CSPulse AI Guide

Kennis: Van Mens naar Machine

Hoe een organisatie zorgt dat Kennis door Mensen én door AI gebruikt kan worden

> **Kernprincipe**
>
> Automatiseer niet het hele gesprek. Automatiseer alleen de eerste minuten die voorspelbaar, herhalend en concreet geautomatiseerd uitvoerbaar genoeg zijn.

Voor wie

- Customer service managers en operations leads die AI-Triage op WhatsApp, chat of andere messaging kanalen willen starten.
- Teams die al een AI Agent platform hebben of dat binnenkort selecteren, maar nog niet scherp hebben waar en hoe ze moeten beginnen.
- Organisaties die eerst de Kennis op orde willen krijgen alvorens AI Agents te laten communiceren met klanten.

Wat deze Guide oplevert

- Een reality-check: is je kennis klaar voor AI?
- Een duidelijke manier om kennis te structureren zodat het voor Mens & Machine bruikbaar is.

Hoe je deze Guide gebruikt

Deze Guide is geen algemene AI-hypegids. Het is een beslis- en uitvoerkit voor teams die de triage op tekstkanalen gedeeltelijk van mens naar machine willen verplaatsen en daarvoor de kennis geschikt willen maken.

Werkvolgorde

1. Doe de readiness scan en bepaal of je organisatie nu, later of nog niet moet starten.
2. Analyseer welke contactstromen binnenkomen en breng volume, routing en frictie in beeld.
3. Zet ruwe contactredenen om in een bruikbaar Intentmodel.
4. Selecteer alleen die Intents die voorspelbaar genoeg zijn voor AI-Triage.
5. Controleer technologie, knowledge, governance en people-skills.
6. Maak een business case light en kies 3 tot 5 pilotIntents.

> **Belangrijk**
>
> Deze Guide gaat over kennis en moet worden gezien in samenhang met de implementatie van Triage door AI.

Knowledge Readiness: is je kennis AI-proof?

Kern van dit hoofdstuk:

Kennis is – samen met data-toegang – het grootste fundament onder elke AI-toepassing in customer service. Kennis die voor mensen is geschreven, werkt niet voor AI. Niet omdat AI dommer is dan mensen, maar omdat AI op een fundamenteel andere manier met kennis omgaat.

Wat je hierna hebt:

Begrip van waarom mens-kennis faalt in een AI-context, een concreet beeld van hoe AI-bruikbare kennis eruitziet, een checklist om je eigen kennis te beoordelen, en een realistisch pad om ernaartoe te groeien.

Wat je nodig hebt:

Eerlijkheid over de huidige staat van je kennisbank — niet de versie die je hoopt dat er staat, maar de versie die er daadwerkelijk staat.

Waarom kennis het grootste fundament is

In de vorige hoofdstukken hebben we het gehad over wat je met AI-Triage wilt doen: welke kanalen, welke Intents, welke kandidaten. Dit hoofdstuk gaat over waarmee je het gaat doen. En dat is kennis.

Er zijn ruwweg twee fundamenten onder elke serieuze AI-toepassing in customer service:

- **Data-toegang**: kan de AI Agent bij de systemen die hij nodig heeft? (API's, documentatie, authenticatie, realtime data.)
- **Kennis**: staat er vastgelegd wat de AI Agent moet doen, in een vorm die AI ook echt kan gebruiken?

De andere zaken, zoals platform, team/management, governance en metrics zijn stuk voor stuk belangrijk. Maar het zijn eerder infrastructuur en organisatie rond die twee fundamenten. Je kunt een geweldig team hebben en een perfect platform; als je kennis niet AI-bruikbaar is, werkt het alsnog niet. Je agent kan dan wel de Intent herkennen en de klantcontext ophalen, maar hij heeft niets om op te volgen. Hij weet niet wat hij moet zeggen of doen.

Er is nog een reden waarom dit hoofdstuk middenin staat en niet achteraan: **management zonder begrip is delegatie zonder controle**. Als je straks een AI Agent aanstuurt, delegeer je werk naar een medewerker die geen vragen stelt, niet terugkomt met "ik snap het niet", en zonder mopperen doet wat hij denkt dat je bedoelt. Dat is het gevaarlijkste soort medewerker. Je kunt een menselijk team aansturen zonder precies te weten hoe ze denken, omdat zij jou

corrigeren. Een AI Agent corrigeert je niet. Hij voert uit wat zijn kennis hem vertelt, inclusief fouten, gaten en verkeerde aannames.

Een CS-manager die AI Agents managet, hoeft geen developer te zijn. Maar hij moet wél begrijpen hoe AI kennis gebruikt, anders heeft hij geen zicht op waar en waarom het misgaat. Dat is de reden dat dit hoofdstuk iets technischer wordt dan de andere; niet om je developer te maken, maar om je in staat te stellen te managen wat je niet zelf bouwt.

Hoe AI met kennis omgaat; net genoeg technisch

Dit hoofdstuk gaat niet over hoe je AI-systemen bouwt. Maar om te begrijpen waarom mens-kennis niet werkt voor AI, moet je in grote lijnen weten hoe AI kennis ophaalt en gebruikt. Ik leg het hier in drie stappen uit.

Stap 1 – AI leest geen documenten, het ontvangt hapjes tekst

Wanneer een AI Agent in jouw CS-context een vraag van een klant krijgt, leest hij niet je hele kennisbank. Dat is praktisch onmogelijk; een kennisbank is te groot, en AI-modellen hebben een beperkt werkgeheugen (het zogenaamde context window). In plaats daarvan werkt het zo:

1) De kennisbank wordt van tevoren in kleine stukken geknipt — **chunks**. Een chunk is typisch een paar zinnen tot een paragraaf, afhankelijk van de instellingen.
2) Als de klant een vraag stelt, gaat de AI op zoek naar de chunks die het meest relevant lijken voor die vraag.
3) Die gevonden chunks worden bij de vraag gevoegd, en het AI-model formuleert daarmee een antwoord.

Deze techniek heet **RAG** — Retrieval Augmented Generation. In een later Hoofdstuk leggen we dat dieper uit. Voor nu is de belangrijkste consequentie:

> ***AI ziet nooit het hele plaatje in één keer. Hij ziet altijd stukjes.***

Dat heeft enorme gevolgen voor hoe je kennis moet opschrijven. Als in jouw artikel op regel 1 staat "Voor alle onderstaande procedures geldt: klanten buiten Nederland vallen onder procedure B" en op regel 47 staat de procedure, dan kan de AI Agent regel 47 ophalen zonder ooit regel 1 te zien. En dus antwoorden met procedure A terwijl procedure B van toepassing was.

Stap 2 – AI vindt wat er lijkt te passen, niet wat je bedoelt

De manier waarop AI relevante chunks vindt, heet semantic retrieval. Dat werkt via embeddings: elk stukje tekst krijgt een soort coördinaat in een hoogdimensionale ruimte, waarbij stukken met vergelijkbare betekenis dicht bij elkaar liggen. Als een klant vraagt "Waar blijft mijn pakket?", zoekt het systeem chunks waarvan de betekenis dicht bij die vraag ligt.

Als jouw kennisbank vol staat met artikelen die allemaal beginnen met "*Algemene procedure bij klantvragen...*", dan zijn die onderling moeilijk te onderscheiden. Het systeem vindt misschien wel het artikel over leveringsvragen, maar net zo goed het artikel over retourvragen. Structuur, duidelijke titels en expliciete trefwoorden helpen de AI om het juiste stuk te vinden.

Stap 3 – AI vult gaten in met aannames

Wat mensen bij onduidelijkheid doen: ze stellen een vervolgvraag. Wat AI bij onduidelijkheid doet (tenzij je expliciet anders instrueert): het vult het gat zelf in. Dat noemen we hallucinatie. Niet uit kwade wil; het AI-model probeert simpelweg een plausibel antwoord te geven op basis van wat het weet.

Als jouw kennisartikel zegt "*Voor een reparatieafspraak neemt u contact op met onze service desk*", kan de AI daarop aanvullen "*U kunt bellen tussen 08:00 en 17:00 op werkdagen*"; niet omdat dat ergens in jouw kennisbank staat, maar omdat dat een plausibel uur is voor een service desk. Als het feitelijk 09:00–18:00 is, heb je een fout antwoord dat als feit wordt gepresenteerd.

Hoe kleiner de gaten in je expliciete kennis, hoe minder ruimte voor dit soort invullen.

Wat dit betekent voor jou als manager

Je hoeft niet te weten hoe de mathematische modellen achter embeddings werken of wat de optimale chunk-size is. Je moet wél weten dat:

- AI werkt met **stukjes tekst**, niet met hele documenten. Je kennis moet daarom **zelfstandig leesbaar** zijn op stukjes-niveau.
- AI vindt wat **semantisch lijkt te passen**. Je kennis moet daarom **specifiek en onderscheiden**d geschreven zijn.
- AI **vult gaten** in als hij ze tegenkomt. Je kennis moet daarom **expliciet en volledig** zijn, geen impliciete aannames.

Deze drie eigenschappen (zelfstandig leesbaar, specifiek, expliciet) zijn de kern van wat we in dit hoofdstuk *AI-bruikbare kennis* noemen. In de volgende secties zien we wat dat concreet betekent.

Mensvriendelijke kennis vs AI-bruikbare kennis: de verschillen

Het punt van deze sectie is niet dat je kennisbank slecht geschreven is. Voor mensen werkt hij waarschijnlijk prima. Het punt is dat de eigenschappen die kennis mensvriendelijk maken, vaak precies de eigenschappen zijn die hem onbruikbaar voor AI maken.

Hieronder de zes belangrijkste verschillen, met concrete voor/na-voorbeelden.

Verschil 1: Expliciete vs impliciete beslisregels

Mensvriendelijke kennis zegt:

> *"Bij een reparatieverzoek beoordeelt u of het product nog binnen de garantietermijn valt en handelt u overeenkomstig."*

Elke ervaren agent weet wat dit betekent. Voor AI is dit leeg. "*Overeenkomstig*" wát? Hoe beoordeel je garantie? Wat is het verschil in handelen?

AI-bruikbare versie:

> **Beslisregel: reparatieverzoek**
>
> *Stap 1: Garantiestatus bepalen:*
>
> - Product gekocht < 24 maanden geleden: **binnen garantie**
> - Product gekocht 24–60 maanden geleden: **check aanvullende garantie** (zie klantdossier, veld `extended_warranty_active`)
> - Product gekocht > 60 maanden geleden: **buiten garantie**
>
> *Stap 2: Actie per status:*
>
> - **Binnen garantie** → Plan reparatieafspraak via flow REPAIR_WARRANTY. Geen kostenindicatie nodig.
> - **Aanvullende garantie actief** → Plan via flow REPAIR_EXTENDED. Vermeld eigen risico €25.
> - **Buiten garantie** → Verstuur kostenindicatie via flow REPAIR_QUOTE. Wacht op akkoord voor afspraak.

Merk op dat de AI-versie langer is, minder elegant, en minder aangenaam om te lezen. Dat is precies waarom mensvriendelijke kennis eindigt bij de eerste versie: mensen vullen de details zelf in.

Verschil 2: Volledige context vs context-aanname

Mensvriendelijke kennis gaat ervan uit dat je weet waar het over gaat. Een artikel getiteld "*Reparatieafspraak boeken*" hoeft niet opnieuw uit te leggen wat een reparatie is of om welk product het gaat; dat staat in de context van het gesprek.

AI-bruikbare kennis moet die context expliciet binnenhalen, niet aannemen.

Verschil 3: Zelfstandige leesbaarheid vs verwijzingen

Mensvriendelijke kennis zegt: "*Voor de stappen, zie bovenstaande tabel.*" Of: *"Zoals eerder genoemd in sectie 3...*". Of: "*Gebruik het standaardproces*." Dat werkt in een samenhangend document dat een mens van begin tot eind leest.

AI-bruikbare kennis kan niet verwijzen naar iets wat elders staat. Zoals we eerder zagen (chunking): de AI krijgt alleen het ene stukje. Een verwijzing naar "*bovenstaande tabel*" is zinloos als de AI de bovenstaande tabel niet ziet.

Regel: elk stuk AI-bruikbare kennis moet op zichzelf begrijpelijk en uitvoerbaar zijn. Wat je een mens niet zou geven zonder de rest van het document erbij, moet je aan AI dus *wél* kunnen geven.

Verschil 4: Modulair vs lopend

Mensvriendelijke kennisartikelen zijn vaak lopende tekst met kopjes. Een goed artikel leest als een verhaal: introductie, context, stappen, uitzonderingen, afsluiting.

AI-bruikbare kennis is modulair opgebouwd. Elke beslisregel, elk stappenplan, elke uitzondering is een op zichzelf staand blok met een eigen titel, expliciete scope en expliciete input/output. Het is minder leesbaar als verhaal, maar veel betrouwbaarder als bron voor geautomatiseerde handelingen.

Verschil 5: Jargon en afkortingen vs gestandaardiseerd taalgebruik

Mensvriendelijke kennis gebruikt het jargon van jouw organisatie: "*Als klant T2+ is, gebruik dan pad F.*" Agents weten wat een T2+-klant is, en wat pad F is.

AI kan dat ook leren, mits je jargon-woordenlijsten expliciet beschikbaar maakt. Maar in de praktijk faalt veel AI-Triage omdat de kennisbank vol zit met ongedocumenteerd jargon. De AI weet niet wat T2+ is, raadt iets, en heb je een fout antwoord.

Regel: óf je elimineert jargon, óf je maakt er een expliciete woordenlijst bij die altijd samen met de kennis beschikbaar is.

Verschil 6: Redactionele versus structurele metadata

Mensvriendelijke kennis heeft metadata als "laatst gewijzigd op 14 maart 2025 door Piet". Dat is handig voor het redactieteam.

AI-bruikbare kennis heeft structurele metadata: wat is de scope van dit artikel (welke producten, welke klantsegmenten, welk land)? Welke Intents triggert dit artikel? Welke vragen moet de AI stellen voordat hij dit artikel gebruikt? Welke data moet hij ophalen? Welke andere artikelen zijn gerelateerd?

Structurele metadata helpen de AI bepalen *wanneer* hij dit stuk kennis moet gebruiken en *onder welke voorwaarden*. Zonder die metadata ben je afhankelijk van semantic retrieval alleen; en dat is onbetrouwbaar voor complexe beslissingen.

Samenvattend

Niet elk mensvriendelijk kennisartikel is ongeschikt voor AI, en niet elk AI-bruikbaar artikel is onleesbaar voor mensen. Maar als je niets verandert aan je huidige kennisbank en hem gewoon aan een AI voert, is de kans zeer groot dat je op minstens drie van deze zes punten faalt. Dat is geen klein probleem dat je later oplost; **dat is de reden dat je pilot fout zal gaan.**

Het belang van context

Hiervoor noemden we context zijdelings. Het verdient een eigen sectie, want het is waarschijnlijk het meest onderschatte aspect van AI-bruikbare kennis.

Stel je hebt een perfect AI-bruikbaar artikel geschreven over "*Reparatieafspraak boeken*". Alle beslisregels expliciet, alle stappen modulair, alle jargon gestandaardiseerd. De AI Agent kan het artikel feilloos toepassen. Toch gaat het fout.

Waarom? Laten we een concreet scenario doorlopen bij *TechNova Retail*.

Scenario: twee identieke vragen, twee verschillende antwoorden

Klant A stuurt een WhatsApp-bericht: "*Mijn koffiezetapparaat werkt niet meer, kan iemand langskomen*?"

Klant B stuurt precies hetzelfde bericht.

De AI Agent herkent de Intent ("*Maken van een reparatieafspraak*") in beide gevallen correct. De AI Agent heeft perfect AI-bruikbare kennis over het reparatieafspraak-proces. En toch moet het antwoord voor Klant A en Klant B fundamenteel anders zijn:

Factor	**Klant A**	**Klant B**
Aankoopdatum	8 maanden geleden	4 jaar geleden
Product	Premium-model	Budget-model
Garantiestatus	Binnen garantie	Buiten garantie
Aanvullende garantie	n.v.t.	Niet afgesloten
Plaats	Nederland	Duitsland (andere serviceprovider)

Het juiste antwoord voor Klant A: "*Geen kosten. We plannen binnen 2 werkdagen een monteur.*" Het juiste antwoord voor Klant B: "*Kosten vanaf €85 voorrijden plus materialen. Buiten Nederland werken we met een partner — ik verbind u door.*"

Dezelfde vraag. Dezelfde perfecte kennis. Compleet ander antwoord. Het verschil zit in context.

Context is de derde factor

Een succesvolle AI-Triage heeft drie factoren nodig, niet twee:

1) **Kennis**: Wat kun je doen in deze situatie?
2) **Context:** Welke situatie is dit precies?
3) **Actie**: Wat is dan de juiste vervolgstap?

Als één van de drie ontbreekt of onjuist is, faalt het systeem. In de praktijk gaat het meestal mis op context, omdat kennis en actie relatief goed te expliciteren zijn, maar context impliciet wordt aangenomen.

Vijf soorten context die er (bijna) altijd toe doen

Voor de meeste CS-Intents zijn dit de contextlagen die je expliciet moet adresseren:

Type context	Voorbeeld	Vraag die de AI moet kunnen beantwoorden
Klantcontext	Klantsegment, locatie, taalvoorkeur, historie	"Met wie hebben we te maken?"
Productcontext	Product, versie, aankoopdatum, garantiestatus	"Waar gaat dit over?"
Procescontext	Lopende orders, openstaande tickets, eerder contact	"Wat speelt er al?"
Tijdcontext	Werkdag/weekend, kantooruren/erbuiten, seizoen	"Wanneer is dit?"
Kanaalcontext	Chat/WhatsApp/telefoon, first response vs follow-up	"Hoe en via welk kanaal?"

Een kennisartikel dat reageert op "Reparatieafspraak boeken" moet al deze contextlagen kunnen gebruiken. Daarom hoort in een AI-bruikbaar artikel een expliciete sectie "**Context die ik moet hebben voordat ik dit artikel gebruik**":

> **Voor deze flow opvragen:**
>
> - Productnaam en aankoopdatum (uit ordersysteem)
> - Garantiestatus inclusief aanvullende garantie (uit klantdossier)
> - Landcode van de klant (uit klantdossier)
> - Lopende reparatietickets (uit ticketsysteem)

Als één van deze velden ontbreekt, is de regel: niet **gokken, maar vragen of overdragen**. Een AI Agent die netjes vraagt "*Om u goed te kunnen helpen, kunt u me vertellen wanneer u het*

apparaat heeft gekocht?" is oneindig veel beter dan een AI Agent die aanneemt "*Laten we ervan uitgaan dat het binnen garantie valt.*"

Wat dit betekent voor je kennisbank

Je kennis AI-bruikbaar maken is niet alleen een herschrijfoefening. Het is ook een **context-inventarisatie**: welke contextvelden zijn cruciaal voor welke Intent, en hoe komt de AI Agent aan die informatie? Als die informatie er niet is, weet je dat je:

- óf in je data-architectuur moet investeren (context ophaalbaar maken)
- óf de AI moet instrueren om actief te vragen naar de missende context (en bij twijfel over te dragen).

Een kennisartikel zonder contextbeschrijving is een half-werkend artikel. En een half-werkend artikel leidt tot fouten die moeilijk te debuggen zijn, omdat het kennisartikel op papier "goed" is.

Eén kennisbank, AI-first, ook bruikbaar voor mensen

Een van de vragen die elke CS-manager krijgt bij het AI-proof maken van zijn kennis: "*Moeten we dan twee kennisbanken bijhouden? Eén voor mensen, één voor AI?*"

Mijn antwoord is helder: **Nee**. Eén goed-gestructureerde kennisbank, geschreven voor AI, ook bruikbaar voor mensen. Dat is het doel.

Waarom twee kennisbanken een slecht idee is

Redundantie leidt onvermijdelijk tot inconsistentie. Als je een kennisartikel over "*Reparatieafspraak boeken*" op twee plekken onderhoudt, gebeurt het volgende:

- De kennisbeheerder past versie A aan, vergeet versie B.
- Een nieuwe uitzondering wordt alleen in versie B gedocumenteerd.
- Een verouderd stuk informatie blijft in versie A staan.

In een customer service context vertaalt dit zich direct naar fouten richting klanten, en erger: naar inconsistente fouten, waarbij Klant A een ander antwoord krijgt dan Klant B omdat de AI versie A gebruikte en de agent versie B. Je bent dan systematisch je eigen reputatie aan het ondermijnen.

Er is nog een tweede reden. Als je kennis ontwerpt voor twee doelgroepen (mensen + AI), ga je compromissen sluiten. Het wordt een beetje mensvriendelijk en een beetje AI-bruikbaar. Resultaat: niet goed voor beide.

Waarom één AI-first kennisbank werkt voor mensen

Een goede AI-first kennisbank is voor mensen in eerste instantie *minder prettig* om te lezen. Dat is eerlijk. De modulaire opbouw, expliciete beslisregels, gestandaardiseerde structuur; het leest niet als een verhaal.

Maar ervaren CS-teams die ermee werken, rapporteren na een paar weken iets onverwachts: ze zijn productiever dan voorheen. Waarom?

- **Snelheid**. Ze hoeven niet meer door 8 alinea's lopende tekst om te vinden wat ze zoeken. De structuur wijst het aan.
- **Betrouwbaarheid**. Expliciete beslisregels laten geen ruimte voor "*wat zou deze collega bedoeld hebben*".
- **Consistentie in het team**. Nieuwe agents en ervaren agents werken nu met dezelfde expliciete regels in plaats van dezelfde impliciete tradities.

Het gevoel van "*dit leest niet fijn*" verdwijnt binnen een week. Wat overblijft is een team dat minder twijfelt en minder fouten maakt.

Het is een transitie, geen schakelaar

Eerlijkheid verplicht: je gaat niet op een vrijdag beslissen en maandag met één AI-first kennisbank werken. De praktijk is:

- **Je kiest een eerste Intent** (bv. "Opvragen orderstatus") en bouwt daarvoor de AI-bruikbare versie.
- **Je vervangt stap voor stap** de mensvriendelijke versies. Sommige Intents zijn eenvoudig, andere complex.
- **Tijdens de transitie** heb je *de facto* een mix; sommige artikelen al AI-first, andere nog mensvriendelijk. Dat is acceptabel, zolang je weet welke welke is en daarnaar handelt.
- **Je eindstaat** is één kennisbank waar elk artikel AI-bruikbaar is. Het moment waarop je daar bent, is niet van tevoren te plannen, het is afhankelijk van hoe groot je kennisbank is en hoeveel capaciteit je hebt voor herschrijfwerk.

Reken op **6 tot 24 maanden** voor een middelgrote organisatie. Sommige organisaties halen dit nooit helemaal, en dat is oké; je hoeft ook niet alle 8.000 artikelen om te bouwen. Je richt je op de kennis die je AI Agents gaan raken, en laat de rest in zijn huidige vorm.

De praktische regel voor de overgang

Gedurende de transitie: **voor elke nieuwe Intent die je met AI wilt afhandelen, moet de bijbehorende kennis AI-bruikbaar zijn vóór je live gaat**. Ga nooit live met een AI Agent die half AI-bruikbare, half mensvriendelijke kennis moet gebruiken. Het gaat fout op de mensvriendelijke helft en je weet pas waar als je de chaos al hebt aangericht.

Hoe bouw je mens-kennis om naar AI-kennis?

Genoeg theorie. Hoe doe je dit in de praktijk? In deze sectie: een stappenplan, gevolgd door een uitgewerkt voor/na-voorbeeld.

Stappenplan voor één kennisartikel

Stap 1: Scope en Intent bepalen.

Welk concreet probleem helpt dit artikel oplossen? Als het antwoord "meerdere dingen" is, heb je al je eerste signaal: splits het artikel in meerdere. Eén artikel = één of een klein aantal gerelateerde Intents.

Stap 2: Contextvelden inventariseren.

Welke context moet bekend zijn voordat dit artikel bruikbaar is? Klantsegment, product, garantiestatus, lopende zaken, tijd, kanaal? Noteer ze expliciet, inclusief de bron (welk systeem, welke API).

Stap 3: Beslisregels expliciteren.

Herschrijf elke beslisregel in "als X, dan Y, anders Z"-vorm. Waar nu "ervaren agents weten wat te doen" staat, moet straks een expliciete regel staan. Dit is meestal de pijnlijkste stap, omdat je ontdekt hoeveel impliciete kennis er in je team zit.

Stap 4: Acties concretiseren.

Welke concrete handeling volgt op elke uitkomst? "Plan een afspraak" is te vaag; "Roep flow REPAIR_WARRANTY aan met productId en aankoopdatum" is concreet. Noem expliciet de systemen, de flows, de datavelden.

Stap 5: Uitzonderingen en handoff-momenten benoemen.

Wanneer moet de AI Agent stoppen en overdragen aan een mens? Bij welke twijfel, welke fout, welke context-afwezigheid? Dit is waar veel artikelen te kort zijn. "Als u er niet uit komt, escaleer" is geen handoff-regel; het is een wens.

Stap 6: Metadata toevoegen.

Scope, gerelateerde Intents, versienummer, eigenaar, laatste review-datum, testgeschiedenis. Dit is wat je kennisbank onderhoudbaar maakt.

Stap 7: Testen.

Zet het artikel in een testomgeving en run er realistische casussen doorheen. Zowel de standaardgevallen als de randgevallen. Wat de AI Agent fout doet, is zelden zijn schuld; het is bijna altijd een gat in je kennis. Vul dat gat en test opnieuw.

Voor/na-voorbeeld: "Reparatieafspraak boeken" bij TechNova Retail

Vóór: de mensvriendelijke versie:

> **Reparatieafspraak maken**
>
> *Laatst gewijzigd: 14 maart 2025*
>
> Een klant kan een reparatieafspraak maken voor een defect product. Check eerst de garantiestatus en handel daarna overeenkomstig.
>
> Bij producten binnen garantie is het eenvoudig: je plant een afspraak via ons reguliere proces en er zijn geen kosten voor de klant. Zorg wel dat je de bevestiging verstuurt.
>
> Voor producten buiten garantie moet je eerst een kostenindicatie geven. Onze monteurs rekenen een vast voorrijtarief plus eventuele materiaalkosten. Als de klant akkoord gaat, plan je de afspraak. Anders verwijs je door naar onze webshop voor een vervangend product.
>
> Let op: voor klanten in België en Duitsland werken we met een externe partner. Geef deze klanten het telefoonnummer van de partnerservice mee.

Dit artikel werkt prima voor een mens. Het heeft een logisch verhaal, lekker te lezen, vier alinea's. Een ervaren agent past het feilloos toe.

Ná — de AI-bruikbare versie:

> **Flow: Reparatieafspraak boeken**
>
> *Scope: producten gekocht bij TechNova Retail*
>
> *Gerelateerde Intents: "Maken reparatieafspraak", "Melden defect"*
>
> *Versie: 2.1, Eigenaar: CS Knowledge Team, Laatste review: 2026-04-23*
>
> **Context die nodig is voor gebruik:**
>
> - `product_id` (uit ordersysteem, veld: order.product_id)
> - `purchase_date` (uit ordersysteem, veld: order.date)
> - `warranty_status` (berekend: zie beslisregel 1)
> - `country_code` (uit klantdossier, veld: customer.country)
> - `extended_warranty_active` (uit klantdossier, veld: customer.extended_warranty)
>
> Als één van deze velden ontbreekt: **hand-off naar agent met reden "missing_context"**.
>
> **Beslisregel 1: Garantiestatus bepalen:**
>
> - `purchase_date` < 24 maanden geleden → status = "in_warranty"
> - `purchase_date` 24–60 maanden geleden EN `extended_warranty_active` = true → status = "extended_warranty"
> - `purchase_date` 24–60 maanden geleden EN `extended_warranty_active` = false → status = "out_of_warranty"
> - `purchase_date` > 60 maanden geleden → status = "out_of_warranty"

Beslisregel 2: Land bepalen:

- `country_code` = "NL" → route = "internal"
- `country_code` in ["BE", "DE"] → route = "partner_handoff"
- Andere landen → **hand-off naar agent met reden "unsupported_country"**

Actie per combinatie:

Status	Route	Actie
in_warranty	internal	Roep flow REPAIR_WARRANTY aan. Bevestig afspraak per e-mail (template: repair_confirmation_nl_warranty). Geen kostenvermelding.
extended_warranty	internal	Roep flow REPAIR_EXTENDED aan. Bevestig afspraak per e-mail (template: repair_confirmation_nl_extended). Vermeld eigen risico €25.
out_of_warranty	internal	Roep flow REPAIR_QUOTE aan. Verstuur kostenindicatie: voorrijdtarief €85 + materialen in nacalculatie. Wacht op akkoord klant (max 48u) voor afspraak. Bij weigering: verwijs naar webshop URL.
—	partner_handoff	Geef klant partnertelefoonnummer door: NL-Partnerservice voor BE (+32 XX XX XX XX), DE-Partnerservice voor DE (+49 XX XX XX XX). Sluit gesprek af met hand-off code "partner_referred".

Handoff-momenten:

- Context ontbreekt (zie hierboven)
- Klant geeft aan "geen standaardreparatie" (bijv. waterschade, vallenschade) → hand-off met reden "complex_damage"
- Klant is emotioneel of klagend → hand-off met reden "emotional_escalation"
- Klant is B2B (customer.segment = "business") → hand-off met reden "b2b_customer"

Testgevallen (onderhouden in aparte test-suite):

- TG-01: NL-klant, 8 maanden oud product → in_warranty / REPAIR_WARRANTY
- TG-02: NL-klant, 3 jaar oud product zonder extended → out_of_warranty / REPAIR_QUOTE
- TG-03: NL-klant, 3 jaar oud product met extended → extended_warranty / REPAIR_EXTENDED

> - TG-04: BE-klant, willekeurige ouderdom → partner_handoff
> - TG-05: Frankrijk-klant → hand-off unsupported_country
> - TG-06: B2B-klant, binnen garantie → hand-off b2b_customer

Wat valt op?

- **De AI-versie is veel langer**. Dat klopt; expliciet is altijd langer dan impliciet.
- **De AI-versie is eenduidig leesbaar voor** mensen. Een CS-agent die deze flow opent, weet precies wat hij moet doen. Misschien minder fijn dan de lopende tekst, maar er is geen ruimte voor interpretatie.
- **Uitzonderingen zijn vooraf benoemd**. Waterschade, emotie, B2B; stuk voor stuk momenten waar de oorspronkelijke versie stilzwijgend overheen stapte.
- **Er is testbaarheid**. De zes testgevallen vormen een contract: als dit artikel slaagt voor TG-01 t/m TG-06, weten we dat het werkt voor de hoofdgevallen.

Dit is het werk. Niet sexy, niet snel, maar het is wat het verschil maakt tussen een AI-pilot die werkt en eentje die een excuus zoekt om wéér uitgesteld te worden.

Template: Knowledge Readiness Checklist

Gebruik deze checklist per kennisartikel óf per Intent (aggregeer dan over de artikelen die aan die Intent gekoppeld zijn). Het doel is niet een perfecte score, maar zichtbaar maken waar je kennis tekortschiet.

Scoor elk criterium 0, 1 of 2 punten:

- 0 = ontbreekt volledig of is onbruikbaar
- 1 = aanwezig maar onvolledig
- 2 = aanwezig en AI-bruikbaar

#	Criterium	Vraag	Score (0/1/2)
1	Scope	Is de scope van het artikel expliciet benoemd (welke Intents, producten, klantsegmenten)?	
2	Zelfstandige leesbaarheid	Kan een AI Agent dit artikel gebruiken zonder naar andere artikelen te hoeven kijken?	
3	Contextvelden	Staan alle benodigde contextvelden expliciet genoemd, inclusief bron?	
4	Expliciete beslisregels	Zijn alle beslisregels in als/dan-vorm opgeschreven, zonder impliciete aannames?	
5	Concrete acties	Staat er per uitkomst een concrete handeling (welke flow, welk systeem, welke template)?	
6	Uitzonderingen	Zijn de relevante uitzonderingen expliciet benoemd, inclusief de actie die dan volgt?	
7	Handoff-momenten	Is duidelijk wanneer en waarom de AI Agent moet overdragen aan een mens?	
8	Gestandaardiseerd taalgebruik	Is jargon vermeden of gedocumenteerd in een woordenlijst?	
9	Structurele metadata	Heeft het artikel scope-, Intent-, versie- en eigenaar-metadata?	
10	Testbaarheid	Zijn er concrete testgevallen gedefinieerd waarmee je kunt valideren of het artikel werkt?	

Maximale score: 20.

Score	Betekenis
18–20	AI-bruikbaar. Klaar voor gebruik in een pilot.
14–17	Bijna klaar. Eén of twee gaten dichten, dan bruikbaar.
10–13	Partieel. Bruikbaar als basis, maar expliciet herschrijven nodig voordat je er een AI Agent op zet.
6–9	Onvoldoende. Dit artikel is in huidige vorm ongeschikt. Herschrijven volgens het stappenplan.
0–5	Mens-kennis. Wat hier staat is misschien goed voor mensen, maar het moet vanaf nul opnieuw voor AI.

Wat te doen met de uitkomst

De verleiding is groot om na de eerste scoring te denken: "*Oké, we moeten alle artikelen naar 20*." Dat is meestal te ambitieus en niet eens nodig.

Praktischer is:

- **Scoor alleen de kennis die bij je kandidaat-Intents hoort** (uit de eerdere Intentselectie). Beginnen bij alle 8.000 artikelen herschrijven is een excuus om nooit te beginnen.
- **Sorteer de resultaten op kansrijkheid**: welke artikelen kunnen met beperkt werk naar 18+? Begin daar.
- **Laat artikelen met score 6 of lager liggen** totdat je fundament staat. Die Intents zijn niet de eerste AI-kandidaten.

Kennisonderhoud in een AI-wereld

In een klassieke setting is kennisbeheer meestal een bijbaan: een redactieteam dat eens per kwartaal een review doet, ervaren agents die incidenteel een correctie aandragen, een content-owner die iets bijhoudt naast zijn hoofdtaak.

In een AI-wereld werkt dat niet meer. Verouderde of slechte kennis leidt direct tot foute antwoorden aan klanten, op schaal, 24/7. Je kunt niet meer wachten op een kwartaalreview. Kennisonderhoud wordt een **continue discipline**.

Vier principes voor AI-kennisonderhoud

1. **Eigenaarschap per artikel, niet per kennisbank.** Elk artikel heeft een duidelijke eigenaar, bij voorkeur iemand die de inhoud ook echt kent (een senior agent, een teamleider, een productmanager). De eigenaar is verantwoordelijk voor correctheid, actualiteit en performance van dát specifieke artikel. Eén persoon die "de kennisbank" bezit, is te algemeen.
2. **Feedback-loops als eerstegraads instrument.** Als een AI Agent een fout maakt, moet die fout traceable zijn tot één of meer kennisartikelen. Dan kan de eigenaar corrigeren. Dit vraagt technische voorzieningen: logging van welk artikel in welke interactie is gebruikt, rapportage van fouten gekoppeld aan artikelen, een laagdrempelig proces om aanpassingen door te voeren. Zonder deze loop leer je niet, en leert je AI niet. Platforms verschillen sterk in hoe goed ze dit ondersteunen; dit hoort onderdeel te zijn van je platformkeuze.
3. **Versiebeheer en rollback.** Elke aanpassing aan een artikel is een versie. Je moet kunnen terugdraaien: "sinds versie 3.4 geeft deze agent verkeerde antwoorden; terug naar 3.3, onderzoek wat er mis is met 3.4." Zonder versiebeheer vertrouw je op geluk.
4. **Testgevallen als onderhoudsmateriaal.** De testgevallen uit het voor/na-voorbeeld (TG-01 t/m TG-06) zijn niet eenmalig. Elke wijziging in het artikel moet opnieuw door de tests. Dit klinkt zwaar, maar is in de praktijk vaak minder werk dan een fout in productie repareren.

Reviewcyclus: van kwartaal naar continu

In een mensvriendelijke wereld is een kwartaalreview acceptabel. In een AI-wereld moet je denken in drie reviewfrequenties naast elkaar:

Trigger	Wanneer	Door wie
Event-getriggerd	Bij een kennisfout of klantklacht die terug te voeren is op een artikel	Eigenaar, binnen 48 uur

Metric-getriggerd	Als prestatie-indicatoren dalen (meer handoffs, lagere CSAT/CES op specifieke Intent)	Eigenaar + AI-manager, binnen 1 week
Cyclisch	Vast ritme (bijv. maandelijks voor hoog-volume artikelen, kwartaal voor laag-volume)	Kennisteam, gepland

Wat nu?

Je hebt nu een beeld van wat AI-bruikbare kennis inhoudt, waarom context het meest onderschatte aspect is, hoe je mens-kennis omzet naar AI-kennis, en hoe je je bestaande kennisbank scoort. De vraag waarmee je dit hoofdstuk afsluit: wat moet je nu concreet doen?

Dat hangt af van de uitkomst van je Knowledge Readiness Checklist. Grofweg drie paden:

Pad A — Meeste kandidaat-artikelen scoren 14–20

Je bent in goede vorm. Focus op de laatste 20% kwaliteitsslag op je kandidaat-artikelen, start met testgevallen voor je pilot-Intent, en begin klein. Je bent niet het fundament aan het leggen; je bent aan het bouwen. Ga door naar het volgende hoofdstuk voor de technologiekant.

Pad B — Kandidaat-artikelen scoren gemengd 8–17

Dit is de meest voorkomende situatie. Pak een eerste kandidaat-Intent, herschrijf de bijbehorende kennis volledig naar AI-bruikbaar (volg het stappenplan), en valideer het op testgevallen. Laat de rest van de kennisbank voorlopig staan. Je pilot wordt de testcase voor hoe je de transitie inricht.

Realistische planning: voor een middelgrote organisatie betekent dit 4–8 weken om de kennis voor één Intent op niveau te brengen, voordat de pilot technisch kan starten. Veel organisaties onderschatten dit en beginnen te bouwen voordat de kennis staat. Vermijd dat.

Pad C — Kandidaat-artikelen scoren overwegend onder 10

Je fundament staat nog niet. Dat is geen ramp, maar het is wel realiteit. Een AI-Triage pilot nu starten is vrijwel zeker een mislukking — niet door de AI, maar omdat er geen kennis is om op te varen.

Drie dingen om te doen:

1) **Accepteer** dat je eerst een kennisfundament moet bouwen voordat AI-Triage realistisch is. Dat is geen omweg; het is de weg.

2) **Start een beperkt kennisprogramma** waarin je top-5 à top-10 kansrijkste Intents krijgen de AI-bruikbare behandeling. 3–6 maanden werk, afhankelijk van je capaciteit.
3) **Gebruik het werk als tussenproduct.** Ook zonder AI profiteren je menselijke agents van expliciete beslisregels, duidelijke context en testbaar artikelen. Dit is nooit weggegooid werk.

Daarna terug naar de checklist en opnieuw scoren.

Aan het eind van dit hoofdstuk heb je:

- Begrip van waarom kennis fundamenteel is voor AI, en wat mens-kennis ongeschikt maakt.
- Een mentaal model van hoe AI kennis gebruikt (chunking, retrieval, hallucinatie).
- De zes verschillen tussen mens- en AI-kennis, met voor/na-voorbeelden.
- Begrip van context als de derde, vaak vergeten factor.
- Een duidelijk standpunt over één versus twee kennisbanken.
- Een stappenplan en uitgewerkt voorbeeld om je kennis om te bouwen.
- Een checklist om je huidige kennis te scoren.
- Een concrete aanpak voor kennisonderhoud in een AI-wereld.
- Drie paden vooruit, afhankelijk van waar je nu staat.

In het volgende Hoofdstuk stappen we van kennis naar technologie: platformen, integraties, en het Capability Framework waarmee je beoordeelt of een platform geschikt is voor jouw ambities.
