

איך בודקים AI agents כשאינ benchmark: כותבים קודם את התשובה

Nathanael Weill מ-Air Canada מסביר איך הופכים את בעיית ה-evaluation: מתחילים
מתשובה ידועה, מייצרים ממנה את הקלט ובודקים אם ה-agent מגיע אליה

Nathanael Weill · AI Engineer · 2026-10-22

משך: כ-20 דקות · YouTube

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זו הרצאה של כ-20 דקות מכנס AI Engineer, לפי השקפים מושב breakout בניו יורק ב-13 באוקטובר 2026 בחסות Arize. מרצה Nathanael Weill, שמוביל צוות שבונה AI agents ב-Air Canada. הבעיה שהוא מציג: המשימות בארגון ספציפיות מדי בשביל שיהיה להן benchmark ציבורי, ואנשי התוכן שיכולים לבדוק תשובות עסוקים ונגישים רק לעתים רחוקות.

הטענה המרכזית: במקום לחכות שמומחה ידרג את פלט ה-agent, כותבים קודם את התשובה, מייצרים ממנה את הקלט ובודקים בקוד אם ה-agent מצא אותה. הוא מראה איך הגישה הזאת העבירה פרויקט RAG קודם מבדיקה אחת בשבוע לעשרות בדיקות ביום, ואיך יישם אותה ב-agent לסקירת חוזי ספקים בעזרת חמש טכניקות לבניית test set. הוא גם מסביר למה הוא מעדיף בדיקה בקוד על פני LLM judge בכל מקום שזה אפשרי.

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות, הראיות והפערים.
- 2. **הלקח מפרויקט ה-RAG:** ארבעה צוותים ומשימה אחת, ולמה מחזור משוב של שבוע תקע את ההתקדמות.
- 3. **להפוך את הבעיה:** כותבים את התשובה קודם, ועוברים מבדיקה אחת בשבוע לעשרות ביום.
- 4. **ה-agent לסקירת חוזים:** המשימה ב-Air Canada וארבעת השלבים של ה-agent.
- 5. **חמש דרכים לבנות test set:** מה עבד, מה לא, ומה עלה בכל אחת.
- 6. **בדיקה בקוד או LLM judge:** סולם הבדיקות ולולאת המשוב מהמומחים.
- 7. **מה לעשות עכשיו:** צעדים מעשיים ושאלות פתוחות.
- 8. **מילון מונחים**

הערות מקור המסמך הופק אוטומטית מהתמלול ומהשקפים של ההקלטה. ייתכן ש-ASR שיבש שמות ומונחים: בתמלול מופיע למשל "clothes" במקום clauses ו-"padding" במקום pattern. הציטוטים מועתקים מילה במילה ובבדקו מול חותמת הזמן שלהם. לא נוסף מידע ממקורות חיצוניים.

1. תקציר מנהלים

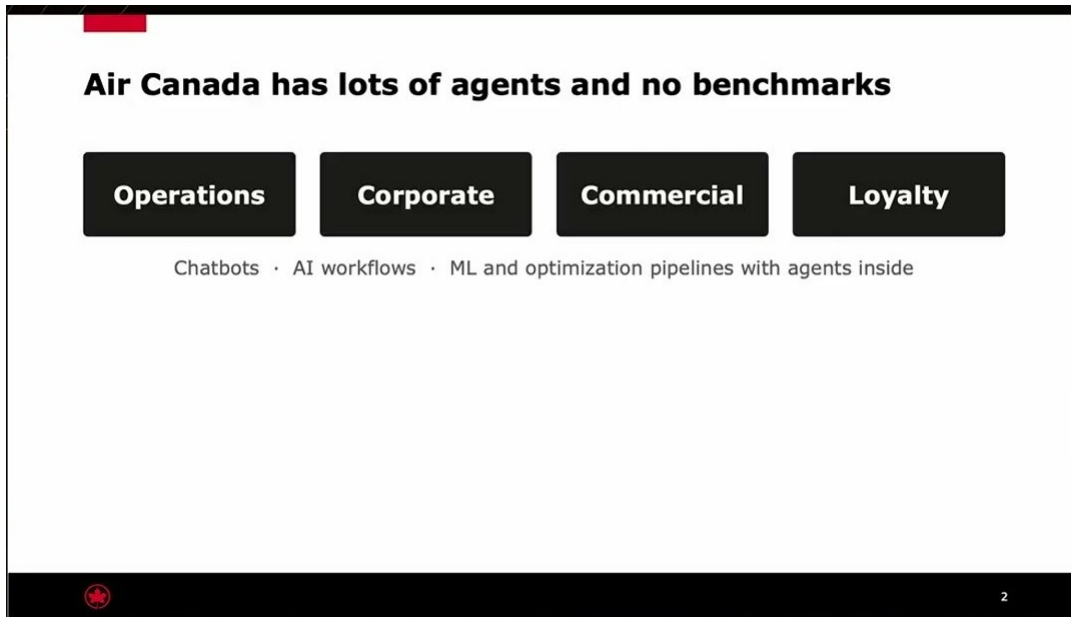
ההרצאה עוסקת במצב שכל צוות AI ארגוני מכיר: אין benchmark, למומחים אין זמן, ולכן כל שינוי ב-agent נבדק לאט או לא נבדק בכלל. Weill מציע מנגנון אחד בכמה וריאציות: מתחילים מהתשובה הנכונה, מייצרים סביבה קלט ומודדים בקוד אם ה-agent מצא אותה.

- **בלי יכולת לדרג אין שיפור:** זה הלקח שהוא לקח מפרויקט Q&A לטלקום ב-2024, שבו הצוות שלו הצליח מול שלושה צוותים מתחרים.
- **המגבלה הייתה מחזור המשוב, לא הרעיונות:** שינוי ב-agent לקח דקות, אבל הבדיקה דרשה ישיבת מומחים שבועית, כלומר ניסיון אחד בשבוע.
- **להוסיף מורכבות זה לא בהכרח לשפר:** טכניקות מהמאמרים לא עזרו, וכל תוספת שברה משהו שעבד קודם.
- **להפוך את הבעיה:** מייצרים שאלה מתוך עמוד ידוע ובודקים בקוד אם ה-retrieval מחזיר את העמוד. כך עברו מ-evaluation אחד בשבוע לעשרות ביום.
- **פישוט עבד יותר מהוספה:** מה שהעלה את הדיוק היה בחירה נכונה של embedding model, של chunking ושל BM25-alpha.
- **בחוזים שותלים את ההפרה:** כותבים clause שמפר כלל ידוע ובונים חוזה סביבו. כך נוצרו 275 בדיקות שהתשובה עליהן ידועה.
- **הסרת clauses היא ה-benchmark הקשה:** היא חשפה שה-agent מפספס את רוב ה-clauses שהוסרו.
- **מארבעה שבועות לחמש דקות:** בשיטת ה-templates הזמן לקבלת משוב ירד מארבעה שבועות לסקריפט שרץ בחמש דקות.
- **LLM judge רק כשאין ברירה:** הרעש של ה-judge היה גדול מהרעש של ה-agent עצמו.

מה לא נאמר

- **אין מספרי דיוק:** לא הוצגו אחוזים לא לפרויקט ה-RAG ולא ל-agent החוזים. גרף ה-"where we want to be" בשקף הוא איור ולא נתון.
- **חוסר עקביות על הטכניקה החזקה ביותר:** את הטכניקה החמישית (templates) הוא מכנה "the most powerful", אבל גם על הסרת clauses הוא אומר "the most powerful approach that we had today".
- **אין מדד ל-coverage:** הוא מודה שאי אפשר להבטיח כיסוי של כל הכללים ושחלק מהם הצוות המציא בעצמו, אבל לא אומר איך מודדים כמה מהמציאות ה-test set מכסה.
- **ההצלחה בטלקום לא מוסברת:** הוא אומר במפורש "I don't know why we actually made it work". ההשערה שלצוותים האחרים לא היה test set לא אומתה.
- **אין מספרים על הייצור:** לא נאמר אם agent החוזים כבר בשימוש, באיזה היקף, ומה שיעור ה-false positives.

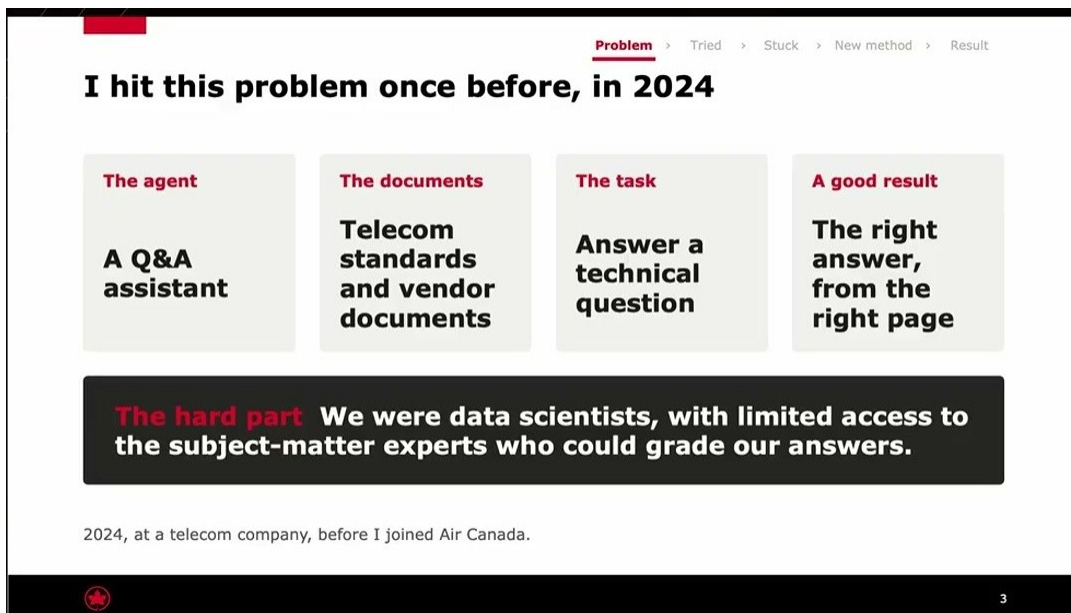
2. הלקח מפרויקט ה-RAG: כשמחזור המשוב הוא שבוע



ב-Air Canada יש הרבה agents ואין benchmarks. ארבעה תחומים (Operations, Corporate, Commercial, Loyalty) שכוללים chatbots, AI workflows ו-pipelines של ML ואופטימיזציה

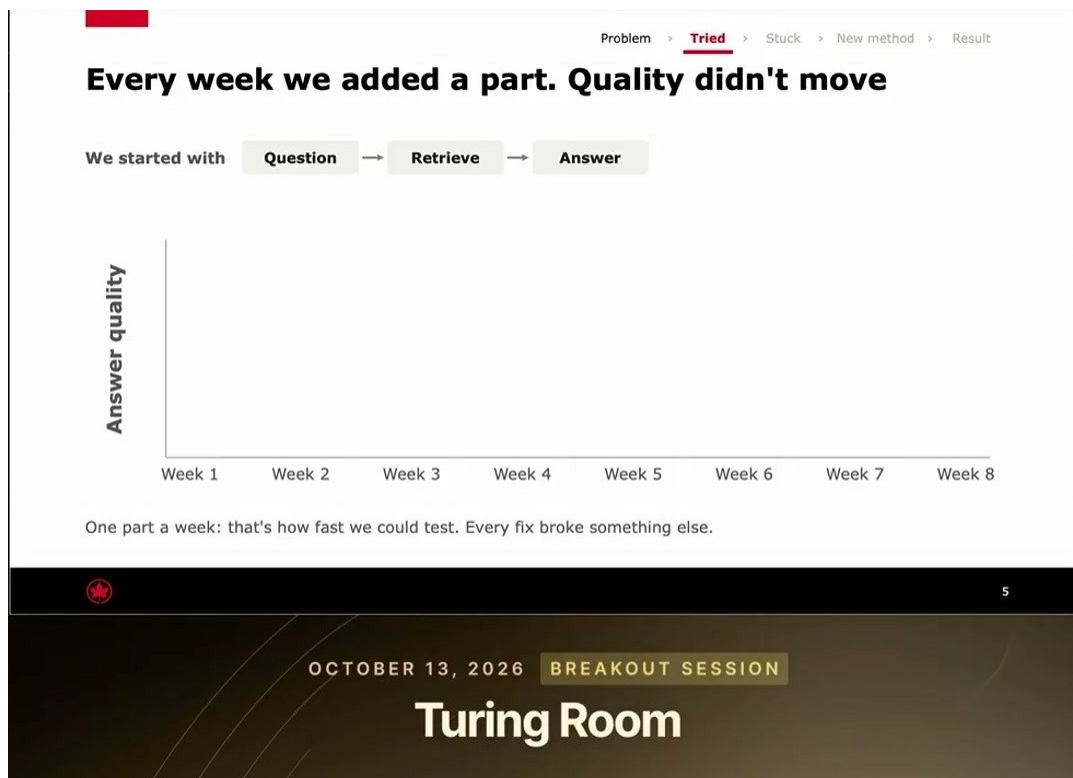
Weill הצטרף ל-Air Canada ביוני השנה. הצוות שלו מספק פתרונות AI לארבעה תחומים, ולכל אחד אתגרים משלו. יש שתי בעיות שמחזקות זו את זו: אין benchmark ציבורי למשימות כאלה, וחברי הצוות הם data scientists ו-engineers, לא אנשי תוכן. המומחים יושבים ביחידות העסקיות, עסוקים בעבודה שלהם, והגישה אליהם מוגבלת.

את אותה בעיה הוא פגש קודם בחברה אחרת, ב-Q&A system למהנדסי טלקום: כ-10,000 מסמכים ועוד מסמכי ספקים, וכדי לענות צריך לדעת באיזה מסמך לחפש. היו עוד שלושה צוותים, שניים חיצוניים ואחד פנימי, שעבדו על אותה משימה עם אותם מסמכים, אחד אחרי השני. הפתרון של הצוות שלו היה היחיד שהמשתמשים באמת אימצו.



הבעיה מ-2024: עוזר Q&A מעל תקני טלקום ומסמכי ספקים. הקושי היה שהצוות היה data scientists עם גישה מוגבלת למומחים שיכלו לדרג את התשובות

לפי Weill, ב-RAG מה שקובע הוא ה-90% retrieval: "if you found the right page or the right documents, 90% of the time you have actually the right answer". הצוות בנה pipeline שהלך והסתבר וניסה טכניקות מהמאמרים, והן לא שיפרו את התוצאה. גרוע מזה, כל תוספת שברה משהו שעבד קודם.

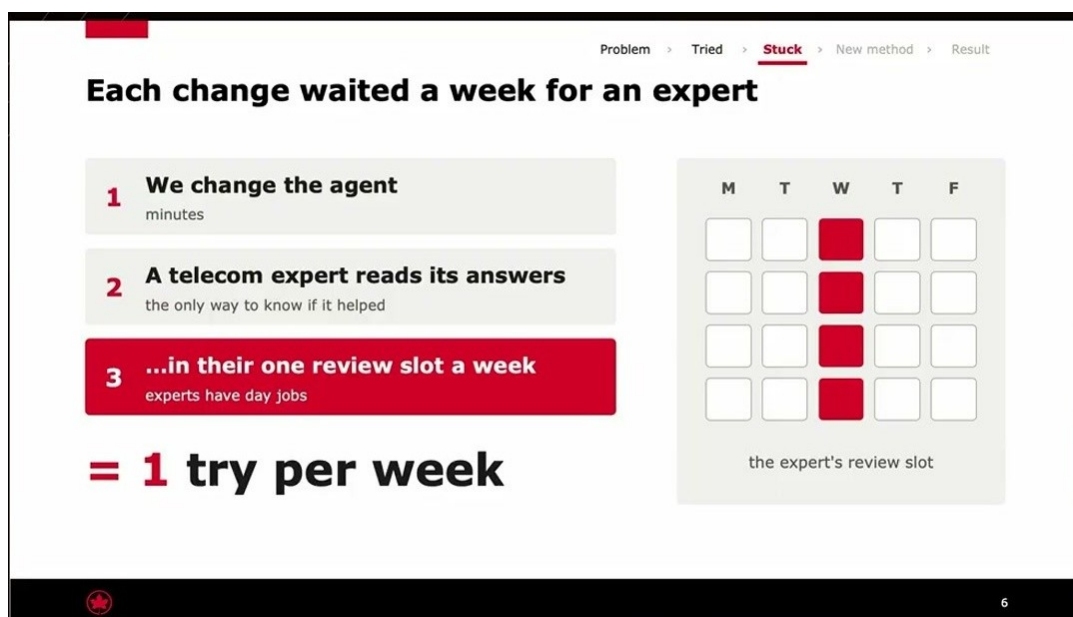


"Every week we added a part. Quality didn't move": כל שבוע נוסף רכיב ל-pipeline (Question → Retrieve → Answer) ואיכות התשובות נשארה על אותו קו לאורך שמונה שבועות

Nathanael Weill So complexity doesn't always help —

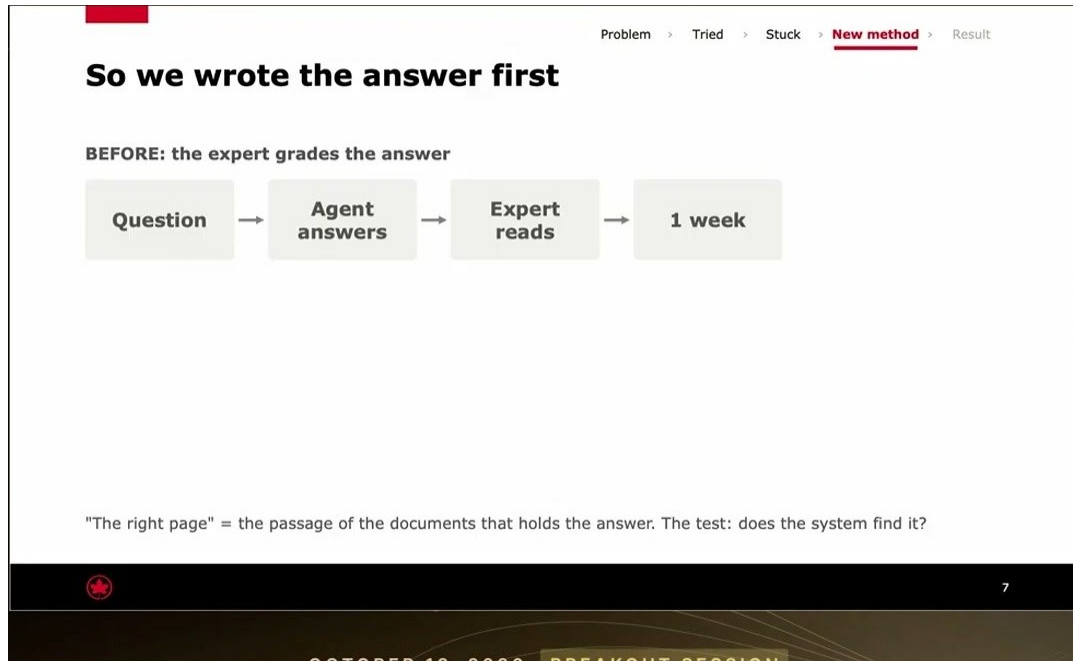
הבעיה השנייה הייתה קצב הבדיקה. שינוי ב-agent, למשל הוספת query rewriting או שלב reasoning, לקח דקות. את הדירוג עשו פעם בשבוע: מסך גדול עם חמישה chatbots, אותה שאלה לכולם, ומומחים שבוחרים איזו תשובה הם מעדיפים. לפעמים גם המומחים לא ידעו את התשובה והיו צריכים לחפש אותה.

Nathanael Weill It's very hard when you put two experts in the room, you get three opinions —



"Each change waited a week for an expert": שינוי ב-agent לוקח דקות, המומחה מדרג תשובות כשיש לו זמן, וחלון הבדיקה הוא פעם בשבוע. התוצאה: ניסיון אחד בשבוע

3. להפוך את הבעיה: כותבים את התשובה קודם



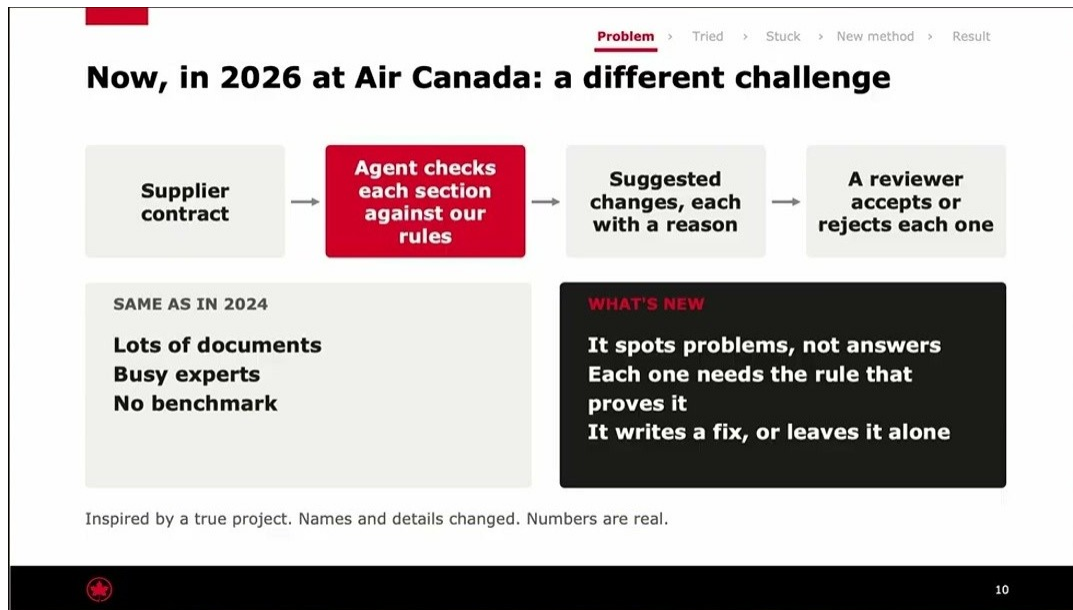
"So we wrote the answer first": במקום $Question \rightarrow Agent\ answers \rightarrow Expert\ reads$ שבוע, מתחילים מה-"right page", כלומר הקטע במסמכים שמכיל את התשובה, ובודקים אם המערכת מוצאת אותו

הפתרון, ש-Weill מציין במפורש שהוא לא המציא: לוקחים עמוד, מבקשים מ-LLM לכתוב עליו שאלה (משימה קלה), מריצים את ה-agent ובודקים בקוד אם הוא שלף את העמוד שממנו נוצרה השאלה. בבדיקה כזו אין מקום לדעות, וכולה אוטומטית.

התוצאה: "we went from one evaluation a week to dozens a day", והציונים שהמומחים נתנו עלו משבוע לשבוע. מה שעזר בעיקר היה הסרת מורכבות וכוונון של היסודות: chunking, embedding model וה-alpha-BM25. ההשערה שלו לגבי שאר הצוותים היא שלא היה להם test set, והם נשארו בלולאה הארוכה מול המומחים.

Nathanael Weill If you can't grade the agent, you can't make it better —

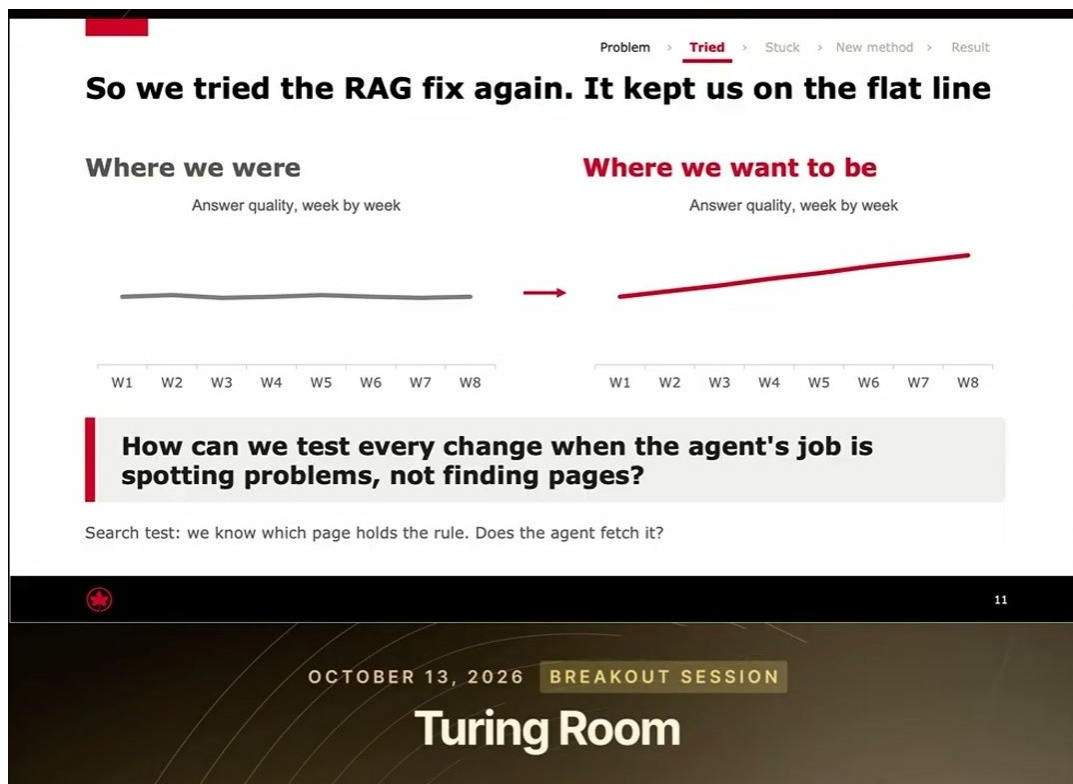
4. ה-agent לסקירת חוזי ספקים



האתגר ב-Air Canada ב-2026: חוזה ספק נכנס, ה-agent בודק כל סעיף מול הכללים ומציע שינויים עם נימוק, ו-reviewer מאשר או דוחה כל אחד. בתחתית השקף: "Names and details changed. Numbers are real"

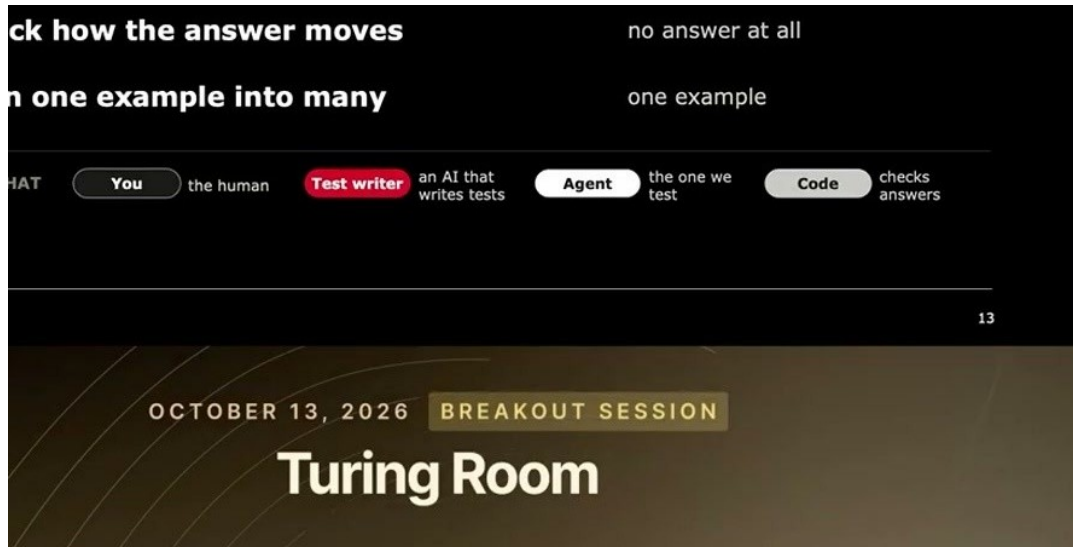
ב-Air Canada יש הרבה ספקים והרבה חוזים. מומחים עוברים על כל clause, מסמנים מה לא compliant ומה צריך לנסח מחדש. הטעות עולה ביוקר בשני הכיוונים: עיכוב בחוזה מעכב את כל השרשרת, ו-clause שהוחמץ יכול לחזור אל החברה. המומחים יקרים (הוא משווה לתעריף שעתי של עורך דין), ואין benchmark ברשת.

לפי Weill ה-agent, בגרסה ממושטת, עובד בארבעה שלבים: spot, כלומר סיווג כל סעיף בחוזה לתחום זיהוי פגם או שינוי נדרש; retrieval של המסמכים שתומכים ב-amendment; כתיבת ה-amendment; וסיווג סופי שמחליט אם להשאיר אותו. לשלב ה-retrieval כבר היה פתרון מעולם ה-RAG. לשאר השלבים לא היה.



"So we tried the RAG fix again. It kept us on the flat line": בדיקת ה-retrieval לא מספיקה כשהתפקיד של ה-agent הוא לזהות בעיות ולא למצוא עמודים

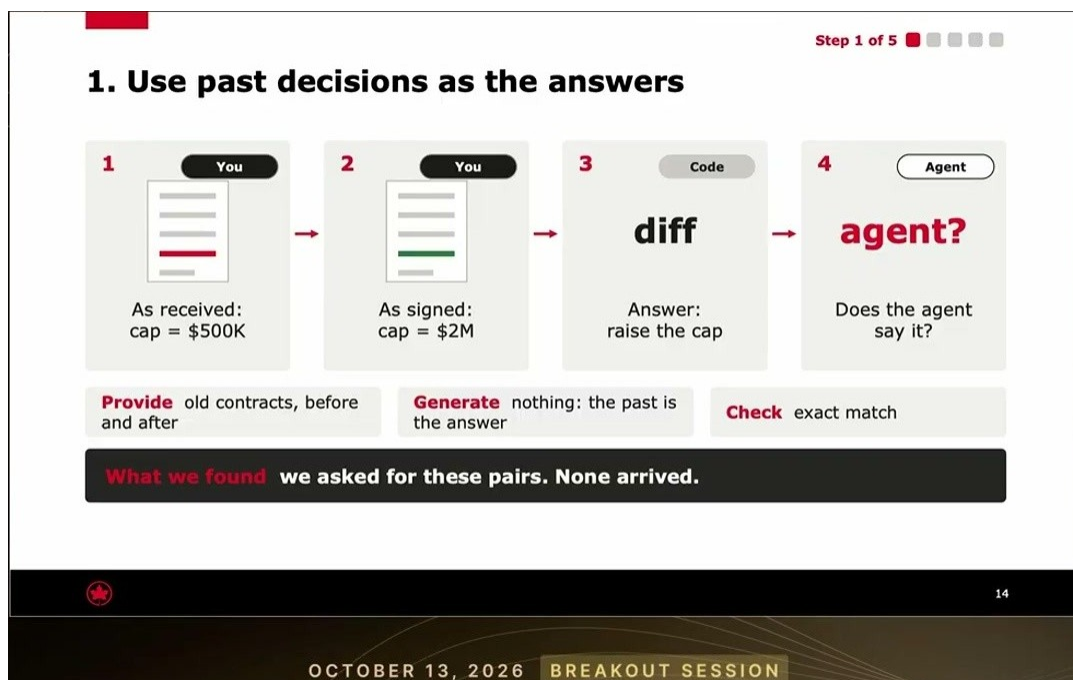
5. חמש דרכים לבנות test set



חמש הטכניקות בלי מומחה: החלטות עבר כתשובות, כתיבת התשובה ואז החוזה, שבירת חוזה טוב, בדיקה אם התשובה זזה, והפיכת דוגמה אחת להרבה. לכל אחת מצוין מאיפה מגיעה התשובה

טכניקה 1: החלטות עבר כתשובות. לא עבדה

הרעיון: לקחת את החוזה כפי שהתקבל ואת החוזה כפי שנחתם, לעשות diff כמו בקוד, ולצפות שה-agent יסמן בדיוק את מה שהשתנה. בפועל הצוות העסקי לא שומר גרסאות. נשמרת רק הגרסה האחרונה, ובאמצע יש משא ומתן. לא היו זוגות כאלה, והטכניקה נפלה.



טכניקה 1: חוזה כפי שהתקבל ($cap = \$500K$) מול כפי שנחתם ($cap = \$2M$). ה-diff נותן את התשובה, "raise the cap", והשאלה היא אם ה-agent אומר אותו דבר. בתחתית: "we asked for these pairs. None arrived"

טכניקה 2: לשתול הפרות כללים

מתחילים מכלל ולא מה-flag. הדוגמה שהוא נותן היא cap מעל מיליון דולר. כותבים clause שמפר את הכלל, מייצרים חוזה סביבו ובודקים אם ה-agent מוצא את ההפרה. זה זול וקל לייצר הרבה בדיקות כך. המגבלה היא שצריך להכיר את הכללים: חלקם הגיעו מהמומחים או מהתיעוד, ואת השאר הצוות ניסח בעצמו, בלי הבטחה לכיסוי מלא.

טכניקה 3: להסיר clauses מחוזה חתום

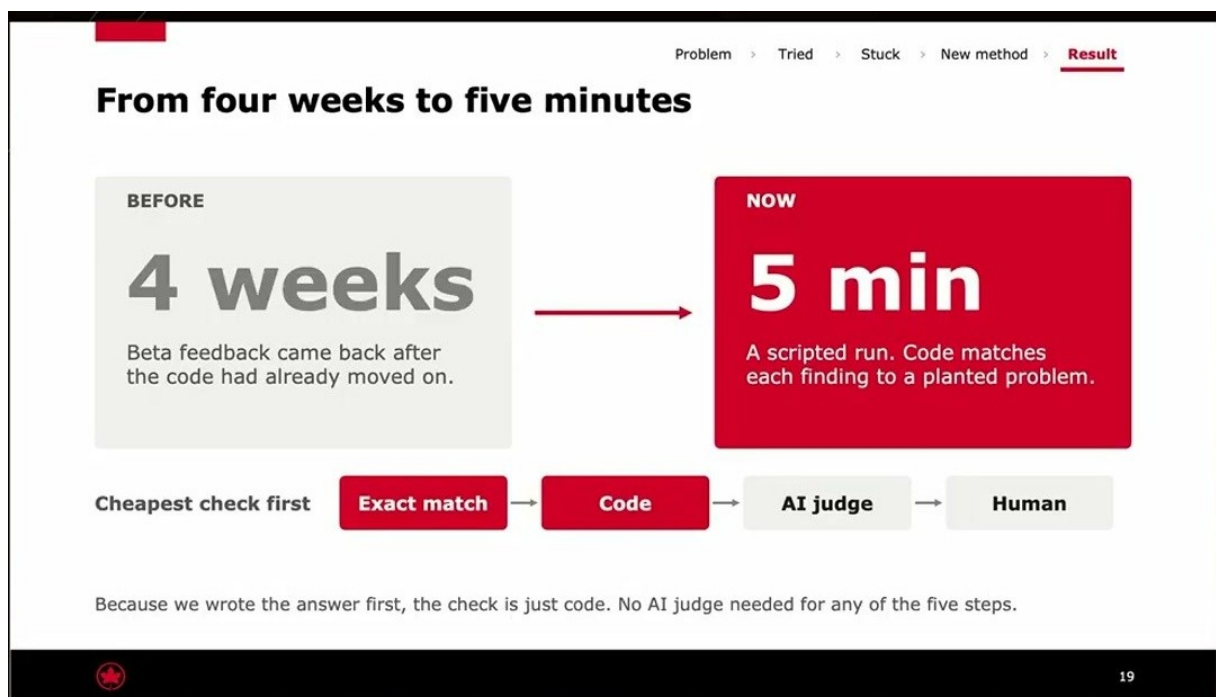
לוקחים חוזה שנחתם ומסירים ממנו clause (בתמלול "move", כנראה remove), וה-agent אמור לזהות שמשהו חסר. זו משימה קשה, כי לספקים שונים יש templates ותוכן שונים, ולדעת מה צריך להופיע בכל חוזה דורש שיקול דעת. מערבבים את החוזים האלה עם חוזים קיימים שלא אמורים להחזיר שום flag. התוצאה, לפי Weill: ה-agent מפספס את רוב ה-clauses שהוסרו, ולכן הוא מגדיר את הטכניקה הזאת כ-"very, very strong benchmark".

טכניקה 4: שינויים שלא אמורים לשנות את התוצאה

משנים את שם הספק או את הכתובת, מערבבים את סדר ה-clauses, או מריצים את אותו חוזה כמה פעמים. מספר ה-flags לא אמור להשתנות. זו בדיקת עקביות שאפשר לעשות כולה בקוד, בלי LLM as a judge.

טכניקה 5: templates עם שדות ריקים

למומחים יש templates של חוזים עם שדות שהם ממלאים. במקום למלא כל שדה פעם אחת, מייצרים לכל אחד מערך ערכים אפשריים, ומהצירופים נוצרים חוזים רבים. חלק מהצירופים מפרים כללים וחלק לא, והתשובה ידועה מראש, כך שגם כאן הבדיקה היא בקוד. היתרון הוא שה-templates כבר קיימים ובשימוש. לפי Weill, המעבר הזה קיצר את המשוב מארבעה שבועות לסקריפט שרץ בחמש דקות.

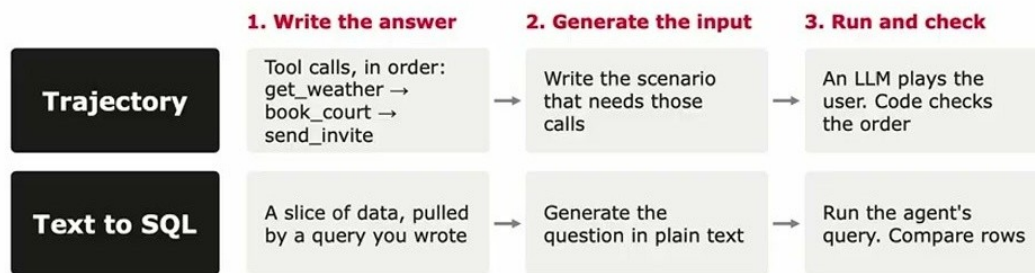


"From four weeks to five minutes": קודם, משוב beta הגיע אחרי שהקוד כבר השתנה. עכשיו ריצה בסקריפט, שבה קוד מתאים כל ממצא לבעיה שנשתלה. בתחתית סולם הבדיקות מהזולה ליקרה: Exact match → Code → AI judge → Human

6. בדיקה בקוד או LLM judge, ולולאת המשוב

הסולם בשקף הוא העיקרון: exact match ובדיקה בקוד מהירים וזולים. ברגע שעוברים ל-LLM as a judge, מתחיל רעש. לדבריו, למצוא את ה-prompt וה-rubric הנכונים ל-judge הוא כמעט קשה כמו לפתור את הבעיה עצמה. יש patterns כמו fact checking, אבל הם דורשים הרבה עבודה, ובפריקט הזה הצוות לא מצא אחד שעובד.

The same trick works far beyond contracts



Same pattern every time: decide the right answer first, then build the test around it.



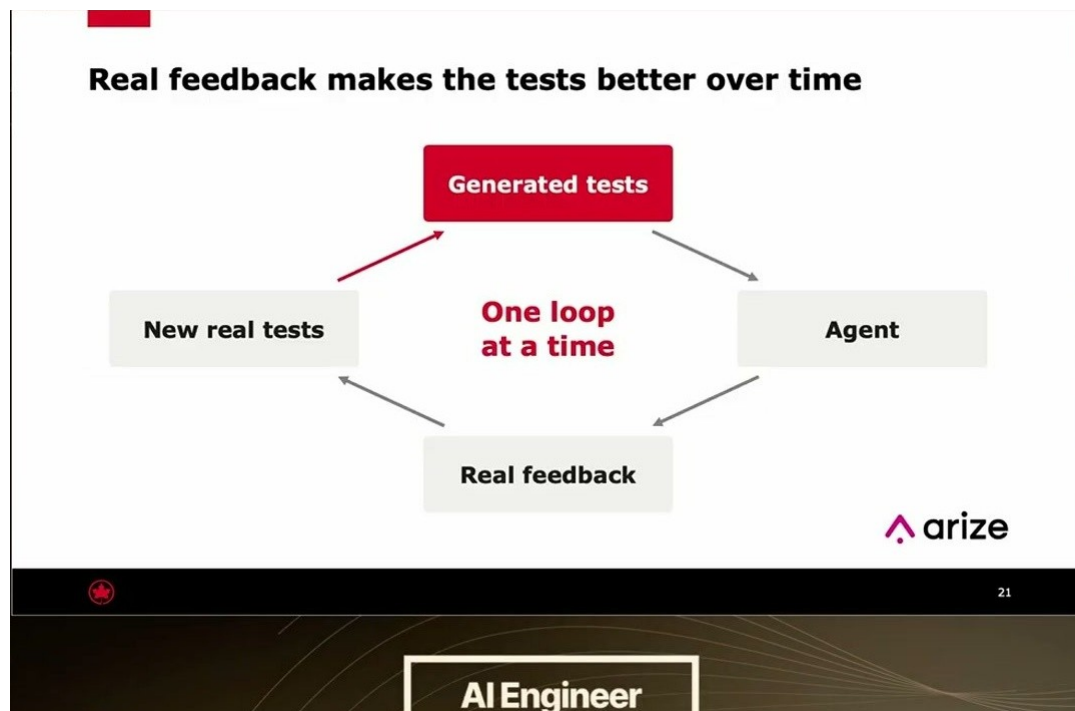
20

OCTOBER 13, 2026 BREAKOUT SESSION

Turing Room

אותו טריק מחוץ לחוזים: ב-Trajectory כותבים את רצף ה-tool calls הצפוי, מייצרים תרחיש שדורש אותו ובודקים את ה-trace. ב-Text to SQL מתחילים מנתונים ומ-query, מייצרים שאלה בטקסט ומשווים שורות

הוא מראה שהשיטה עובדת גם במקומות אחרים. ב-trajectory, כשה-agent צריך לקרוא ל-endpoints בסדר מסוים, כותבים את הסדר, מייצרים סיפור שמחייב אותו, נותנים ל-LLM לשחק את המשתמש ובודקים את ה-trace. ב-text to SQL התהליך מוכר. יש לזה מחיר: לפעמים הצוות שינה את ארכיטקטורת הפתרון כדי שאפשר יהיה לבדוק אותו, "the original design we could not evaluate it, we switched to something that we can actually evaluate" "because of that".



Generated tests → Agent → Real feedback → New real tests, לולאה של "Real feedback makes the tests better over time" "one loop at a time"

הנקודה האחרונה היא מה עושים עם משוב המומחים כשהוא מגיע. הוא לא מתקן ישירות את ה-agent. הוא מזין את ה-test set: בדיקות טובות יותר, prompts טובים יותר, ורעיונות חדשים לאילו clauses להסיר. וגם ה-test set עצמו הוא כלי למפתח, לא החלטת go/no-go.

Nathanael Weill we don't use this feedback to fix the agent. We use this feedback to generate better tests

Nathanael Weill It's a tool to help and support the developer. It's not the go-no-go for the solution

7. מה לעשות עכשיו

- **למפות מאיפה אפשר להשיג תשובה ידועה:** לכל שלב ב-agent, לשאול אם אפשר לייצר את הקלט מתוך תשובה ידועה. אם לא, לשקול לשנות את התכנון כדי שאפשר יהיה לבדוק אותו.
- **להתחיל מהבדיקה הזולה:** exact match, אחריו קוד, ורק אז LLM judge. לבדוק judge מול ה-agent לפני שסומכים עליו.
- **לשתול הפרות של כללים ידועים:** להוציא כללים מהמומחים ולהתיעוד ולייצר מכל אחד כמה מקרים. לתעד בנפרד את הכללים שהצוות ניסח בעצמו.
- **להוסיף בדיקות של מה שאסור שישתנה:** שמות, סדר, ריצות חוזרות. זה זול וחושף חוסר עקביות מיד.
- **לבקש templates ולא תשובות:** templates שהמומחים כבר משתמשים בהם מאפשרים לייצר הרבה בדיקות בלי לצרוך עוד מזמנם.
- **להעביר משוב מומחים ל-test set:** כל הערה של מומחה הופכת לבדיקה חדשה, לא לתיקון נקודתי ב-prompt.

שאלה פתוחה	למה זה חשוב
איך מודדים coverage של כללים שהצוות המציא?	275 בדיקות לא אומרות כמה מהמציאות הן מכסות
האם חוזים סינטיים מייצגים חוזים אמיתיים?	agent שמצליח על חוזים מיוצרים עלול להיכשל על ניסוחים של ספקים אמיתיים
מה קרה אחרי שהתגלה שה-agent מפספס clauses שהוסרו?	לא הוצג שיפור במספרים
מתי בכל זאת להחליט go/no-go?	ה-test set הוא לא הקריטריון, ולא נאמר מה כן
האם ויתור על LLM judge מתאים גם לשלב כתיבת amendment?	את איכות הטקסט שנכתב קשה לבדוק בקוד

8. מילון מונחים

מונח	פירוש
Benchmark	מערך בדיקות ציבורי וסטנדרטי להשוואת ביצועים
Test set	אוסף מקרי בדיקה עם תשובה ידועה
RAG	שליפת מסמכים רלוונטיים ואז ניסוח תשובה על בסיסם
Retrieval	שלב השליפה של המסמך או העמוד הנכון
Chunking	חלוקת מסמכים לקטעים לצורך אינדוקס
BM25 / alpha	דירוג חיפוש לפי מילות מפתח, ופרמטר האיזון בינו לבין חיפוש וקטורי
Embedding model	מודל שממיר טקסט לווקטור לחיפוש סמנטי
Clause	סעיף בחוזה
Amendment	שינוי מוצע לנוסח של סעיף
Flag	סימון של סעיף בעייתי על ידי ה-agent
LLM as a judge	שימוש במודל שפה כדי לדרג פלט של מודל אחר
Exact match	השוואה מדויקת של הפלט לתשובה הצפויה
Trajectory	רצף הפעולות או הקריאות ל-tools שה-agent מבצע
Trace	תיעוד הריצה שממנו בודקים את ה-trajectory
Template	חוזה תבנית עם שדות למילוי
Go/no-go	ההחלטה אם להעביר את המוצר לייצור