Design Document: Pain Point to Solution Agent

for Filum.ai

Author: Nguyen Cong Nguyen

August 16, 2025

Contents

Agent Input Structure

Proposed Input Fields

The agent requires a structured input where the `pain_point` is mandatory, and additional context is optional to enhance matching precision within Filum.ai's ecosystem:

- **Pain Point (string):** The primary problem statement, expressed in the customer's words.
- **Context (object, optional):**
  - **Channel / Touchpoint:** e.g., `"email"`, `"mobile app"`, `"Zalo"`, `"POS"`.
  - **Customer Profile:** e.g., `"B2B"`, `"retail"`, `"B2C"`.
  - **Priority or Severity:** e.g., `"high"`, `"medium"`.
  - **Language / Region:** e.g., `"Vietnamese"`, `"APAC"`.
  - **Industry:** e.g., `"e-commerce"`, `"telecom"`.
  - **Specific Touchpoints:** e.g., `"post-purchase"`, `"support ticket"`.

Sample JSON Inputs

Minimal JSON Example (No Context)

```
{
  "pain_point": "We're struggling to collect customer feedback
      consistently after a purchase."
}
```

Full JSON Example (With Context)

```
{
  "pain_point": "We have no clear idea which customer touchpoints are
      causing the most frustration.",
  "context": {
    "channel": "multi-channel (Web, Zalo, Email)",
    "customer_type": "B2C retail",
    "priority": "medium",
    "language": "Vietnamese",
    "industry": "e-commerce",
    "specific_touchpoints": "post-purchase, support interactions"
  }
}
```

Rationale

- JSON is machine-readable and easy to parse programmatically.
- Expanded context fields (e.g., industry, specific touchpoints) enable finer-grained matching, prioritizing features like VoC - Journeys for multi-touchpoint issues in APAC regions, while remaining optional for flexibility.
- Free-text pain point ensures flexibility in capturing nuanced problems, with support for multi-language inputs like Vietnamese.

Agent Output Structure

Proposed Output Fields

The output includes a summary and an array of suggestions, each with:

- **Feature Name** (string)
- **Category / Subcategory** (string)
- **Description** (string)
- **How it Helps** (string, 1–2 sentences)
- **Relevance Score** (float)
- **Documentation Link** (string, optional)
- **Integration Notes** (string, optional)

Sample JSON Output

JSON Example


```
{
  "summary": "Based on your pain point, here are the top 2 relevant
     Filum.ai solutions ranked by relevance.",
  "suggestions": [
    {
      "feature": "Customer Journey Experience Analysis",
      "category": "Insights - Experience",
      "description": "Analyzes feedback and data across journeys to
         pinpoint friction.",
      "how_it_helps": "Identifies key touchpoints causing frustration
         through topic and sentiment analysis.",
      "relevance_score": 0.95,
      "docs_link": "https://filum.ai/docs/insights-experience",
      "integration_notes": "Combines with VoC - Surveys for deeper
         insights."
    },
    {
      "feature": "AI-Powered Topic & Sentiment Analysis",
      "category": "VoC - Conversations/Surveys",
      "description": "Automatically processes text feedback for themes
         and emotions.",
      "how_it_helps": "Reduces manual effort in spotting patterns across
         thousands of responses.",
      "relevance_score": 0.88,
      "docs_link": "https://filum.ai/docs/voc-analysis",
      "integration_notes": "Integrates with Insights - Experience for
         operational monitoring."
    }
  ]
}
```


Rationale

The JSON format ensures structured, machine-friendly results. A top-level **summary** provides an overview for multiple suggestions. The mandatory **relevance score** improves trust and enables ranking. Adding **integration notes** makes outputs more actionable, helping users visualize adoption within Filum.ai's ecosystem (e.g., cross-links to related features).

Feature Knowledge Base (KB) Structure

Schema Design

Each feature is represented as a structured record:

- **ID / Name**
- **Category / Subcategory**
- **Description**
- **Keywords / Tags**
- **Sample Pain Points (optional)**
- **Documentation Link**
- **Supported Channels** (array, optional)
- **Related Features** (array of IDs, optional)

Sample JSON KB Records

The KB is stored as an array of records covering Filum.ai's key categories (VoC, AI Customer Service, Insights, Customer 360, AI & Automation). Below are examples:

Knowledge Base Examples (Excerpt)

```
[
  {
    "id": "VoC_Surveys_PostPurchase",
    "name": "Automated Post-Purchase Surveys",
    "category": "Voice of Customer",
    "sub_category": "Surveys",
    "description": "Automatically sends surveys via email or SMS after
       purchase.",
    "keywords": ["post-purchase", "survey", "feedback", "automated",
       "consistent collection"],
    "sample_pain_points": ["struggling to collect customer feedback
       consistently after a purchase"],
    "docs_link": "https://filum.ai/docs/voc-surveys",
    "supported_channels": ["Email", "SMS", "Zalo"],
    "related_features": ["Insights_Experience_Analysis"]
  }
]
```

Rationale

- **Keywords:** Improve recall for keyword-based matching.
- **Descriptions:** Support semantic similarity approaches.
- **Categories:** Useful for filtering results or clustering features.
- Expanded attributes like **supported channels** and **related features** enhance matching accuracy and extensibility, ensuring comprehensive coverage of Filum.ai's features for better scalability.

Core Logic & Matching Approach

Overview

The agent follows a hybrid matching strategy that combines keyword and semantic approaches for robust, accurate suggestions.

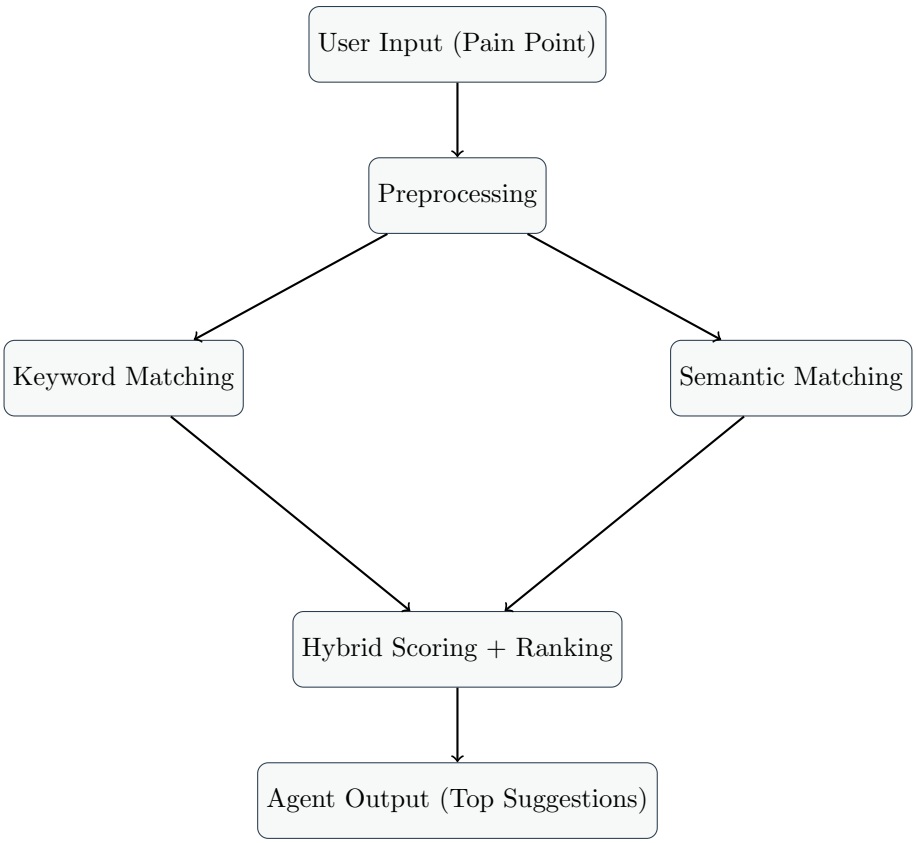
Steps

1. **Preprocessing:** Normalize text (lowercasing, removing punctuation, lemmatization) using libraries like NLTK.
2. **Keyword Matching:** Apply TF-IDF (via scikit-learn) to match pain points with feature keywords/descriptions.
3. **Semantic Matching:** Use embeddings (e.g., Sentence-BERT from Hugging Face or OpenAI) to compute cosine similarity.
4. **Hybrid Scoring:** Weighted average (e.g., 40% keyword, 60% semantic) of similarities.
5. **Ranking & Thresholding:** Select top-k features (e.g., $k = 3$) above a relevance threshold (e.g., 0.7 minimum score).
6. **Output Formatting:** Present final suggestions in JSON, sorted by score.

Handling Edge Cases

- For ambiguous pain points, fallback to category-based filtering using context fields.
- Multi-language support: Use multilingual embeddings (e.g., supporting Vietnamese) for APAC-focused queries.
- No matches: Return a summary prompting for more context, with zero suggestions.

Workflow Diagram



Examples of Pain Points & Solutions

Illustrative Mappings

The following table maps example pain points to Filum.ai solutions, drawing from the provided challenge examples:

Pain Point	**Solution (Feature & Category)**	**How it Helps**
We're struggling to collect customer feedback consistently after a purchase.	Automated Post-Purchase Surveys (VoC - Surveys)	Triggers surveys automatically via email/SMS after a transaction to ensure consistent feedback collection.
Our support agents are overwhelmed by the high volume of repetitive questions.	AI Agent for FAQ & First Response (AI Customer Service - AI Inbox)	Deflects common queries and provides instant answers, freeing up human agents and reducing workload.
We have no clear idea which customer touchpoints are causing the most frustration.	Customer Journey Experience Analysis (Insights - Experience)	Identifies friction points by analyzing feedback and operational data across the journey.
It's difficult to get a single view of a customer's interaction history when they contact us.	Customer Profile with Interaction History (Customer 360 - Customers & AI Inbox)	Consolidates all touchpoints and past interactions for a comprehensive view.
Manually analyzing thousands of open-ended survey responses for common themes is too time-consuming.	AI-Powered Topic & Sentiment Analysis for VoC (VoC - Conversations/Surveys, Insights - Experience)	Automatically extracts key topics and sentiment from text feedback, saving time.