System One Model

Jev

「文章を生成しない」判断特化型AIモデルの概要と、プロダクト開発への取り込み方

```
unstructured state  →  typed decision + probability
```

非構造データを入れ、型付きの判断と確率を返す

社内エンジニア・デザイナー向け勉強会資料 | 2026年9月18日時点・一次情報ベース

TL;DR

結論（3行）

1

LLMの代替ではなく「賢いif文」

Jevは文章を書かず、事前定義した選択肢・段階・真偽に対して確率付きの判断だけを返す。

2

速度・コストが桁違い（ベンダー公称）

応答70〜500ms、入力$0.042/MTok、出力無料。リアルタイムUXや大量データ処理に効く。

3

設計の主役はコード側

問いを原子的に分解し、確信度で自動実行／確認／人に回すを分岐させる設計が前提。

Overview

Jevとは何か

開発元	TypeSafe AI（米西海岸拠点）
創業者	Diogo Almeida （元OpenAI、ChatGPTの基盤研究に関与）
公開	2026年9月15日 early access開始 （ウェイトリスト制）
分類	新カテゴリ「System One Model」の第1弾
学習手法	RLCD：Reinforcement Learning for Calibrated Decisions
提供形態	REST API／Python・JavaScript SDK／Playground／エージェント用Skill
資金調達	DCVC等から4,000万ドルを調達と報道 （2026年9月17日）

名前の由来

System One
カーネマン『ファスト＆スロー』の直感的・高速な思考（システム1）から。

Jev
経済学者ジェヴォンズから。効率化で知能の需要がむしろ増える、という賭け。

https://typesafe.ai/

出典: TypeSafe AI Blog「Introducing System One Models & Jev」(2026-09-15)

LLM vs System One

既存LLMとの違い

観点	既存LLM	Jev（System One）
学習の最適化	RLHF／RLVR（人の好み・検証可能な報酬）	RLCD（較正された確率での判断）
出力	文字列。パース・検証が必要	事前定義スキーマの型付き値。型エラーは原理上ゼロ
生成方式	逐次（1トークンずつ）	並列（全出力を1クエリで算出）
応答時間	数秒〜数分	70〜500ms
確信度	自己申告は過信・不安定になりがち	全回答に確率と確信度を付与
得意領域	チャット、コーディング、人が監督する作業	分類・ルーティング・スコアリング・検証

出典: TypeSafe AI Blog「Frontiers, Old and New」比較表を要約

Numbers

公称スペック

70–500
ms
エンドツーエンド応答時間

$0.042
/MTok
入力トークン単価（出力は無料）

255
options
Choiceの選択肢数上限

~193x
faster
自社ワークフロー評価での最大値

読み方の注意　いずれもベンダー公称値。約193倍速・約445倍安は「実運用で得られる上限側」と本人たちが明記。計測は米西海岸から。価格は補助されていないことを証明できない、とも自認している。

出典: TypeSafe AI Blog「Evidence / Technical Results」「Wikiracing」

Primitives

3種類の「問い」＝AIプリミティブ

Choice

リストから1つ選ぶ

返り値

`choice / probabilities / confidence`

例：問い合わせの担当部署は billing / technical / sales のどれか

Score

段階評価で採点する

返り値

`score / probabilities / confidence`

例：顧客の苛立ち度を「冷静／不満だが丁寧／激怒」の3段階で

Noul

文が真である確率

返り値

`noul（0〜1）`

例：このメッセージは緊急性を示しているか

3種は1回のAPI呼び出しに混在可。各問いは同じstateに対し並列・独立に評価されるため、問いを増やしても応答時間はほぼ変わらない。

出典: docs.typesafe.ai「Introduction」

API

呼び出し例（Python SDK）

```
from typesafe_sdk import Choice, Noul, Score, TypeSafeClient
client = TypeSafeClient()  # TYPESAFE_API_KEY を参照

res = client.system_one(
  state=ticket_text,
  questions={
    "department": Choice(
      instructions="Which team should handle this",
      criteria={"billing": "...", "technical": "...",
                "sales": "..."}),
    "frustration": Score(
      instructions="How frustrated the customer appears",
      criteria=["Calm", "Frustrated but civil", "Very
angry"]),
    "is_urgent": Noul(
      instructions="The message conveys urgency"),
  })
```

出典: docs.typesafe.ai「Quick start」サンプルを簡略化

レスポンス（抜粋）

```
department: "technical"
  billing 0.159
  technical 0.84
  sales 0.001
  confidence 0.596

frustration: 1.035
  confidence 0.842

is_urgent: 0.999
```

POST api.typesafe.ai/v1/systemone

model: "jev-latest"／pip install typesafe-sdk（Python 3.10以上）

Architecture

設計原則：コードが主導し、AIは狭い判断だけ担う

01

決定的処理はコードで

期限超過日数などルールで書けるものはAIに渡さない

02

stateを絞る

その問いに必要な文脈だけをJSONで渡す

03

問いを原子分解

「スパムか？」でなく「認証情報を要求しているか？」等へ

04

まとめて並列に聞く

投機的な問いも1回で投げ、使うかはコードが決める

05

コードで合成・分岐

重み付き和や閾値。優先度変更は係数を変えるだけ

公式ドキュメントは「03 問いの原子分解」を最重要概念と位置付けている。Jevはエージェントではなく、ソフトウェアに埋め込む部品という思想。

出典: docs.typesafe.ai「How to build with TypeSafe」

Pattern

実装例：エミュレータ状態を構造化してJevに渡す

stateに入れる情報（画面画像は送らない）

`player` 位置・速度・接地・パワーアップ／`trajectory` 滞空時間・踏切からの距離／`hazard` 敵3体までの予測位置と接触時刻／`terrain` 障害物と穴の形状／`reaction_timing` 実測した観測から操作までの遅延／`recent_control` 直前の操作と結果／`episode` 残機・タイム・進捗

1回の呼び出しで3つの判断

`Choice`	次のコントローラ操作を7候補から選ぶ
`Noul`	いま前方ジャンプが有効かの確率
`Score`	その瞬間の危険度（可視化用）

設計上の勘所　タイミング計算はコード側で行い「この判断でジャンプを開始すべきか」という型付きの事実に変換してから渡す。知覚と算術はコード、判断だけがモデル——設計原則03の実例。

8フレームごとに1判断。各判断は行動確率・確信度・レイテンシとともにJSONLへ記録され、後から検証できる。

出典: GitHub: fhshaik/typesafe-mario（README、2026年9月時点）

Confidence

確信度で「実行／確認／人へ」を分岐する

高

自動で実行

人を介さず処理を進める

中

慎重に進める

ユーザー確認、レビュー旗、追加情報の取得

低

実行しない

人にエスカレーション、上位の推論LLMへフォールバック

```
if a.confidence < 0.5:
    route_to_human()
elif a.choice == "check_balance":
    show_balance()      # 低リスク
elif a.choice == "approve_transfer":
    if a.confidence > 0.9:
        confirm_then_execute()
    else:
        ask_user_to_confirm()
```

閾値はリスクに比例させる。 同じ判断でも、取り消せない操作ほど高い確信度を要求する。閾値は自社データで確信度×正解率をプロットして決める。

出典: docs.typesafe.ai「Confidence」

Use Cases

活用領域と、職種別の着眼点

スマートなif文

分類・ルーティング・採点・抽出など、手書きルールが脆い分岐

大量データのmap-reduce

大規模テキストを特徴量・インサイトに変換

リアルタイム機能

約100msならUXを損なわずにAIを差し込める

LLMの検証・ガードレール

入出力の採点、脱獄検知、引用の裏取り

エンジニア
LLMの出力パース・リトライ処理を削減。プロンプトより「問いのスキーマ設計」とテストが中心に。

デザイナー
入力中の即時判定など待ち時間のないAI体験。確信度「中」を扱う確認UI・不確実性の見せ方が設計対象に。

右の職種別観点は、公式情報を踏まえた当社の推測・提案を含む

出典: TypeSafe AI Blog「Use cases」、docs.typesafe.ai Cookbooks一覧

Early Signals

初期ユーザーによる試用報告（X投稿ベース）

分類・ルーティング

@MichaelLee04

約5,000リクエストで約$2。分類・モデルルーティング・意図判定などに使用。スレッドではp50 150ms／p95 350msと紹介。

メール分類

@ryanvogel

自分のメール約1,500通で試し、分類精度の高さに驚いたと投稿。LLMでは割高な地味タスクの代替候補。

LLM-as-a-judge

@Yuchenj_UW

多数のプロンプトを試しても$0.10を使い切れなかったと投稿。評価器用途が高速・低コスト化する可能性。

オセロ

@4ba_ba_baba

盤面状態を渡し最大確率のマスに着手。探索アルゴリズム（α β剪定）の評価と概ね一致して見えると報告。

扱い方　個人の初日試用報告であり、評価条件や再現性は未検証。社内判断の根拠にはせず、PoCで自社データを使って確かめる前提の「兆候」として扱う。

出典: AGIラボ(@ctgptlb)のXスレッド(2026-09-16)と、引用された各投稿

OSS

コミュニティ実装カタログ（GitHub）

fhshaik/typesafe-mario

エミュレータのRAMを構造化JSONにしてマリオを操作。画像は送らない

lukaske/jev-doom-agent

ブラウザ上のDoom（WASM）。空間状態＋判断ログ、公式デモと同系で約$7/時

jarrodwatts/jev-trader

ブロックごとに売買を1判断。モデル遅延81msと作者が報告

awlevin/typesafe-computer-use

macOS操作をOCR＋Choiceで実行。1判断$0.0002（Opus 5比で約1/160）と作者が報告

devagrawal09/jev-review

段階的コードレビュー。Noulのリスク行列→重大度判定→担当振り分け

DevMortimer/pi-warden

エージェントの逸脱・堂々巡り・不可逆操作を検知するガードレール

扱い方　いずれも個人の実験で公式提供ではない。数値も作者の自己申告。読むべきは「状態をどう型に落としたか」という設計で、社内PoCの雛形として参照する。

出典: GitHub各リポジトリのREADME、awesome-jev-usecases（2026年9月時点）

Caveats

限界と、評価時の注意点

できないこと	文章生成・推論過程の説明はしない。重い判断は問いに分解し、コード側で再結合する必要がある。
ベンチの中立性	比較基準は GPT-6 Astra と Fable 5.1 の平均で、ワークフローも自社チーム作成。偏りの可能性を本人たちも明記。
「幻覚しない」の範囲	保証されるのは出力構造の適合のみ。構造が正しくても、判断内容が誤っている可能性は残る（X上でも同様の指摘あり）。
提供状況	early accessでウェイトリスト制。本番導入事例は未公表で、第三者による独立ベンチマークもまだない。
入力の上限	入力は約32,000トークンが上限との報道。問い合わせ1件は十分でも、長い契約書などは前処理が必要。
未確認事項（要検証）	日本語精度、国内からの実測レイテンシ、データ保持・学習利用の条件、SLA。ISMS審査前に規約確認が必要。

出典: TypeSafe AI Blog「Nuance」各節、NewsBytes、@pipix1121のX投稿（いずれも2026-09-16）

Next Actions

次アクション

1 利用条件の確認

利用規約・プライバシーポリシーでデータ保持と学習利用を確認し、機密データ投入可否を判断

2 ウェイトリスト登録

typesafe.ai から申請。Playgroundで日本語の問い合わせ文を使い精度と応答時間を実測

3 PoC題材を1つ選ぶ

既存のLLM分類・ルーティング処理を置き換え、速度・コスト・一致率を比較

4 デザイン検討

確信度3段階に応じたUIパターン（自動／確認／エスカレーション）を試作

参考: typesafe.ai/blog/introducing-system-one-models-and-jev ／ docs.typesafe.ai ／ evals.typesafe.ai ／ The Register・AI News・NewsBytes ／ AGIラボ(@ctgptlb)のXスレッド ／ GitHub: fhshaik/typesafe-mario ほかコミュニティ実装