Zotero → LLM → Notion

Semantic Tag Automation with Actions & Tags and Notero

This guide configures a fully automated pipeline: when you add a new item to Zotero, a custom JavaScript snippet extracts its abstract, calls an LLM API to generate 3–5 semantic keyword tags, writes them back to the item, and lets Notero carry those tags directly into a Notion Multi-select property.

Step	Actor	Action
1	Zotero	User saves item (connector, drag-drop, manual)
2	Actions & Tags	createItem event fires → script runs
3	LLM API	Abstract sent → 3–5 keyword tags returned
4	Zotero item	Tags written via addTag(); item.saveTx() called
5	Notero	Detects item modification → syncs Tags to Notion
6	Notion	Tags column (Multi-select) populated automatically

Figure 1 — End-to-end pipeline sequence

Part 1 — Notion Database Setup

1.1 Required: Tags Multi-select property

Open your Notion database and add a property with the exact name Tags and type Multi-select. Notero is case-sensitive — any deviation will silently skip tag sync.

1.2 Recommended full property schema

Property Name	Type	Required for this workflow
Name	Title	Yes (Notero required)
Abstract	Text	Recommended
Authors	Text	
Year	Number	
DOI	URL	
Item Type	Select	
Collections	Multi-select	
Tags	Multi-select	YES — receives LLM tags

Property Name	Type	Required for this workflow
Zotero URI	URL	
Full Citation	Text	
Date Added	Date	

1.3 Connect Notero

- In Notion: database → ··· → Connections → search Notero → add.
- In Zotero: Tools → Notero Preferences → Connect to Notion (OAuth flow).
- Under Monitored Collections, tick the collections you want synced.
- Enable Sync when items are modified — this is what picks up tag changes.

Part 2 — Install & Configure Actions & Tags

2.1 Install the plugin

- Download the latest `.xpi` from github.com/windingwind/zotero-actions-tags/releases.
- In Zotero: Tools → Add-ons → gear → Install Add-on From File → select the `.xpi`.
- Restart Zotero.

2.2 Store your API key securely

Paste the following snippet into Tools → Developer → Run JavaScript (run once only — do not include it in the action script itself):

```
Zotero.Prefs.set('extensions.actionsTags.llmApiKey', 'sk-YOUR-KEY-HERE', true);
```

> The key is stored in `zotero.prefs`, never synced to Zotero's cloud servers.

2.3 Create the LLM tagging action

Edit → Preferences → Actions & Tags → click + to add a new action.

Field	Value
Name	LLM Semantic Tags
Event	createItem
Operation	Script
Enabled	[x] checked
Data	(paste the full script from Part 3)
Shortcut	(leave blank — runs automatically)

Part 3 — The Automation Script

Paste the following into the Data field of the action. The script handles absent abstracts, de-duplicates tags, and fails gracefully on network errors.

```
// LLM Semantic Tagger — paste into Actions & Tags > Script > Data
// Event: createItem
// ── CONFIGURATION ──────────────────────────────────────────────
const LLM_ENDPOINT = 'https://api.openai.com/v1/chat/completions';
const LLM_MODEL    = 'gpt-4o-mini';  // swap for gpt-4o, claude, etc.
const TAG_PREFIX   = '';             // e.g. 'ai/' → produces 'ai/pedagogy'
const MIN_TAGS = 3, MAX_TAGS = 5;
// ──────────────────────────────────────────────────────────────
(async () => {
  if (!item) return '[LLMTagger] No item.';
  if (!item.isRegularItem()) return '[LLMTagger] Not a regular item.';
  const abstract = item.getField('abstractNote');
  if (!abstract || abstract.trim().length < 50)
    return '[LLMTagger] Abstract absent/too short.';
  const apiKey = Zotero.Prefs.get('extensions.actionsTags.llmApiKey', true);
  if (!apiKey) return '[LLMTagger] No API key stored.';
  const title    = item.getField('title') || '(no title)';
  const itemType = item.itemType || 'unknown';
  const systemPrompt = `You are an academic research librarian.
Generate concise, semantically meaningful keyword tags for scholarly items.
Tags: lowercase, 1-3 words, specific, disciplinary vocabulary.
Avoid generic terms like "research" or "study".`;
  const userPrompt = `Generate ${MIN_TAGS}-${MAX_TAGS} keyword tags.
Type: ${itemType}
Title: ${title}
Abstract: ${abstract.substring(0, 1500)}
Return ONLY a JSON array of strings.
Example: ["higher education", "self-determination theory", "curriculum design"]`;
  let generatedTags = [];
  try {
    const response = await fetch(LLM_ENDPOINT, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${apiKey}`,
      },
      body: JSON.stringify({
        model: LLM_MODEL,
        messages: [
          { role: 'system', content: systemPrompt },
          { role: 'user',   content: userPrompt   },
        ],
        temperature: 0.3,
        max_tokens: 150,
        response_format: { type: 'json_object' },
      }),
    });
    if (!response.ok) {
      return `[LLMTagger] API error ${response.status}: ${await response.text()}`;
    }
    const data = await response.json();
    const raw  = data?.choices?.[0]?.message?.content ?? '[]';
    let parsed;
    try   { parsed = JSON.parse(raw); }
    catch { const m = raw.match(/\[.*?\]/s); parsed = m ? JSON.parse(m[0]) : []; }
    const arr = Array.isArray(parsed) ? parsed
              : Array.isArray(parsed?.tags) ? parsed.tags : [];
    generatedTags = arr
```

```
      .filter(t => typeof t === 'string' && t.trim())
      .map(t => (TAG_PREFIX + t.trim().toLowerCase()).substring(0, 60))
      .slice(0, MAX_TAGS);
  } catch (e) {
    return `[LLMTagger] Network error: ${e.message}`;
  }
  if (!generatedTags.length) return '[LLMTagger] No usable tags returned.';
  const existing = new Set(item.getTags().map(t => t.tag.toLowerCase()));
  let added = 0;
  for (const tag of generatedTags) {
    if (!existing.has(tag)) { item.addTag(tag); added++; }
  }
  if (added > 0) await item.saveTx();
  // saveTx() triggers Notero's "sync on modify" — tags flow to Notion.
  return `[LLMTagger] Added ${added} tag(s): ${generatedTags.join(', ')}`;
})();
```

Part 4 — Adapting to Other LLM Providers

Anthropic Claude

```
// Endpoint + auth
const LLM_ENDPOINT = 'https://api.anthropic.com/v1/messages';
const LLM_MODEL    = 'claude-3-haiku-20240307';
// In the fetch() call, replace headers + body:
headers: {
  'Content-Type': 'application/json',
  'x-api-key': apiKey,
  'anthropic-version': '2023-06-01',
},
body: JSON.stringify({
  model: LLM_MODEL, max_tokens: 150,
  system: systemPrompt,
  messages: [{ role: 'user', content: userPrompt }],
}),
// Parse: data?.content?.[0]?.text ?? '[]'
```

Local Ollama (no API key needed)

```
// Start Ollama with CORS open for Zotero:
// OLLAMA_ORIGINS=app://zotero.org ollama serve
const LLM_ENDPOINT = 'http://localhost:11434/api/chat';
const LLM_MODEL    = 'llama3.2';   // or mistral, gemma3, etc.
// Replace fetch body:
body: JSON.stringify({
  model: LLM_MODEL, stream: false,
  messages: [
    { role: 'system', content: systemPrompt },
    { role: 'user',   content: userPrompt   },
  ],
}),
// No Authorization header needed.
// Parse: data?.message?.content ?? '[]'
```

Part 5 — Timing & Sync Sequence

The pipeline relies on Notero's sync-on-modify trigger. The script calls `item.saveTx()` after writing tags, which fires the same item-modification event Notero monitors — so tags are always present before the Notion push happens.

Handling late-arriving abstracts

The Zotero browser connector sometimes saves metadata in stages: the item record exists before the abstract arrives. If the script fires before the abstract is populated, add a one-shot retry observer:

```
// Replace the early-return for missing abstracts with:
if (!abstract || abstract.trim().length < 50) {
  const obs = Zotero.Notifier.registerObserver({
    notify: async (event, type, ids, _) => {
      if (event === 'modify' && type === 'item' && ids.includes(item.id)) {
        const ab = item.getField('abstractNote');
        if (ab && ab.trim().length >= 50) {
          Zotero.Notifier.unregisterObserver(obs);
          // Re-run the tagging logic here (wrap in a shared function)
        }
      }
    }
  }, ['item']);
  return '[LLMTagger] Waiting for abstract...';
}
```

Part 6 — Troubleshooting

Symptom / Error	Resolution
Script never runs	In Actions & Tags Prefs, confirm Enabled [x] and Event = createItem
'No API key found'	Tools → Developer → Run JavaScript: Zotero.Prefs.set('extensions.actionsTags.llmApiKey', 'sk-...', true)
Tags in Zotero but not Notion	Notero Prefs: enable Sync when items are modified; confirm the collection is monitored
Tags column absent in Notion	Create property named exactly Tags (capital T), type Multi-select
LLM returns {} with no array	Remove response_format: {type:'json_object'} from the fetch body
CORS error — Ollama	Restart with OLLAMA_ORIGINS=app://zotero.org ollama serve
Duplicate Notion pages	Expected — Notero deduplicates via the 'notion' tag link attachment it adds to each item

Part 7 — Optional Enhancements

7.1 Colour-code AI tags in Zotero

```
// After item.saveTx():
```

```
for (const tag of generatedTags) {
  await Zotero.Tags.setColor(
    item.libraryID, tag,
    '#5C9EAD',  // teal — any hex colour
    0           // priority (0 = no keyboard shortcut)
  );
}
```

7.2 Write AI tags to a separate Notion property

Rather than mixing LLM tags with manual Zotero tags, write them to the Extra field with a prefix. Notero syncs Extra to a Notion Text property, from which a Notion formula can extract a list:

```
// Instead of item.addTag(), write to Extra:
const existing = item.getField('extra') || '';
const aiLine   = `AI Tags: ${generatedTags.join(', ')}`;
item.setField('extra', existing ? `${existing}\n${aiLine}` : aiLine);
await item.saveTx();
```

7.3 Back up your action configuration

In Actions & Tags Preferences, click the export button to save a `.yml` backup file. Action configurations are not included in Zotero's cloud sync — keep the backup in your notes folder or a version-controlled repo.

Plugins referenced: windingwind/zotero-actions-tags · dvanoni/notero