Topic Shift Detection for Chatbots: An Embedding-Centric Approach

1. Overview

We introduce a topics collection

Each topic stores its own summaries, embeddings, and message IDs.
When a new query arrives, we use vector similarity to determine:

- Should it belong to an existing research topic?
- Should a **new topic** be created?

This method is fast, deterministic, and scalable.

---

2. Database Schema

`topics` collection (new)

```
{
  "_id": ObjectId,
  "conv_id": ObjectId,
  "topic_id": "topic_1",

  "interaction_ids": [ObjectId, ...],

  "long_summary": "Detailed summary of this topic's research-related discussion.",
  "short_summary": "1–2 line description of this research topic.",

  "centroid_embedding": [0.23, -0.14, ...],
  "entity_set": ["Google", "Q3 earnings", "Alphabet financials"],

  "message_count": 12,
  "created_at": ISODate,
  "updated_at": ISODate
```

```
}
```

Examples of entity sets:

- Topic “Google Earnings”
  → `["Google", "Alphabet", "Q3 revenue", "ad business", "cloud margins"]`

- Topic “BlackRock Macro Risk Outlook”
  → `["BlackRock", "risk commentary", "macro strategy", "bond yields"]`

- Topic “NVIDIA Client Note Q4”
  → `["NVIDIA", "GPU demand", "data center revenue", "client outlook"]`

---

3. Embedding-Based Topic Classification

When a user interacts with the chatbot, each message is transformed into a **semantic embedding**. These embeddings naturally cluster according to the underlying research theme.

By representing each topic with a **centroid embedding**, we can:

1. Assign messages to existing clusters

2. Create new clusters when content diverges

This ensures topic purity even with highly interwoven research queries.

---

4. End-to-End Processing

4.1 Step 1 — Generate embedding for the new message

Example:

> “What were BlackRock’s risk warnings for 2025 in their latest macro note?”

Compute normalized embedding:

$$e(m) \in \mathbb{R}^{d}$$

Where d = embedding dimension (e.g., 768).

4.2 Step 2 — Fetch all topics in the current conversation

These may include:

- Topic 1 → Google Earnings Discussion
- Topic 2 → BlackRock Risk Outlook
- Topic 3 → NVIDIA Client Quarterly Update

4.3 Step 3 — Compute similarity with each topic centroid

For each topic t with centroid μ□:

Compute similarity

Interpretation:

- 0.80–0.95 → Same research domain

- 0.50–0.75 → Similar but not certain
- <0.50 → Likely unrelated

For example:

Topic	**Centroid Similarity**
Google Earnings	0.18
BlackRock Risk	**0.91**
NVIDIA Q4 Note	0.22

New query → assigned to “BlackRock Risk”.

4.4 Step 4 — Threshold Decision

Let t* = topic with maximum similarity S*.

If:

If S* > 0.75 , then

→ Assign message to existing topic t*.

Else:

→ Create **new topic**, e.g.:

- “JP Morgan Equity Outlook”
- “Macro Fed Interest Commentary”
- “ESG Sector Probabilities”
- “AI Equity Trend Analysis”

4.5 Step 5 — Update Topic Centroid

Given:

- μ□ = old centroid
- e(m) = new embedding
- n = previous message_count

Update via running average:

$$\mu_{t,\text{new}} = \frac{n \cdot \mu_{t,\text{old}} + e(m)}{n+1}$$

Normalize:

$$\mu_{t,\text{new}} \leftarrow \frac{\mu_{t,\text{new}}}{\left\|\mu_{t,\text{new}}\right\|}$$

This keeps the centroid aligned with evolving discussions such as:

- Shifts in BlackRock rhetoric
- Changes in NVIDIA guidance
- Updates in Google revenue segmentation

4.6 Step 6 — Update Topic Document

Fields updated:

- `interaction_ids`
- `centroid_embedding`
- `entity_set`
- `long_summary`
- `short_summary`

- `updated_at`
- increment `message_count`

---

5. Why This Works (Simple Mathematical Explanation)

1. Messages live in a “semantic space”

Each financial query:

- “Google’s cloud margins?”
- “BlackRock’s treasury yield assumptions?”
- “NVIDIA’s data center revenue?”

is represented by a vector $e(m)$.

2. Similar research questions form clusters

Queries about the same company/sector fall into dense groups.

Mathematically:

$$\begin{aligned}\text{Google Cluster} &= \{e(m_1), e(m_2), ...\} \\ \text{BlackRock Cluster} &= \{e(m_k), e(m_{k+1}), ...\}\end{aligned}$$

3. The centroid represents topic meaning

$$\mu_t = \frac{1}{n}\sum_{i=1}^{n} e(m_i)$$

This is the “center of gravity” of the topic.

4. Cosine similarity identifies closest topic

$$\cos(e(m), \mu_t)$$

Highest similarity → best semantic match.

5. Threshold decides new vs existing topic

This avoids wrongly mapping:

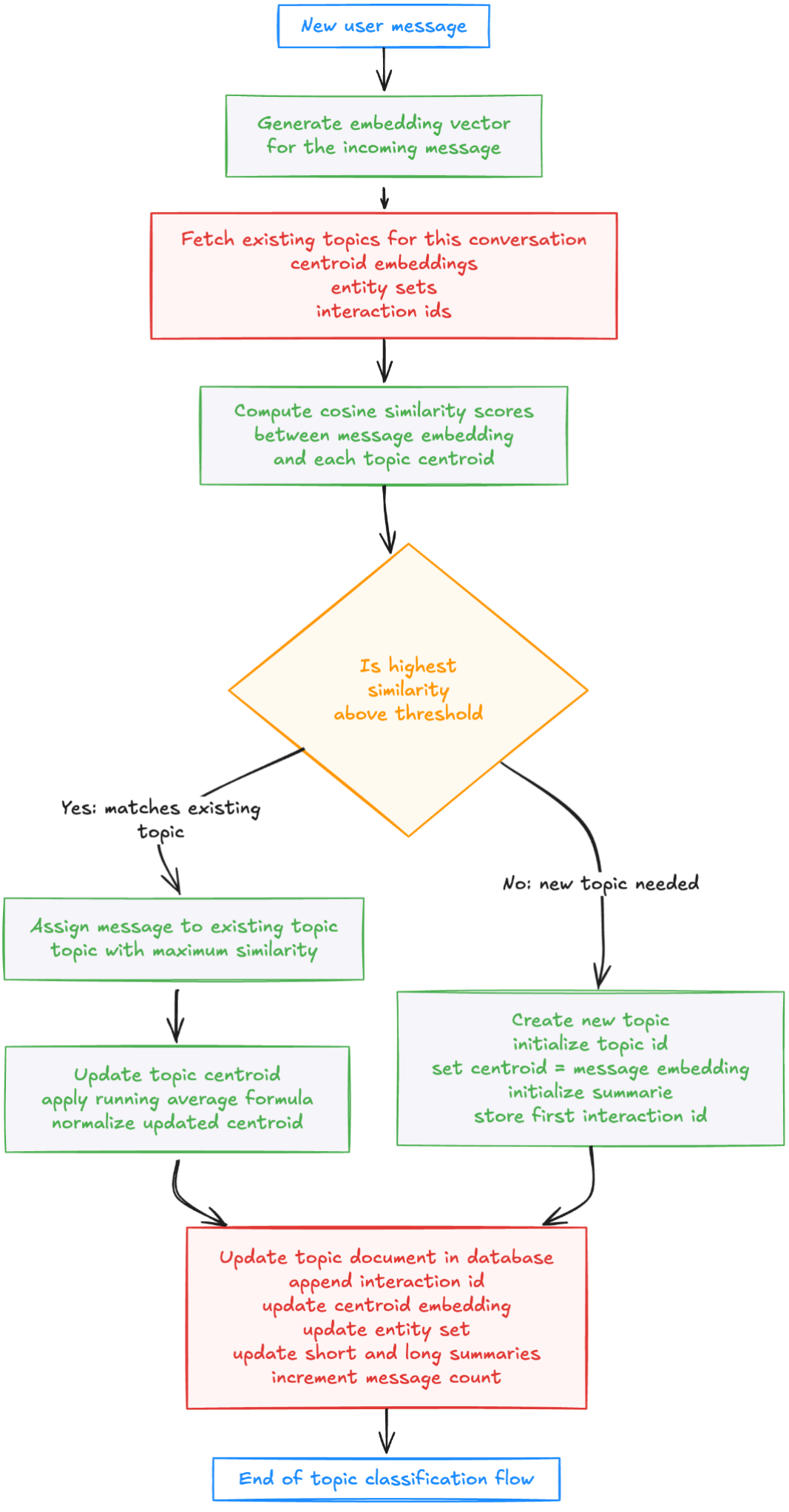
- “Create a JP Morgan EM outlook brief”
  to

- Google Earnings

Even if the conversation is interleaved like:

`Google → BlackRock → Google → NVIDIA → JP Morgan`

The vectors naturally separate these themes.


New user message
Generate embedding vector
for the incoming message
Fetch existing topics for this conversation
centroid embeddings
entity sets
interaction ids
Compute cosine similarity scores
between message embedding
and each topic centroid
Is highest
similarity
above threshold
Yes: matches existing
topic
No: new topic needed
Assign message to existing topic
topic with maximum similarity
Update topic centroid
apply running average formula
normalize updated centroid
Create new topic
initialize topic id
set centroid = message embedding
initialize summarie
store first interaction id
Update topic document in database
append interaction id
update centroid embedding
update entity set
update short and long summaries
increment message count
End of topic classification flow

[Simulating Dual-Process Thinking in Dialogue Topic Shift Detection]