HAYSTACK EUROPE 2026

BEYOND LLM JUDGES

Deep Evaluation for Conversational Search

Ruchi Juneja

Sr. Data Scientist, Search ML — MediaMarktSaturn

THE SETUP

The Pre-Production Evaluation Problem

How do you evaluate a conversational search agent before a single real user has typed to it?

WHAT WE DON'T HAVE

Historical conversations

No past dialogues to replay against a new version

Clicks & conversions

No behavioral signal for what worked

User satisfaction feedback

No ratings, no complaints, no praise

Production failure patterns

No known bugs to regression-test against

SO WE NEED

Controlled, offline evaluation

1. Generate test data
2. Run the agent
3. Measure signals
4. Diagnose failure modes

Is the agent ready for release?

01

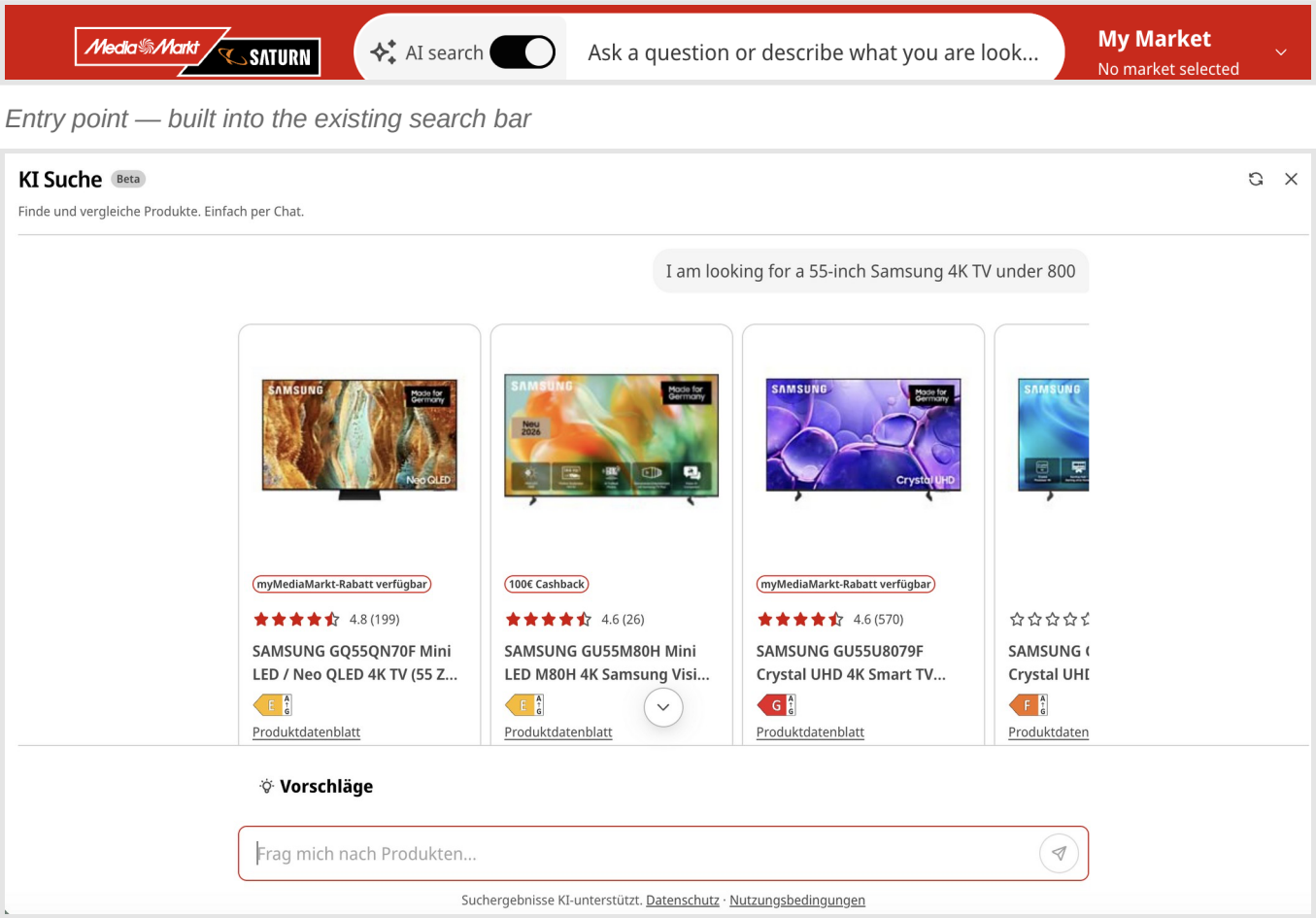
Intro to the Conversational Agent

Where does the agent make decisions?

THE PRODUCT

This Is What We're Evaluating

MediaMarktSaturn's AI Search



Entry point — built into the existing search bar

WHAT THE CUSTOMER SEES

1. Types a request in plain language — no filters, no dropdowns.

2. AI Search returns a ranked set of matching products directly in the chat.

3. Every decision behind this one response — the search term, the filters, whether to even search — is what the rest of this talk evaluates.

Screenshots shown on the German (DE) storefront.

SYSTEM OVERVIEW

Agent Interaction with the Search System

User prompt: "I am looking for a 55-inch Samsung 4K TV under 800€"

1

Understand user request

- Decide to call Search
- Extract search term
- Extract filter context

→

2

Discover candidate filters

- Asks the search system which filters exist
- e.g. brand, size, energy class, display size

→

3

Match requirements to filters

- Maps "Samsung" → brand
- "800 euros" → price

→

4

Get products

- Sends search term and filters
- Retrieves the product set

Every one of these decision points is a place things can quietly go wrong — and a place a single relevance score can't tell you which.

02

Offline Test Data Generation

No production data yet — so we build our own

FOUNDATION, NOT A FOOTNOTE

Building the Offline Test Set

1

Data Source

Well-distributed (Head/Torso/Tail) user queries from the conventional search system

2

Sampling

Stratified sampling to represent different product categories

3

Labelling

Label queries by category via LLM like Brand, Brand+Product, Product+Feature

4

Prompt Generation

Generate synthetic user prompts with good style coverage like generic asks, specific models, negation

SAMPLE TEST DATA

search term: *built-in stove set*
category: *Product*
intent: *budget*
prompt: *"Can I find a built-in stove set that costs less than 800 euros?"*

---

search term: *A3 printer*
category: *Product + Feature*
intent: *negation*
prompt: *"I'm looking for an A3 printer, but please not an Epson model."*

03

Evaluation Framework

One agent, five complementary questions

THE FRAMEWORK

One Agent, Multiple Evaluation Questions

Conversational search combines NLU, dialogue management, and retrieval — one score can't diagnose all three.

1	**OUTCOME**	Did the agent provide a relevant response?
2	**SEARCH UNDERSTANDING**	Did it understand what to search for, and capture the user's constraints?
3	**SEARCH BEHAVIOR**	Did the search interaction actually produce results?
4	**CONVERSATION BEHAVIOR**	Did the agent reliably satisfy the user across multiple turns?

04

Baseline: LLM-as-a-Judge

A useful starting point — but not yet a diagnosis

THE BASELINE METRIC

Single-Turn Search Relevance Score @Top3

WHAT DOES “SINGLE-TURN” MEAN?

- One user request → one agent response
- No clarification, no follow-up
- There's no opportunity to recover through dialogue

WHAT ARE WE MEASURING?

Do the top 3 products the search tool returned actually help the customer find what they asked for?

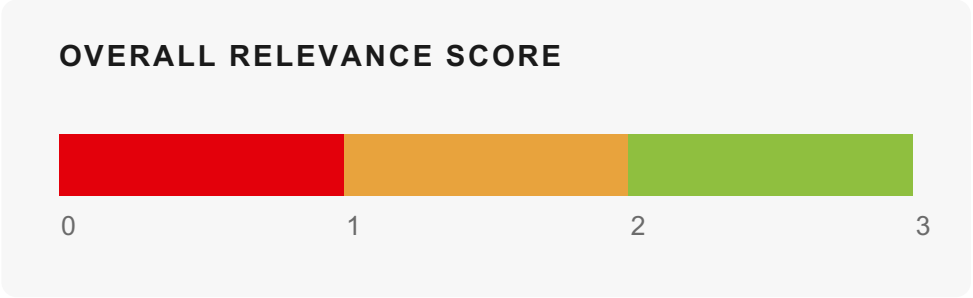
Only the top 3 products from the tool call are scored — not the broader catalog available to the system.

HOW IS IT CALCULATED?

LLM-as-a-judge scores each query–product pair 0–3 with reasoning → averaged per query → averaged across the eval set.

“Did we help the customer discover what they asked for?”

This is the starting point — not the diagnosis.



0 Not relevant

Asked for a fridge → got a TV

1 Slightly relevant

Related accessory / wrong category or brand

2 Moderately relevant

Generally useful, but misses a requirement

3 Highly relevant

Directly matches the user's needs

05

Decompose the Score

Measure → Diagnose → Improve → Re-evaluate

05.1

Search Term Accuracy

Did we understand what the user is looking for?

SEARCHTERM — ITERATION 1

Search Term Accuracy: Our First Definition

WHAT IS IT?

Does the agent extract the expected search term?

WHY IT MATTERS

If we misread the main term, everything downstream suffers.

METRIC

$$\frac{\text{count(extracted == expected)}}{\text{total prompts}}$$

Hypothesis: exact match between expected and extracted search term

EXAMPLES

Prompt	Expected	Extracted	
Can I find a built-in oven set under 800€?	built-in oven set	*built-in oven set*	✓
Do you have samsung galaxy phones available?	samsung galaxy	*smartphone*	✗
nice cheap sony headphones	sony headphones	*headphones*	✗
powerful gaming laptop, great graphics card	gaming laptop w/ GPU	*gaming laptop*	✗
iphone 15 pro	iphone 15 pro	*smartphone*	✗
ipad	ipad	*tablet*	✗

SEARCHTERM — DIAGNOSIS

What Iteration 1 Revealed

The agent was often too broad — optimizing for recall, when search needed specificity.

1

Search term too generic

`iPhone 15 Pro → "smartphone"`

Product-specific request collapses into a broad category

2

Broad terms trade precision for recall

`sony headphones → "headphones"`

More results, lower zero-result risk — but more irrelevant results

3

Heavy dependence on facets

`sony headphones, noise cancelling → headphones + filters`

A missing facet can turn a broad search term into a bad result set

4

Language normalization

`"lavatrice" → washing machine`

Locale/ language not respected

The agent kept Search term broad, trusting brand/price/color to be captured as facets. **That works beautifully — unless a facet is missed, and the useful information is simply gone.**

SEARCHTERM — ITERATION 2

Improving Search Term Generation & Evaluation

Not just a relaxed metric — real changes at three layers.

SEARCH UNDERSTANDING

- Preserve user specificity in the term
- Include distinguishing features as part of search term
- Thread language / locale context

SEARCH PIPELINE

- Fallback to semantic search
- Candidate filters tied to the actual candidate set, not query category

EVALUATION

- Generate valid query variants
- Add normalization (multilingual, lemmatization)
- Add fuzzy matching

We didn't just change the evaluation criteria — we used the evaluation to drive real improvements in the agent and the pipeline.

SEARCHTERM — FINAL DEFINITION

What Counts as a Match?

✓ CORRECT MATCH

- Exactly matches the expected Search term
- A semantically equivalent formulation mapping to the same taxonomy node
- Adds product-identifying info like brand or type

✗ INCORRECT MATCH

- Includes filters that belong separately — "Samsung 55″ TV under 800€"
- Too generic to preserve intent — "TV" instead of "Samsung TV"
- Wrong product/category — "PS5" instead of "PS5 controller dock"

WHAT WE LEARNED

- **Synthetic data introduces noise**

  Generated prompts add descriptive detail the expected query lacks → higher false negatives

- **Generic search terms remain a risk**

  "HDMI cable, extra-long" → should not stay "extra-long HDMI cable", just "HDMI cable"

The goal is practical correctness — not linguistic completeness.

05.2

Filters Evaluation

Did we understand the constraints?

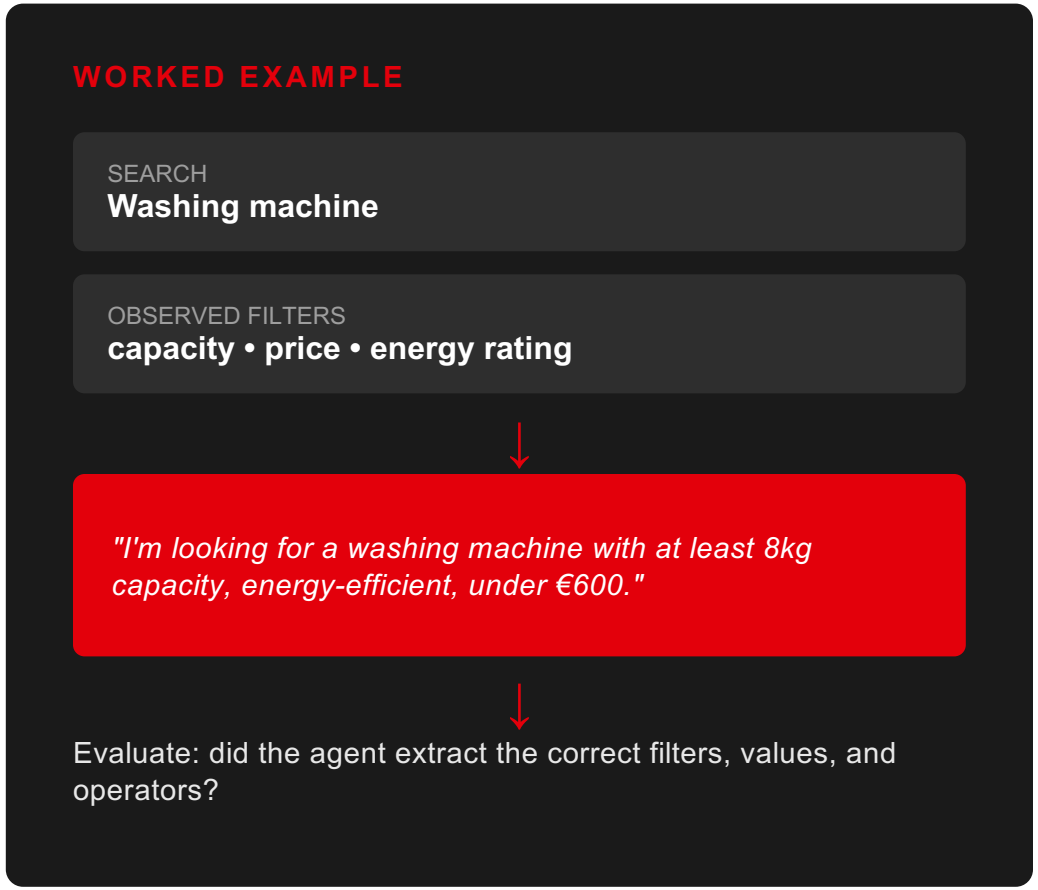
FACETS — TEST DATA

Building the Filter Evaluation Dataset

Filters are contextual. The data has to reflect that.

1. **Data Source**: well-distributed queries + applied filters from conventional search
2. **Query Sampling**: stratified across product categories
3. **Filter Sampling**: 1–3 filters per query, coverage-aware
4. **Prompt Generation**: synthetic prompts combining query + sampled filters

We combined the search query with filters real users had applied to that same query, then generated prompts expressing both together.



FACETS — ITERATION 1

Filter Accuracy: The Initial Question

Did the agent extract the same filters as the expected answer?
Not just "can the model extract attributes" — can it translate realistic user constraints into the structured filters search expects?

STARTED WITH A STRICT MATCHING APPROACH

- ✓ Attribute must match
- ✓ Filter value must match
- ✓ Operator must match
- ✓ No unexpected filters

```
Filter Accuracy = count(extracted == expected) / total prompts
```

Match → Correct | Mismatch → Incorrect

BUT…

The first evaluation showed that filter quality was not just an extraction problem.

A lot of what looked wrong was coming from the search schema and filtering strategy underneath the agent — not from what the model extracted.

FACETS — DIAGNOSIS

What Iteration 1 Revealed

1

Schema can confuse the model

- Attribute's name in the product dataset can be confusing
- Grouping similar attributes helps

2

Implicit filters can hurt relevance

- Having additional implicit filters can affect relevance
- Hallucinated constraints → irrelevant or zero results

3

Static filter logic isn't always right

- Fixed slider ranges default to 0 value lower bound
- Strict filters can produce zero results
- Filter relaxation may be needed

4

Some user intent is hard to extract

- Some filter values are hard to recover from prompts

KEY OBSERVATION

Filter values are usually extracted correctly.

The bigger challenge is deciding which facet to apply, how to apply them, and when to relax them.

FACETS — ITERATION 2

Facet Match Rate (FMR)

From strict facet/filter matching → search-aware matching.

Does the agent apply the right constraints for the user's intent — without introducing extra filters that cause harm?

✓ MATCHING

- Expected facets are represented (filter or in search term)
- Equivalent representations accepted
- Extra facets allowed if they don't cause zero results

✗ NON-MATCHING

- Missing — expected facet not applied
- Mismatch — wrong facet or value applied
- Additional — unexpected facet that harms retrieval

$$\text{FMR} = \frac{\#\ \text{Matching Facets}}{\#\ \text{Total Test Cases}} \times 100$$

"smartphone up to €1,200" → expected: price, applied: price → MATCHING

05.3

No Results Evaluation

Even correct pieces can combine into nothing

OUTCOME SIGNAL

No Results

How often does the agent return zero products for a product-seeking request?



No search invocation

Agent asks a follow-up instead of searching — e.g. "Can I buy a used ABCDEFG123?"

Over-restrictive filtering

SearchTerm + filters produce an empty set — e.g. "Hades Switch, offers under €20"

Search capability limitation

Limitation in underlying retrieval layer — e.g. "Thermos with an extra-large opening"

No-results is an outcome, not a single failure mode.

06

Multi-turn Evaluation

Can the agent remain correct across a conversation?

MULTI-TURN — TEST DATA

What Defines a Test Case?

Users build on what they said before. A single-turn snapshot can't test whether context survives.

PERSONA

- Name: Budget Shopper, Gamer, Gift Buyer, Deal Hunter…
- Traits: context/metadata for rubric framing
- Satisfaction signals: persona-specific regex
- Follow-up templates: price is missing, spec is missing, compare options…

GOAL

- Intent: find headphones, TV, exact phone, accessories…
- Expected tools the agent should invoke
- Completion signals: regex for goal satisfaction
- Rephrased goal prompt as fallback

POLICY

- Clarify → provide a structured profile
- Add constraint → ask for in-budget alternative
- Compare → request a focused comparison
- Ask for the final pick

MULTI-TURN — GENERATION

Deterministic Test Case Generation

Structured persona + goals → controlled, repeatable conversations. No free-form turn writing.

1 Define

User persona + goal + policy → a user scenario

→

2 Detect

Inspect the assistant's latest response — what's still missing?

→

3 Apply

Deterministic policy selects the right behavior

→

4 Generate

Predefined template produces the next user turn

RULE-BASED USER SIMULATOR

Template builders + rule triggers select or compose from predefined templates — never free-form invention — based on the assistant's response state.

Deterministic **Debuggable** **Repeatable** **Scalable**

Known expected state at every turn → we can score whether context actually survived.

MULTI-TURN — METRIC

Conversational Search Reliability Score

[LLM-as-a-judge, scored at the conversation level — not turn by turn]

Measures how reliably the agent:

- Understands the customer's actual shopping goal
- Grounds all claims in real product data (no fabrication)
- Respects all stated constraints (budget, features, location)
- Progresses toward a concrete, actionable recommendation

```
Score = (Average LLM Rubric Score across all
conversations) × 100
```

A request can be individually correct while the conversation as a whole becomes unreliable.

90–100 — **Excellent**

Customers fully trust the agent — constraints respected, no fabrication.

70–89 — **Good**

Mostly reliable; occasional gaps, but safe for most interactions.

50–69 — **Fair**

Frequent issues — customers may repeat themselves or verify facts elsewhere.

< 50 — **Poor**

Unreliable — high risk of hallucination and missed constraints.

BEYOND LLM JUDGES

DON'T JUST ASK

"How good is my agent?"

ASK

"Where does it fail — and why?"

SearchTerm | Facets | No Results | Multi-turn Reliability

Measure → Diagnose → Improve → Re-evaluate

A good evaluation doesn't just tell you whether your agent is good.

It tells you what to fix next.

LET'S CONNECT

Thank You

Questions, feedback, or just want to talk search and AI — I'd love to hear from you.

Ruchi Juneja

linkedin.com/in/ruchi-juneja

Search & AI Expert

SCAN TO CONNECT

linkedin.com/in/ruchi-juneja