

Agentic AI: מבוא, design patterns וטיפים מעשיים לבניית agentic workflows

Andrew Ng על מה שמבדיל צוות שבונה agents טוב מצוות שבונה agents בינוניים:
פירוק משימות, evals ו-error analysis

Andrew Ng · DeepLearning.AI (קורס Agentic AI, לפי המיתוג בשקפים) · פורסם מחדש ב-X על ידי Komal
Prajapati · 2026-10-06

משך: כ-106 דקות · X

סיכום אוטומטי — (aibriefing.dev) AI Briefing

על המסמך הזה

הסרטון הוא קומפילציה של כ-106 דקות מתוך קורס Agentic AI של DeepLearning.AI. בשקפים מופיעים המיתוג של DeepLearning.AI והשם Andrew Ng, והדובר עצמו מדבר כל הזמן על "the next video" ועל "module". הוא פורסם מחדש ב-X עם כיתוב שיווקי של מי שהעלה אותו. את הכיתוב הזה לא לקחנו כמקור: כל הטענות כאן מבוססות על התמלול ועל השקפים בלבד. בתמלול יש קפיצות זמן (למשל מ-1:31 ל-4:32), ולכן חלקים מהקורס, ובעיקר המודול על tool use והחלק על "code as plan", חסרים בו.

הטיעון המרכזי של Ng: agentic workflow הוא תהליך שבו אפליקציה מבוססת LLM מבצעת כמה צעדים כדי להשלים משימה, ולא מייצרת תשובה אחת בבת אחת. הוא מסביר למה זה משפר ביצועים יותר ממעבר לדור מודל חדש, ומציג ארבעה design patterns: reflection, tool use, planning ו-multi-agent. לטענתו, הדבר שמנבא יותר מכל הצלחה של צוות הוא תהליך פיתוח ממושמע שבנוי על evals ועל error analysis, ולא הארכיטקטורה.

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות, כל אחת עם הראיה והחותמת שלה, ומה לא נאמר.
- 2. **מה זה agentic ומידת האוטונומיה:** מאיפה בא המונח, ולמה הוא מעדיף תואר ולא הגדרה בינארית.
- 3. **למה זה עובד ואיך מפרקים משימה:** שלושת היתרונות, ומבחן ה-"LLM או tool?" לכל צעד.
- 4. **ארבעת ה-design patterns:** reflection, tool use, planning ו-multi-agent, ומה הרמה הנוכחית של השליטה בכל אחד.
- 5. **Evals ו-error analysis:** הלב של הקורס: מערך evals של 20 דוגמאות, טבלת 2x2, traces וגיליון ספירת שגיאות.
- 6. **מודלים, עלויות, latency ותהליך העבודה:** איך משפרים רכיבים, איך מפתחים אינטואיציה למודלים, ומתי כדאי לדאוג לעלויות.
- 7. **Planning ו-multi-agent:** תוכניות ב-JSON, ומנהל agent שמתאם צוות.
- 8 ו-9. **מה לעשות עכשיו ומילון מונחים.**

הערות מקור המסמך הופק אוטומטית מהתמלול ומהשקפים של ההקלטה. שמות ומונחים עלולים להיות משובשים בגלל זיהוי הדיבור (ASR), למשל "LOM" במקום "Tavilli", "LLM" במקום "Tavily" ו-"U.com" במקום "You.com". המקומות שבהם השם לא ודאי מסומנים בטקסט. הציטוטים מועתקים מילה במילה ונבדקו מול החותמת שלהם. שום דבר לא נוסף ממקורות חיצוניים.

1. תקציר מנהלים

זה תוכן לימודי, לא הכרזה. אין בו מוצר, אין בו מספרי עסקים ואין בו ארכיטקטורה של מערכת production ספציפית. מה שכן יש בו: מסגרת חשיבה מסודרת שמגובה בדוגמאות קטנות וקונקרטיות, כמו research agent, עיבוד חשבונות, מענה למייל של לקוח, הסרת PII ו-agent לחנות משקפי שמש. המסר שחוזר לאורך כל הקורס הוא שבנייה בלי ניתוח היא בזבז זמן.

- **Agentic זה תואר ולא סטטוס:** Ng טבע את המונח כדי לעקוף את הוויכוח "זה agent או לא", ומודה ש-"a bunch of marketers would get hold of this term and use it as a sticker" [0:00]. מערכות נעות על רצף בין צעדים קבועים מראש לבין LLM שמחליט בעצמו על סדר הצעדים [10:41].
- **workflow על מודל ישן מנצח החלפת דור:** לדבריו, השיפור שמתקבל ממעבר דור מודל קטן מזה שמתקבל מהוספת agentic workflow לדור הקודם [13:46]. מספרי ה-HumanEval עצמם לא הוקראו.
- **פירוק משימות הוא המיומנות הבסיסית:** לגבי כל צעד בודקים אם LLM או tool יכולים לבצע אותו. אם לא, מפרקים אותו עוד, כמו בדוגמת החיבור שעבר מצעד אחד לשלושה ואז לחמישה [19:55, 23:10].
- **ארבעה design patterns:** reflection, tool use, planning ו-reflection [30:58]. multi-agent collaboration [37:10, 35:35]. "isn't magic" אבל נותן שיפור ניכר [32:32]. planning ו-multi-agent קשים יותר לשליטה [37:10, 35:35].
- **evals קטנים ומהירים עדיפים על שיתוק:** 10 עד 20 דוגמאות מספיקות כדי להתחיל, ומשפרים את מערך ה-evals יחד עם המערכת [41:48, 49:31]. הוא מסווג evals לפי שני צירים: קוד מול LLM-as-judge, ו-ground truth לכל דוגמה מול בלי [49:31].
- **error analysis במקום תחושת בטן:** קוראים traces של הדוגמאות הכושלות וסופרים בגיליון איזה רכיב אשם. אחרת "many months of work with very little progress" [54:13, 58:52].
- **מודלים גדולים מציינים יותר להוראות:** במשימת הסרת PII, Llama 3.1 8B פספס שם ושבר את הפורמט, ומודל חזק יותר (GPT-5 לפי השקף) הצליח [1:08:12].
- **איכות קודם, עלות ו-latency אחר כך:** מודדים זמן ועלות לכל צעד ורק אז בוחרים מה לייצל. בדוגמה: 7, 5, 3, 11 ו-18 שניות [1:12:49].
- **צוותים חסרי ניסיון בונים יותר מדי ומנתחים מעט מדי:** תהליך העבודה הנכון עובר הלוך ושוב בין בנייה לניתוח, ולא מתקדם בקו ישר [1:17:33].

מה לא נאמר

- **אין מספרים לגרף ה-HumanEval:** השקף מציג את GPT-3.5 ואת GPT-4 בצירוף ארבעה patterns, אבל באודיו לא הוקראו אחוזים, ולכן הטענה "workflow מנצח דור מודל" נשענת על גרף שלא קראנו את ערכיו.
- **ספירת השגיאות נקטעת:** מגיליון ה-error analysis נשמר בתמלול רק "5% מהזמן" עבור search terms. שאר האחוזים חסרים בגלל קפיצה בתמלול.
- **לא הומלץ על framework:** הוא אומר שיש "quite a lot of software frameworks" ל-multi-agent, אבל לא נוקב בשם אף אחד מהם. הכלי היחיד שהוא מזכיר בשמו הוא AI Suite, שבו הוא עצמו משתמש.
- **חלקים חסרים:** המודול על tool use והחלק על "code as plan" לא נכללו בקומפילציה, וגם הפתרון בפועל לבעיית ה-latency נקטע.
- **אין דיון על בטיחות, הרשאות או guardrails ל-agents אוטונומיים.** הוא אומר רק שהם "harder to control".
- **הטענות בכיתוב של הפוסט ב-X, כמו "godfather of AI" ו-"free course", לא מופיעות בתמלול ולא נבדקו.**

2. מה זה agentic ומידת האוטונומיה

Degrees of autonomy

Agentic AI can be less or more autonomous



 DeepLearning.AI

Andrew Ng

השקף Degrees of autonomy: ציר מ-Less autonomous ל-More autonomous. בקצה הפחות אוטונומי כל הצעדים קבועים מראש, כל השימוש ב-tools מקודד מראש, והאוטונומיה היא רק בטקסט שה-LLM מייצר.

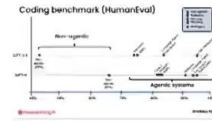
Ng פותח בהשוואה: לבקש מ-LLM לכתוב חיבור זה כמו לבקש מאדם להקליד "by writing from the first word to the last word on one go and without ever using backspace" [1:31]. ב-agentic workflow מתכננים outline, מחפשים ברשת, כותבים טיוטה, קוראים אותה ומתקנים. ההדגמה שמלווה את הקורס היא research agent שמקבל את השאלה איך להקים חברת רקטות שתתחרה ב-SpaceX. הוא מתכנן, מחפש ברשת, מדרג ממצאים, מעביר את הטיטות ל-editor agent ומפיק דוח Ng [4:32]. markdown מספר שבנה research agents כאלה לציות משפטי, לבריאות ולמחקר מוצר [6:05].

בשאלה "מה זה agent" הוא מציע להפסיק להתווכח: לדבריו, הוא התחיל להשתמש במונח "if we use it as an adjective rather than a binary" [6:05]. בקצה הפחות אוטונומי המהנדס קובע את רצף הצעדים ואת הקריאות ל-tools. בקצה האוטונומי ה-LLM מחליט מה לחפש, כמה עמודים להביא, אם להמיר PDF לטקסט ואם לחזור אחורה, ויש agents שאפילו כותבים tools חדשים. השורה התחתונה שלו: רוב הערך העסקי היום נמצא בקצה הפחות אוטונומי, והקצה האוטונומי "less easily controllable a little bit more unpredictable" ועדיין נמצא במחקר [10:41].

3. למה זה עובד ואיך מפרקים משימה

Key benefits of agentic workflows

- Much better performance
- Faster than humans because of parallelization
- Modular: can add or update tools, swap out models



DeepLearning.AI

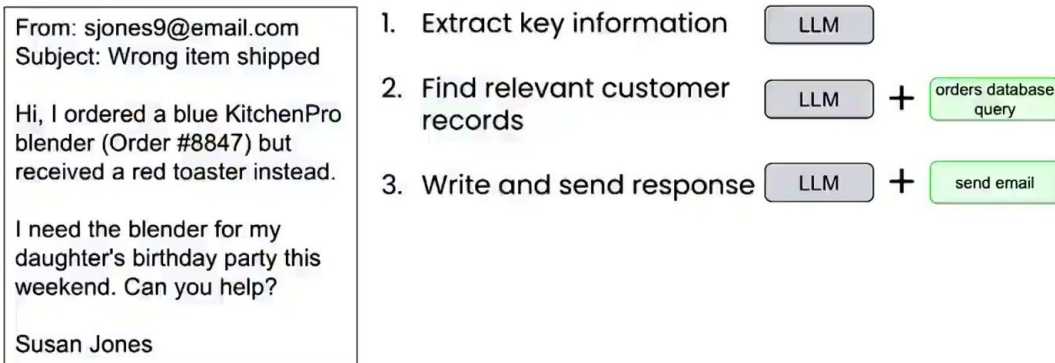
Andrew Ng

השקף Key benefits of agentic workflows: שלושה יתרונות. ביצועים טובים בהרבה (עם תמונה מוקטנת של גרף HumanEval), מהירות גבוהה מזו של אדם בזכות parallelization, ומודולריות: אפשר להוסיף או לעדכן tools ולהחליף מודלים.

היתרון הראשון הוא ביצועים. על גרף ה-HumanEval (שקף GPT-3.5 ו-GPT-4, מול Non-agentic, Multiagent-Reflection, Tool Use, Planning) הוא אומר ש-"the improvement from one generation of model to another, which is huge, is still not as big a difference as implementing an agentic workflow on their previous generation of model" [13:46]. היתרון השני הוא מקביליות. ה-workflow איטי יותר מבקשה אחת למודל, אבל מהיר יותר מאדם, כי אפשר להוריד תשעה עמודים במקביל במקום לקרוא אותם אחד אחרי השני. השלישי הוא מודולריות: אפשר להחליף מנוע חיפוש (הוא מונה את Google דרך Serper, את Bing, DuckDuckGo, את You.com ו-Tavily, כשהשמות האחרונים משובשים ב-ASR), להוסיף מנוע חדשות, או להשתמש ב-LLM שונה לכל צעד [15:17].

פירוק משימה מתחיל בגרסה הנאיבית ומעמיק לפי מה שהתוצאות מראות. דוגמה: חיבור שיצא "disjointed" גרם לו לפרק את צעד הכתיבה לטייטה, ביקורת ותיקון [18:22]. דוגמה נוספת: מענה למייל של לקוח, שמתפרק לחילוף פרטים, query למסד הנתונים ושליחת מייל דרך API [19:55]. לגבי כל צעד הוא שואל: "can this step be implemented with either an LLM or with one of the tools, such as an API or a function call that I have access to?" ואם התשובה שלילית, הוא שואל איך אדם היה מבצע את הצעד ומפרק עוד [23:10].

Look for low-quality outputs



DeepLearning.AI

Andrew Ng

מייל מלקוחה שקיבלה מצגם אדום במקום בלנדר (הזמנה #8847), לצד הפירוק לשלושה צעדים: חילוץ פרטים עם LLM, איתור רשומות עם LLM ו-`orders database query`, וכתיבה ושליחה עם LLM ו-`send email`.

4. ארבעת ה-design patterns

Agentic Design Patterns

1. Reflection
2. Tool use
3. Planning
4. Multi-agent collaboration

DeepLearning.AI

Andrew Ng

השקף *Agentic Design Patterns*: רשימה ממוספרת של *Reflection*, *Tool use*, *Planning* ו-*Multi-agent collaboration*.

Reflection: מבקשים מהמודל לבדוק קוד שהוא עצמו כתב, "for correct and style and efficiency", מחזירים לו את הביקורת ומקבלים גרסה 2. אם גם מריצים את הקוד ומעבירים את הודעות השגיאה, מקבלים גרסה 3. הוא מסייג: "this design pattern isn't magic. It does not result in everything working 100% of the time" [32:32]. אפשר גם להפריד לשני agents, כותב ו-critic. בעיניו, agent הוא בסך הכול LLM עם prompt של persona.

Planning: ה-LLM ולא המפתח קובע את רצף הצעדים. "Agents that plan today are harder to control and somewhat more experimental, but sometimes it can give really delightful results" [35:35]. **Multi-agent**: הוא מביא את ChatDev של Chen Qian ושותפיו (השם כפי שנשמע ב-ASR), שבו agents בתפקידי CEO, programmer, tester ו-designer מתפקדים כחברת תוכנה וירטואלית, ואת הדוגמה של researcher, marketer, editor ו-shopper. ההסתייגות שלו: "Multi-agent workflows are more difficult to control since you don't always know ahead of time what the agents will do". מצד שני, לדבריו מחקרים מראים שיפור במשימות כמו כתיבת ביוגרפיות ובחירת מהלכי שחמט [37:10]. המספרים עצמם מופיעים רק בשקף.

Multi-agentic workflows



Multiagent Debate

Task	Single agent	Multi-agent
Biographies	66.0%	73.8%
MMLU	63.9%	71.1%
Chess move	29.3%	45.2%

(Du et al., 2023)

השקף Multi-agentic workflows: משמאל איור של ChatDev, ומימין טבלת Multiagent Debate של single agent מול multi-agent. Biographies: 66.0% מול 73.8%. MMLU: 63.9% מול 71.1%. Chess move: 29.3% מול 45.2%. המספרים נקראו מהשקף ולא נאמרו באודיו.

5. Evals ו-error analysis: הלב של הקורס

Evaluate with code (objective) LLM-as-judge (subjective)

השקף Two axes of evaluation: 2x2 טבלת. ציר אחד: Evaluate with code (objective) מול LLM-as-judge (subjective). ציר שני: per example ground truth מול no per example ground truth.

השיטה שלו: בונים מהר מערכת "quick and dirty end-to-end", קוראים את הפלטים, ונותנים לשגיאות שחוזרות על עצמן לקבוע מה מודדים. בדוגמת החשבוניות התגלה שהמערכת מבלבלת בין תאריכים, ולכן הוא אסף "10 to 20 invoices and manually write down what is the due date", ביקש מהמודל פורמט YYYY-MM-DD, חילץ אותו ב-regex והשווה ל-[41:48] ground truth. בדוגמת copy שיווקי היעד אחיד (10 מילים לכל דוגמה), ולכן לא צריך ground truth לכל דוגמה [44:52]. ב-research agent יש 3 עד 5 נקודות gold standard לכל נושא, ו-LLM-as-judge סופר כמה מהן מופיעות ומחזיר JSON עם ציון והסבר [46:23]. הארבעה משתלבים בטבלת 2x2, והטבלה מתייחסת ל-end-to-end evals.

הטיפים: "First, quick and dirty evals is fine to get started". הוא רואה צוותים "almost paralyzed because they think building evals is this massive multi-week effort" [49:31]. כשה-eval לא מראה שיפור שהעין שלכם רואה, זה סימן להגדיל את המערך או לשנות את המדד. ככל שה-evals מבשילים, "you can then shift more".

[51:07] "and more of your trust to the metric-based evals" research agent-כשה: error analysis ומשם ל- and more of your trust to the metric-based evals. מפספס נקודות מפתח, כל אחד מהרכיבים יכול להיות האשם, כלומר search terms, מנוע החיפוש, בחירת המקורות, fetch-ה או הכתיבה. ניחוש לפי תחושת בטן "sometimes that leads to many months of work with very little" [54:13] "progress in the overall performance of the system". הפתרון: לקרוא את ה-trace, כלומר "the overall set of outputs of all of the intermediate steps" (פלט של צעד בודד נקרא span), ולהתמקד רק במקרים הכושלים [57:19].

Counting up the errors

Prompt	Search terms	Search results	Picking 5 best sources	...	...
Recent developments in black hole science		Too many blog posts, not enough papers			
Renting vs buying a home in Seattle			Missed well-known blog		
Robotics for harvesting fruit	Terms too generic	Website for elementary school students			
...	...	...	...		
Batteries for electric vehicles		Only selected US-based companies	Missed magazine		

...

השקף Counting up the errors: גיליון שבו כל שורה היא prompt (חורים שחורים, שכירות מול קנייה בסיאטל, רובוטיקה לקטיף פירות, סוללות לרכב חשמלי) וכל עמודה היא רכיב (Search terms, Search results, Picking 5 best sources). בתאים: "Too many blog posts, not enough papers", "Terms too generic", "Only selected US-based companies".

שגיאה מוגדרת כמצב שבו צעד מפיק "something that performs significantly worse than maybe what a human expert would have given a similar input" [58:52]. ההגדרה הזאת חשובה, כי היא קובעת שלא מאשימים רכיב שקיבל קלט גרוע. אם תוצאות החיפוש היו בלוגים פופולריים, צעד בחירת חמשת המקורות עשה את הטוב ביותר שאפשר היה. סופרים כמה פעמים כל רכיב אשם (בתמלול נשמר רק ש-search terms אשמים ב-5% מהמקרים), ומשקיעים במקום שבו הספירה הכי גבוהה [1:02:01].

6. שיפור רכיבים, מודלים, עלות ותהליך העבודה

Instruction following results (GPT-5)

Identified PII (type → value):

1. Full Name → Jessica T. Alvarez
2. Social Security Number → 524-18-7629
3. Physical Address → 1024 Maple Ridge Lane, Boulder, CO 80301

Followed formatting instruction

Identified all PII

REDACTED:

On July 14, 2023, ***** (SSN: *****), residing at *****,
submitted a support ticket...

DeepLearning.AI

Andrew Ng

השקף (GPT-5) Instruction following results: המודל מזהה את כל ה-PII (שם מלא, מספר ביטוח לאומי וכתובת), מחזיר טקסט מצונזר, ומקבל את ההערות Followed formatting instruction ו-PII Identified all PII. בשקף הקודם, Llama 3.1 8B מסומן Didn't follow the instructions properly. Missed some PII instructions properly.

ברכיבים שאינם LLM מכווננים hyperparameters, למשל מספר תוצאות וטווח תאריכים בחיפוש, similarity threshold ו-chunk size ב-RAG, או detection threshold בזיהוי אנשים. אפשר גם להחליף ספק [1:03:34]. ברכיבי LLM מוסיפים הוראות מפורשות ו-few-shot examples, או מחליפים מודל. בדוגמת ה-Llama 3.1 8B PII, (נשמע "OpenWay Lambda 3.1" ב-ASR) הפר את הפורמט, פספס את השם וצנזר רק חלק מהכתובת. באודיו המודל השני מתואר רק כ-"a more intelligent model", והשקף מזהה אותו כ-GPT-5. המסקנה שלו: "the larger the frontier models tend to be much better at following instructions". כדי לפתח אינטואיציה הוא ממליץ לשחק עם כל מודל חדש, להחזיק סט evals אישי ולקרוא prompts של אחרים. הוא משתמש ב-AI Suite כדי להחליף מודלים בקלות [1:11:18].

בעלות וב-latency הוא מטפל רק אחרי שהאיכות טובה. הוא מספר שמוצר הצליח כל כך שהעלות הפכה לבעיה, ומוסיף: "[1:12:49] But that's a good problem to have." השיטה: מודדים זמן לכל צעד (7 שניות ל-search terms, 5 לחיפוש, 3, 11 ו-18 לכתיבת החיבור) ושוקלים parallelism, למשל ב-fetch. כך מתגלה לפעמים שרכיב מסוים לא תורם משמעותית לעלות או ל-latency [1:16:00]. בסיכום הוא מתאר תהליך שעובר הלוך ושוב בין build ל-analyze: מערכת end-to-end, קריאת eval, traces של 10 עד 20 דוגמאות, error analysis, ובסוף component-level evals [1:17:33].

— **Andrew Ng [1:17:33]** And what I see less experienced teams often do is spend a lot of time building and probably much less time analyzing with error analysis, building evals, and so on

7. Planning ו-multi-agent: תוכניות ב-JSON ומנהל שמתאם צוות

Formatting plan as JSON

Updated system prompt

You have access to the following tools:

{description of tools}

Create a step-by-step plan in JSON format.

Each step should have the following items: step number, description, tool name, and args.

Do you have any round sunglasses in stock that are under \$100?

LLM

```
{
  "plan": [
    {
      "step": 1,
      "description": "Find round sunglasses",
      "tool": "get_item_descriptions",
      "args": {"query": "round sunglasses"}
    },
    {
      "step": 2,
      "description": "Check available stock",
      "tool": "check_inventory",
      "args": {"items": "results from step 1"}
    },
    ...
  ]
}
```

DeepLearning.AI

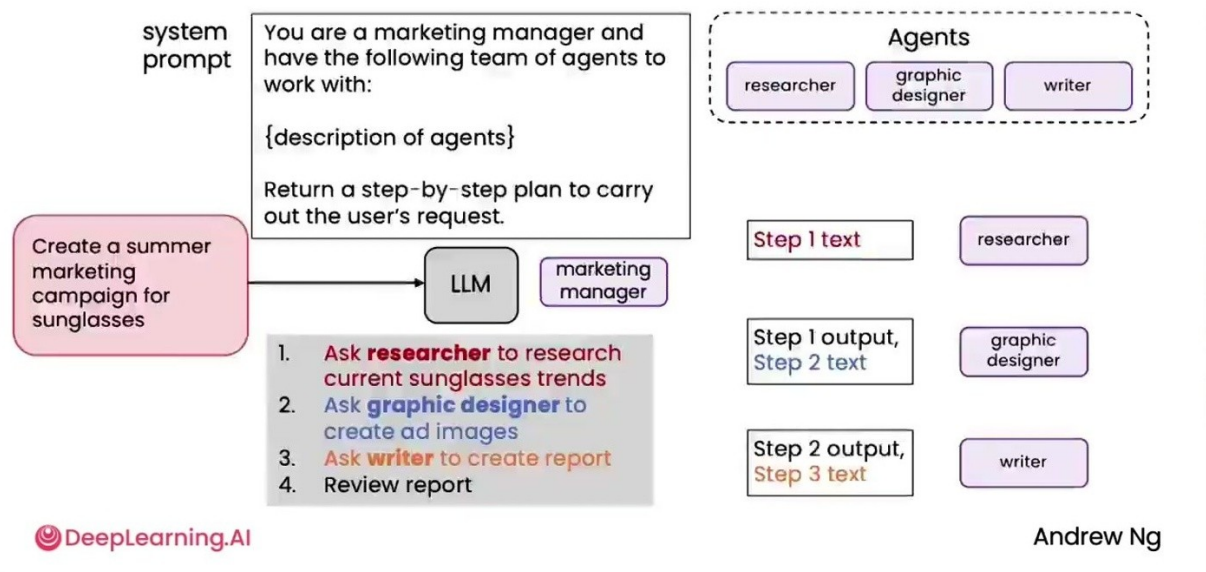
Andrew Ng

השקף Formatting plan as JSON: ה-system prompt המעודכן מבקש תוכנית ב-JSON שבה לכל צעד יש מספר, תיאור, שם tool וארגומנטים. מימין דוגמת plan: צעד 1 הוא get_item_descriptions עם query "round sunglasses", וצעד 2 הוא check_inventory.

דוגמת planning: חנות משקפי שמש עם tools כמו get_item_descriptions, check_inventory, process_item_return ו-get_item_price. לשאלה "round sunglasses in stock under \$100" ה-LLM כותב תוכנית ומבצע אותה צעד אחרי צעד, ומקבל לכל צעד את הפלט של הצעד הקודם. היתרון: "we did not have to decide in advance" באיזה סדר לקרוא ל-tools, ובקשה אחרת, כמו החזרת משקפיים עם מסגרת זהב, תייצר תוכנית אחרת [1:23:40]. לדבריו, planning כבר עובד היטב ב-agentic coding, ובשאר התחומים הוא עדיין ניסיוני: "one of the challenges of planning is it makes the system sometimes a little bit harder to control" [1:25:12]. לגבי פורמט, JSON או XML הם האפשרויות הטובות, חוץ מ-markdown עמום יותר, ו-"I think plain text is maybe the least reliable of these options" [1:28:21].

ב-multi-agent האנלוגיה שלו היא threads ו-processes על מחשב אחד: חלוקה לתפקידים מקלה על המפתח לחשוב [1:29:55]. researcher עם web search, graphic designer עם image generation או code execution, ו-writer בלי tools, וכל אחד מהם הוא LLM עם prompt של תפקיד [1:32:57]. הוא מציג שני דפוסים תקשורת נפוצים: linear, ומנהל agent שמתכנן ומאציל, כשהמנהל הוא בעצמו "fourth agent" [1:37:30]. שאר הדפוסים פחות צפויים, למשל all-to-all, שבו כל הודעה נכנסת ל-context של מי שמקבל אותה: "In practice, I find the results of all-to-all communication patterns a bit hard to predict." [1:42:07]. לדבריו הוא מתאים רק כשאפשר לסבול קצת כאוס ולהריץ שוב.

Example: Planning with multiple agents



השקף Planning with multiple agents: system prompt של marketing manager עם צוות (researcher, graphic designer, writer). התוכנית: לבקש מה-researcher מחקר טרנדים, מה-graphic designer תמונות ומה-writer דוח, ואז Review report. כל שלב מועבר ל-agent המתאים.

8. מה לעשות עכשיו

- **התחילו מהקצה הפחות אוטונומי:** workflow שבו המהנדס קובע את הצעדים מראש. שם נמצא רוב הערך, ושם הכי קל לשלוט במערכת [10:41].
- **פרקו כל משימה עד שכל צעד הוא LLM או tool:** ואם התוצאה לא טובה, פרקו שוב את הצעד החלש [23:10].
- **בנו מחר eval של 10 עד 20 דוגמאות:** בחרו את השגיאה הנפוצה ביותר שראיתם בפלטים, ושייכו אותה לרבע המתאים בטבלת ה-2x2 [41:48, 49:31].
- **תעדו traces מהיום הראשון:** בלי פלטי ביניים אי אפשר לעשות [57:19] error analysis.
- **החזיקו גיליון ספירת שגיאות לפי רכיב:** אל תאשימו רכיב שקיבל קלט גרוע, והשקיעו במקום שבו הספירה הכי גבוהה [58:52].
- **השתמשו ב-wrapper שמאפשר להחליף מודל בקלות:** והשוו מודלים לכל צעד בנפרד בעזרת ה-evals [1:11:18].
- **מדדו זמן ועלות לכל צעד לפני שאתם מייעלים:** ורק אחרי שהאיכות כבר טובה [1:16:00].
- **העדיפו plans ב-JSON או ב-XML על פני טקסט חופשי:** וכשאפשר, תקשורת linear או דרך מנהל ולא all-to-all [1:28:21, 1:42:07].

שאלה פתוחה	למה זה חשוב	מה נאמר בהרצאה
מה בדיוק הפער ב-HumanEval בין non-agentic ל-agentic?	זו הראיה המרכזית לטענה שה-workflow חשוב יותר מדור המודל	רק טענה איכותית. הגרף מוצג בלי שהמספרים מוקראים [13:46]
מתי כדאי לעבור מ-end-to-end evals ל-component-level evals?	בניית evals לכל רכיב עולה זמן	"when it's even more mature", בלי סף מוגדר [1:17:33]
איך שולטים ב-agent שמתכן בעצמו בזמן ריצה?	מפתח לא יודע מראש איזו תוכנית תיווצר	הוכר כאתגר, ופתרון לא הוצג [1:25:12]
באיזה framework כדאי להשתמש ל-multi-agent?	בחירת framework משפיעה על דפוסי התקשורת שאפשר לממש	"quite a lot of software", בלי שמות [1:42:07]
האם LLM-as-judge אמין מספיק כדי לסמוך עליו?	ה-evals הסובייקטיביים נשענים עליו	לא נדון כיול של ה-judge מול שיפוט אנושי
איך זה משתלב עם הרשאות ואבטחה כש-agent שולח מיילים או מבצע מכירות?	ה-tools בדוגמאות מבצעים פעולות אמיתיות	לא נדון

9. מילון מונחים

מונח	משמעות
Agentic workflow	תהליך שבו אפליקציה מבוססת LLM מבצעת כמה צעדים (תכנון, חיפוש, טיוטה, ביקורת) כדי להשלים משימה
Degrees of autonomy	רצף בין מערכת שבה המהנדס קובע את הצעדים לבין מערכת שבה ה-LLM מחליט עליהם
Task decomposition	פירוק משימה לצעדים שכל אחד מהם LLM או tool יכולים לבצע
Reflection	המודל, או agent מבקר, בודק את הפלט ומשפר אותו לפי הביקורת או לפי הודעות שגיאה
Tool use	מתן אפשרות ל-LLM לקרוא לפונקציות או ל-APIs, כמו חיפוש, הרצת קוד או query למסד נתונים
Planning	ה-LLM כותב בעצמו את רצף הצעדים ואת ה-tools, ואחר כך מבצע אותם
Multi-agent	כמה LLMs עם prompts של תפקידים שונים שמשתפים פעולה על משימה
Communication pattern	מבנה התקשורת בין agents: linear, מנהל וצוות, או all-to-all
Evals	מדדים אוטומטיים לאיכות המערכת על מערך דוגמאות קבוע
End-to-end / component-level eval	מדידה של הפלט הסופי מול מדידה של רכיב בודד
LLM-as-judge	שימוש ב-LLM אחר כדי לתת ציון לפלט סובייקטיבי
Ground truth	התשובה הנכונה שנכתבה ידנית לכל דוגמה, למשל תאריך הפירעון
Error analysis	בחינה של מקרים כושלים וספירה של הרכיב האשם בכל אחד מהם
Trace / Span	כל פלטי הביניים של הרצה אחת, ופלט של צעד בודד בתוכה
Few-shot prompting	הוספת דוגמאות של קלט ופלט רצוי ל-prompt
RAG	Retrieval augmented generation: שליפת קטעי טקסט רלוונטיים והזנתם למודל
Hyperparameter	פרמטר של רכיב שאפשר לכוון, כמו chunk size או סף דמיון
PII	מידע מזהה אישי שצריך להסיר מהנתונים
Latency	זמן התגובה של המערכת או של צעד בודד