition, keeping things in sync, syntax warning

Having a long complicated regular expression is bad enough

Having two copies of the regular expression that need to be kept in sync is worse

We were getting syntax warnings because some of the strings needed to be marked as raw strings, by starting the strings with r"

What is wrong with this

In this case, the HTML line starts with 2 spaces, <bullet>, a space, and then a speaker name

This doesn't match 1-2 spaces or a bullet

Claude suggested changing it to 1-2 spaces followed by a bullet

But that wouldn't match either, because there is a space between bullet and the speaker's name

digging in deeper

Where does bullet appear in the html?

I asked Claude to identify all the spaces where a <bullet> appeared in the html, and see how those were classified.

Trying to explore further, Claude found that the regular expression was being used on pre text that had been turned into plain text by BeautifulSoap, and there was never a <bullet> in the text being examined.

Instead, the two spaces before the bullet and the space after the bullet were turning into 3 spaces

Claude suggested changing the pattern to searching for 1-3 spaces

Is 1-3 spaces the right answer?

I asked Claude to examine all the places where a line started with any number of spaces followed by a speaker name, and it verified that 1-3 characters seemed to correctly identify all the cases where the name was being used to indicate a person speaking.

> Seems like a hack, but we can roll with it

Let's look at that regular expression again

```
   re_speakers = (
        r"^(\s{1,2}|<bullet>)(?P<name>(("
        + speaker_list - people we have a speaker_bioguide for
        + ")|(((Mr)|(Ms)|(Mrs)|(Miss))\. (([-A-Z'])(\s)?)+( of [A-Z][a-z]+)?)|(((The
((VICE|ACTING|Acting) )?(PRESIDENT|SPEAKER|CHAIR(MAN)?)( pro tempore)?)|(The PRESIDING
OFFICER)|(The CLERK)|(The CHIEF JUSTICE)|(The VICE PRESIDENT)|(Mr\. Counsel [A-Z]+))(
\([A-Za-z.\- ]+\))?)))\.")
else:
    re_speakers = r"^(\s{1,2}|<bullet>)(?P<name>((((Mr)|(Ms)|(Mrs)|(Miss))\.
(([-A-Z\'])(\s)?)+( of [A-Z][a-z]+)?)|((The ((VICE|ACTING|Acting)
)?(PRESIDENT|SPEAKER|CHAIR(MAN)?)( pro tempore)?)|(The PRESIDING OFFICER)|(The
CLERK)|(The CHIEF JUSTICE)|(The VICE PRESIDENT)|(Mr\. Counsel [A-Z]+))( \([A-Za-z.\-
]+\))?))\."
```

Procedural names

Many of the speaker names are procedural names

Such speakers do not have speaker_bioguide

Not a bug in the code

But should we separate out procedural_speaker from speaker as two different kinds?

Might make it easier to look for certain kinds of entries

But might break existing workflows

Other potential issues

The pattern for matching a known potential speaker should be covered by the pattern that matches arbitrary names

- Claude thinks the use of explicit names makes pattern matching more efficient, but would it matter?

Is the pattern for arbitrary names sufficient? Does a speaker ever have Rev, Dr. or Prof. in their name?

Is assumes that Counsel is always preceded by Mr.

-

Issues

Does adding the known potential speaker names actually improve matching?

Seems to be an assumption that a Counsel is always prefixed by Mr.

- Only comes up in impeachment proceedings, all instances male for 2020-2021 impeachment proceedings

All of the procedural names are a bit of a hack – is this a complete list?

- They never have speaker bioguides

repetition in regular expression

Complaints about syntax error in strings

- need to use r" xxxxx " to indicate raw string

Where do speaker names come from

1. **Source**: mods.xml files downloaded from GovInfo (congressionalrecord/govinfo/cr_parser.py:269-272)
2. **Extraction**: find_people() method (cr_parser.py:157-195) parses <congmember> tags from mods.xml
3. **Name used**: The type="parsed" name attribute (line 166), which contains the speaker name **as it appears in the Congressional Record**: None of the examples match the fallback regular expression
   - Mr. DeFAZIO
   - Mr. WILSON of South Carolina
   - Mrs. CAROLYN B. MALONEY of New York
   - Mr. SEAN PATRICK MALONEY of New York
4. **Regex generation**: make_re_newspeaker() (line 118-134) builds a regex that includes ALL these parsed names from metadata as alternates: (name1|name2|name3|...)
5. **Bioguide assignment**: Each <congmember> tag includes a bioguideid attribute (e.g., bioguideid="D000191"), which is extracted by people_helper() (line 138-143)

Fallback RE for speaker names

We need the explicit names for matching speakers

Unclear if the fallback RE is ever actually useful

Asked claude to check

From 100,000 JSON files:
- 653 distinct speakers without bioguide IDs
- 587 are procedural speakers (The SPEAKER, etc.)
- **66 are matched by the fallback regex**

Those 66 matches are almost entirely:
- **Misspellings**: CICILLLINE, CLYBRUN, CAARDENAS
- **Incomplete names**: "Mr. BRENDAN F", "Mr. C", "Mrs. CAROLYN B"
- **Typos**: "Mr. CHAFFETZ-", "Mr. CONNOLLY " (trailing characters)
- **State errors**: "of Tennesee", "of George"

We could try to match those to existing speakers to pick up bioguides, but only 132 occurrences for 100,000 files

So, dealing with speaker names

Change the regex to expect it to start with 1-3 spaces, rather than 1-2 space or a <bullet>

Classify a section starting with a procedural speakers as procedural_speech, rather than speech – breaking change

For the Mr. Counsel pattern, perhaps this should be kept as speech

- such people never have bioguides
- perhaps generalize to allow for Ms., etc

For speakers that match fallback pattern, perhaps look for a near match in speaker list – hard to do precisely

Issue 10 - Things that are not getting parsed but could be if desirable

The following could, theoretically, be added to the Congressional Record parser:

Foreign travel expenditure reports

Congressional authority statements

Any feedback on whether these would be worth doing? Would there be a better source for this information? Are there any other additional documents in the Congressional Record that would be useful to parse?

Current version of the parser is pretty good at identifying speeches but can't pick up on much else. There's a probabilistic test in the test suite that fails and should continue to fail until all text in each record page is correctly parsed.

CREC-2025-10-21-pt1-PgH4545-7.htm

From the Congressional Record Online through the Government Publishing Office [www.gpo.gov]

EXPENDITURE REPORTS CONCERNING OFFICIAL FOREIGN TRAVEL

Reports concerning the foreign currencies and U.S. dollars utilized for Official Foreign Travel during the third quarter of 2025, pursuant to Public Law 95-384, are as follows:

jjj

REPORT OF EXPENDITURES FOR OFFICIAL FOREIGN TRAVEL, DELEGATION TO CANADA, EXPENDED BETWEEN SEPT. 4 AND SEPT. 5, 2025

Name of Member or employee	Date		Per diem \1\		Transportation		Other purposes		Total	
	Arrival	Departure	Country	Foreign currency	U.S. dollar equivalent or U.S. currency	Foreign currency	U.S. dollar equivalent or U.S. currency	Foreign currency	U.S. dollar equivalent or U.S. currency	Foreign currency	U.S. dollar equivalent or U.S. currency
Speaker Mike Johnson.....	9/4	9/5	Canada.....	.....	407.00	(\3\)	.				

CREC-2025-10-21-pt1-PgH4545-7.json

"content": [
{
"kind": "Unknown",
"speaker": "Unknown",
"text": " Reports concerning the foreign currencies and U.S. dollars utilized \nfor Official Foreign Travel during the third quarter of 2025, pursuant \nto Public Law 95-384, are as follows:\n",
"turn": -1,
"itemno": 0
},
{
"kind": "title",
"speaker": "None",
"text": " REPORT OF EXPENDITURES FOR OFFICIAL FOREIGN TRAVEL, DELEGATION TO CANADA, EXPENDED BETWEEN SEPT. 4 AND SEPT. 5, 2025\n----------\n Date Per diem \\1\\ Transportation Other purposes Total",
"turn": -1,
"speaker_bioguide": null,
"itemno": 1
},
{
"kind": "linebreak",
"speaker": "None",
"text": " ---------- ----------\n U.S. dollar U.S. dollar U.S. dollar U.S. dollar\n Name of Member or employee Country Foreign equivalent Foreign equivalent Foreign equivalent Foreign equivalent\n Arrival Departure currency or U.S. currency or U.S. currency or U.S. currency or U.S.\n currency currency currency currency\n----------\\2\\----------\\2\\----------\\2\\----------\\2\\----\nSpeaker Mike Johnson.....
9/4 9/5 C

CREC-2025-10-28-pt1-PgH4559-2.htm

Congressional Record Volume 171, Number 179 (Tuesday, October 28, 2025)]
[House]
[Page H4559]
From the Congressional Record Online through the Government Publishing Office [www.gpo.gov]

By Mr. BRESNAHAN:

H.R. 5836.

Congress has the power to enact this legislation pursuant to the following:

Article I, Section 8 of the U.S. Constitution

</pre></body>
</html>

CREC-2025-10-28-pt1-PgH4559-2.json


```
"content": [
  {
    "kind": "recorder",
    "speaker": "The RECORDER",
    "text": "          By Mr. BRESNAHAN:",
    "turn": -1,
    "speaker_bioguide": null,
    "itemno": 0
  },
  {
    "kind": "title",
    "speaker": "None",
    "text": "      H.R. 5836.\n      Congress has the power to enact this legislation pursuant \n    to the following:\n      Article I, Section 8 of the U.S. Constitution\n\n",
    "turn": -1,
    "speaker_bioguide": null,
    "itemno": 1
  }
],
"related_bills": [
  {
    "congress": "119",
    "context": "TITLE",
    "number": "5836",
    "type": "HR"
  }
],
```

Looking at kinds

Overall Kind Distribution (150-day sample):

- `title`: 65,901 (36.5%)
- `speech`: 60,643 (33.6%)
- `linebreak`: 27,720 (15.4%)
- `recorder`: 18,143 (10.0%)
- `Unknown`: 7,500 (4.2%)
- `metacharacters`: 746 (0.4%)
- `clerk`: 539 (0.3%)

More generally, understanding kinds

- speech - starts with speaker name
- linebreak - dashed line, possible with NOTE or END NOTE in the middle
- title - All caps, starting with letter, possibly preceded by dashed line
- recorder - various speech patterns suggesting a recorder roll

Lots of titles

Problem: Large numbers of consecutive `title` classifications, often hundreds or thousands in a single document.
Statistics (150-day sample):
- **2,274 documents** have consecutive title sequences
- **Max consecutive:** 3,586 titles in the sample
- **Average when it occurs:** 17.8 consecutive titles
Statistics (Full dataset - 301k files):
- **721 documents** with 100+ consecutive titles
- **Max consecutive:** 8,414 titles in a single document
- Extreme cases all involve bill/amendment text

Kind Bigrams (2-item sequences):

1. `('title', 'title')`: 50,706 occurrences ⚠️ **Most common bigram**
2. `('speech', 'speech')`: 39,900 occurrences ✓ Expected
3. `('speech', 'linebreak')`: 12,456 occurrences
4. `('title', 'speech')`: 7,913 occurrences
5. `('speech', 'recorder')`: 7,385 occurrences
6. `('linebreak', 'title')`: 7,255 occurrences
7. `('recorder', 'recorder')`: 6,024 occurrences ⚠️ Bill introduction lists
8. `('recorder', 'speech')`: 5,343 occurrences
9. `('recorder', 'title')`: 5,038 occurrences
10. `('title', 'linebreak')`: 4,120 occurrences

Kind Trigrams

Trigrams (3-item sequences):
1. `('title', 'title', 'title')`: 46,607 occurrences ⚠️ **Most common trigram**
2. `('speech', 'speech', 'speech')`: 32,028 occurrences ✓ Expected
3. `('title', 'speech', 'linebreak')`: 5,999 occurrences
4. `('linebreak', 'title', 'speech')`: 5,799 occurrences
5. `('speech', 'speech', 'recorder')`: 5,370 occurrences
6. `('recorder', 'recorder', 'recorder')`: 4,787 occurrences ⚠️ Bill introduction lists
7. `('speech', 'recorder', 'speech')`: 4,417 occurrences
8. `('recorder', 'speech', 'speech')`: 3,801 occurrences
9. `('speech', 'speech', 'linebreak')`: 2,334 occurrences
10. `('title', 'title', 'linebreak')`: 2,288 occurrences

Parser primarily targets speeches

Current parsing of content items isn't particularly useful for these types

How do we identify them?

Look at doc_title

CREC-2025-10-21-pt1-PgH4545-7.json

"doc_title": "EXPENDITURE REPORTS CONCERNING OFFICIAL FOREIGN TRAVEL",
"title": "EXPENDITURE REPORTS CONCERNING OFFICIAL FOREIGN TRAVEL"

CREC-2025-10-28-pt1-PgH4559-2.json

"doc_title": "Constitutional Authority Statement for H.R. 5836",
"title": null

But what else can we catch?

Analysis of 111,706 documents starting with "Unknown" kind reveals that 49.4% (55,179 documents) contain ONLY linebreaks and/or titles after the Unknown item. This suggests nearly half of these documents may not benefit from the current content-item breakdown.

Going down the rabbit hole here. I think tackling these should probably be part of a 3.0 release of congressional-record

Key Finding: Structure Distribution

Content Structure (after Unknown)	Count	Percentage	Interpretation
only_linebreaks	31,728	28.4%	Just preamble + linebreak separator
speeches_and_linebreaks	24,946	22.3%	One-minute speeches (substantive)
only_linebreaks_and_titles	21,334	19.1%	**Tables, lists, schedules broken into pieces**
speeches_with_other	6,997	6.3%	Complex debates (substantive)
speeches_with_linebreaks_titles	6,463	5.8%	Speeches with structure (substantive)
recorders_and_linebreaks	5,355	4.8%	Bill lists (substantive)
only_unknown	5,101	4.6%	Just the preamble, nothing else
only_speeches	4,571	4.1%	Pure debate (substantive)
recorders_with_linebreaks_titles	2,937	2.6%	Bill lists with structure
only_titles	2,117	1.9%	**Lists broken into title items**
only_recorders	152	0.1%	Pure recorder lists

Top Document Types with linebreak/title Only

Document Title	Count	Why This Matters
PRAYER	2,966	Prayer text broken into preamble + linebreak
EXECUTIVE COMMUNICATIONS, ETC.	1,715	Communication lists
ADDITIONAL COSPONSORS	1,527	Sponsor lists as linebreak/title
PLEDGE OF ALLEGIANCE	1,367	Pledge text as preamble + linebreak
REPORTS OF COMMITTEES...	1,239	Committee reports broken down
AMENDMENTS SUBMITTED...	988	Amendment lists
NOMINATIONS	761	Nomination lists as titles
SENATE COMMITTEE MEETINGS	684	**Meeting schedules as title list**
TEXT OF AMENDMENTS	563	**Amendment text broken into pieces**

Documents That Could Be Single Items

These patterns suggest the document content is already in the Unknown item:

- only_linebreaks (31,728): Unknown text + separator linebreak
- only_unknown (5,101): Just Unknown text, no follow-on content

Examples:

- PRAYER (2,966 docs): Unknown contains full prayer
- EXECUTIVE COMMUNICATIONS (1,715 docs): Unknown contains communication list
- PLEDGE OF ALLEGIANCE (1,367 docs): Unknown contains pledge text
- Speaker designations (1,535 docs): Unknown contains designation letter

Documents Needing Better Structure Representation

These have `only_linebreaks_and_titles` or `only_titles` patterns:

- **only_linebreaks_and_titles** (21,334)
- **only_titles** (2,117)

Examples:

- Amendment text (16,083 docs): Broken into title/linebreak sequences
- Committee schedules (682 docs): Meeting entries as separate titles
- Nomination lists (755 docs): Each nomination as a title
- Expenditure reports (200 docs): Tables broken into title/linebreak

Well-Parsed Documents

These have substantive content with meaningful structure:

- **speeches_and_linebreaks** (24,946): One-minute speeches
- **speeches_with_other** (6,997): Debates
- **speeches_with_linebreaks_titles** (6,463): Structured speeches
- **recorders_and_linebreaks** (5,355): Bill introduction lists
- **only_speeches** (4,571): Pure debates
- **recorders_with_linebreaks_titles** (2,937): Structured bill lists
- **only_recorders** (152): Recorder lists

This all have speeches and/or recorders

Implementation Strategy: Reclassifying Unknown Items

Overview

When the first item kind is `Unknown`, we need to:

1. Determine the appropriate new kind based on `doc_title` (and possibly content analysis)
2. Decide if it's a "whole_doc" kind (consume all remaining content) or continue with existing parsing

Can We Classify by `doc_title` Alone?

Answer: YES for 96%+ of cases

Our analysis shows very strong correspondence between `doc_title` and structure:

- Documents with the same title almost always have the same structure pattern
- Example: "PRAYER" → 99% are `only_linebreaks` structure
- Example: "DESIGNATION OF SPEAKER PRO TEMPORE" → 100% are `only_linebreaks` structure

Need content check for <4% of cases where:

1. **Title is ambiguous** between whole_doc and parsed
    - Example: "ADDITIONAL SPONSORS"
        - 41% are `only_unknown` (whole_doc)
        - 40% are `only_linebreaks` (whole_doc)
        - 10% are `recorders_and_linebreaks` (continue parsing)
    - **Solution:** Check if content[1] exists and what kind it is
2. **Title varies in structure**
    - Example: "TEXT OF AMENDMENTS"
        - 48% are `only_linebreaks_and_titles`
        - 40% are `recorders_with_linebreaks_titles`
    - **Solution:** Check if next items are recorders (continue parsing) or just titles/linebreaks (whole_doc)
3. **Generic or unclassified titles**
    - Example: "AUTHORITY FOR COMMITTEES TO MEET"
        - Could be speeches or simple notice
    - **Solution:** Check if next items are speeches (continue parsing) or linebreaks (whole_doc)

Or maybe a simpler approach

Maybe we don't need to add a lot of new kinds

Instead, if the content consists of a possible `unknown`, following by just `titles` and `linebreaks`, combine them all into a `body` kind

use `doc_title` to identify the document kind, add it as `doc_kind` as a top-level attribute

Asked Claude to evaluate this approach

Your Proposal vs. Better-Classification Docs

Similarities

1. **Use doc_title for classification**: Both approaches recognize that doc_title is highly effective (92%+ success rate) for classifying unknown items.
2. **Pattern matching on doc_title**: Your idea to create a doc_kind through pattern matching on doc_title is essentially what classify_by_title() does in IMPLEMENTATION_STRATEGY.md.
3. **Simplify content structure**: Both approaches recognize that ~50% of documents with unknown items could be simplified (what you call combining into a "body" kind).

Key Differences

Your Simpler Approach:
- Add **one new field**: doc_kind (top-level)
- Add **one new content kind**: body (for simplified content)
- Keep existing content kinds unchanged
- Rule: If content is just [unknown?, title*, linebreak*] → combine into single body item

Better-Classification Approach:
- Add **multiple new content kinds** (~15-20 new kinds like prayer, pledge, interchamber_message, etc.)
- Add **whole_doc attribute** to item_types
- More complex logic for handling different document types
- Attempts to create semantic categories for each type

Advantages/disadvantages

Advantages of Your Approach

1. **Much simpler implementation**: One new field, one new kind vs. 15-20 new kinds
2. **Less invasive**: Doesn't require expanding item_types significantly
3. **Cleaner separation**: doc_kind describes the document, content kinds describe structure
4. **Easier to extend**: Adding new doc_kind patterns doesn't require new content parsing logic

What You're Trading Off

1. **Less semantic specificity in content**: The better-classification approach makes content[0].kind = 'prayer' whereas yours would be doc_kind = 'prayer' with content[0].kind = 'body'
2. **Uniform handling**: Your body kind treats all simplified documents the same, which is actually a feature (simpler), not a bug
3. **Future structure handling**: The better-classification docs acknowledge but defer structured parsing (tables, schedules), which your approach also does

Claude's Assessment

Your approach is **pragmatically superior** for these reasons:

1. **It solves the same core problem** (55% of documents don't fit the parsing model) with much less complexity
2. **The separation is cleaner**: doc_kind for document semantics, content kind for structural semantics
3. **It's extensible**: Future work on structured parsing (tables, sections) can still happen within the body kind by adding more attributes
4. **It avoids proliferation**: The better-classification approach adds 15-20 new kinds that mostly encode "this is a whole document" - your body kind captures that uniformly