Task Completion Detection

A Study in the Context of Intelligent Systems

Ryen W. White, Ahmed Hassan Awadallah, Robert Sim
Microsoft Research

Challenges in Task Management

- Intelligent systems (digital assistants, etc.) store / remind users about tasks

- Tasks can be explicitly specified or inferred (e.g., from email)

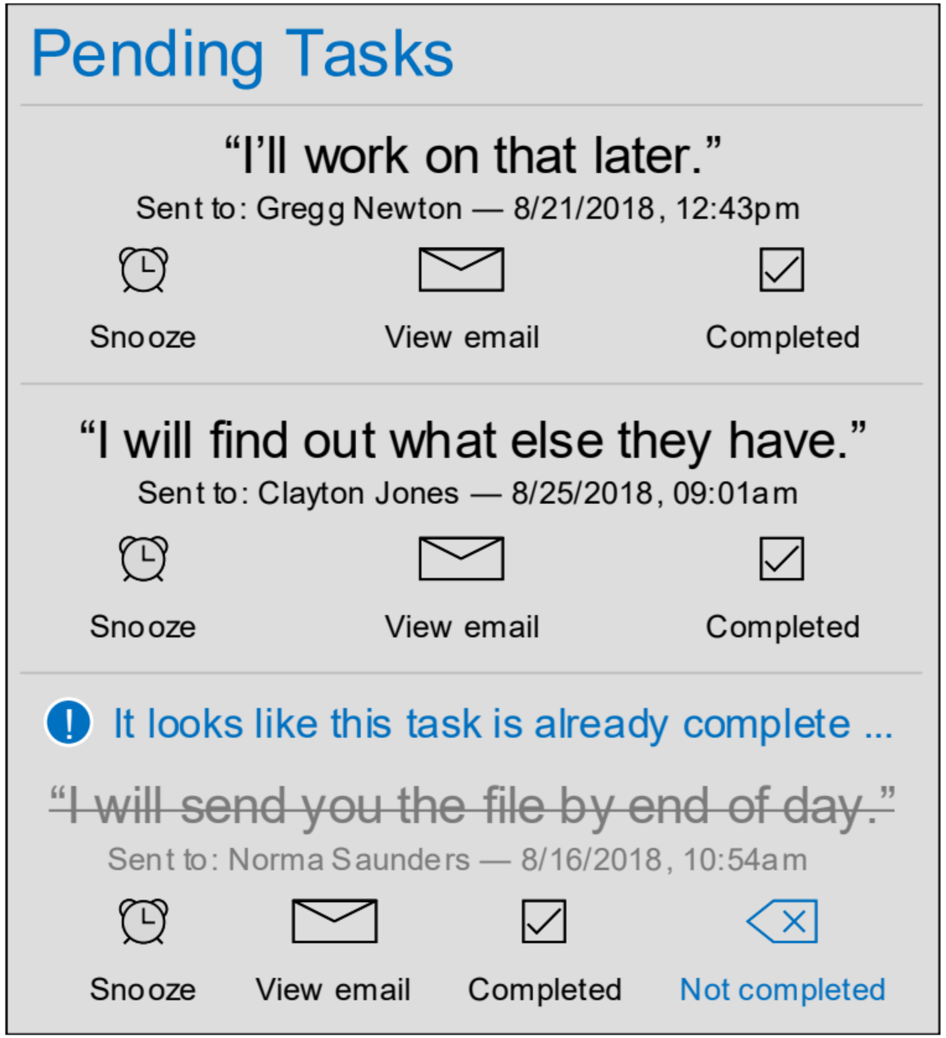
- Users face least two challenges:

  1. Task lists grow over time making it difficult to focus attention on pending tasks
  2. By ignoring task status, systems can remind users about completed tasks

- Methods to more intelligently flag completed tasks are required

Example Scenario: Task Auto-Deprecation

- Show pending tasks (e.g., commitments)
- Flag or deprecate completion candidates
- Provide recourse links to undo
- Other applications possible, incl. task ranking, task prioritization, etc.
- Focus on <u>reminder/notification suppression</u>



This Study

- Introduce <u>task completion detection</u> as an important new ML challenge
- Analyze data from popular digital assistant (Microsoft Cortana)
  - Reveal trends in temporal dynamics of completion per task attributes
- Train ML classifiers to detect task completion
  - Use many signals, including time elapsed, context, task characteristics
- Present design implications for intelligent systems from being able to automatically detect task completion

Commitments Data

- 1.2M consenting users of Microsoft Cortana in en-US
- Cortana tracks commitments made by users in outgoing email, e.g.,
  - *"I will send you the report"*
  - *"I'll get back to you by EOD"*
  - *"I'll work on it this evening"*
  - *"Will get back to you next week"*

= Tasks in our study

- 3M commitments collected during 2017-18 (avg. ~2.3 per user)
- Commitments persist in system for max 14 days (our focus here)

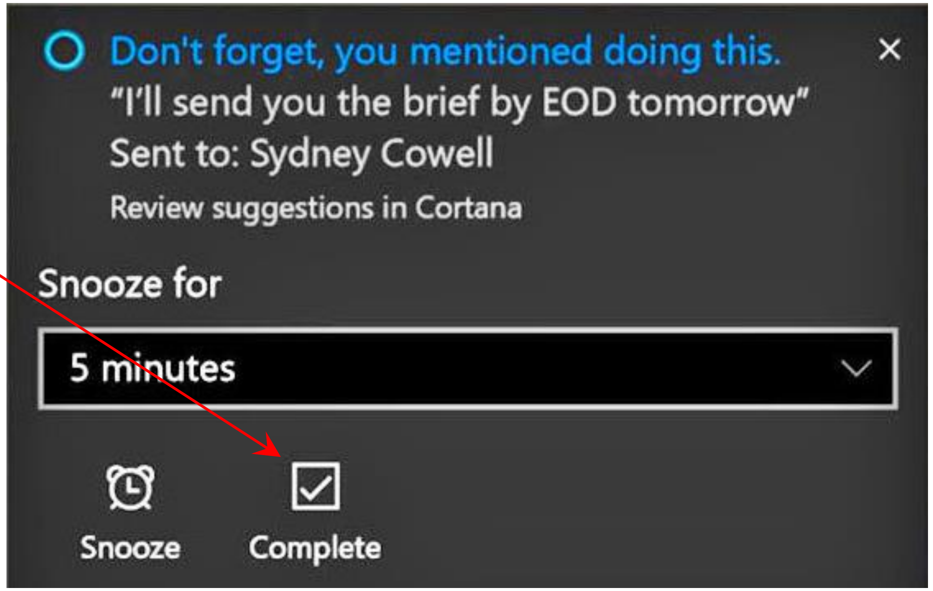
Commitment Meta-Data

- E.g., due dates ("I'll get this to you by <u>next Friday</u>")
- Extracted from commitment text using proprietary methods
- Statistics:
  - 24% of commitments have a due date
  - Due dates fall within avg 1.78 days of commitment (stddev 3.62, med 0.71)
  - Most commitments (86.3%) are made on weekdays
  - Presence of intervening weekend days increases time until due date

Labeling Methodology (1 of 2)

- Use Cortana commitments usage data to compute completion labels

- Cortana has a feedback affordance for users to indicate task completion



- "Complete" clicks help form ground truth
  - Only says task was <u>completed BY some time, not WHEN the task completion occurred</u>

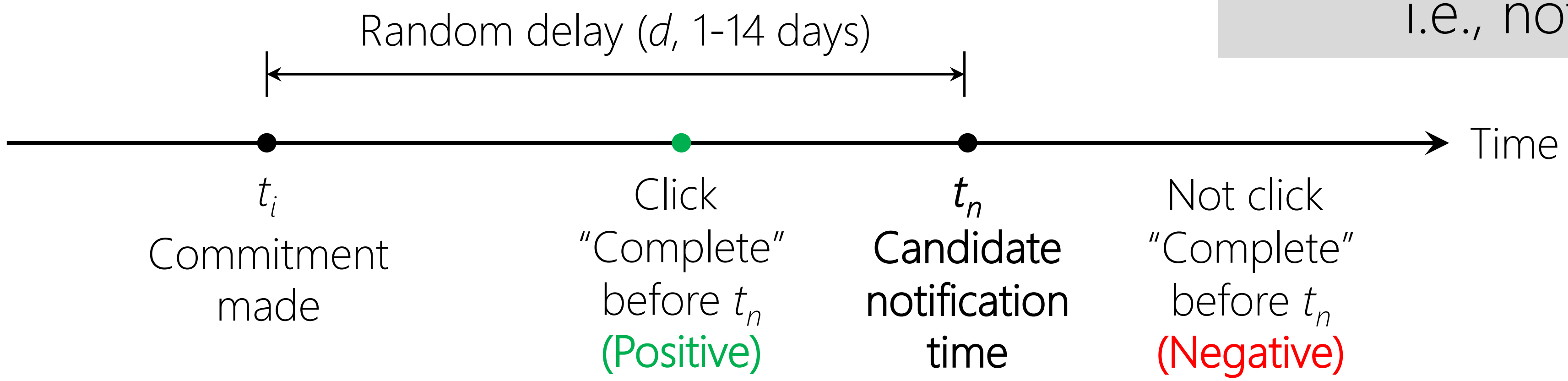
- OUR GOAL: Only remind/notify users for tasks that are not yet completed

Labeling Methodology (2 of 2)

- For each of 3M commitment tasks:

GOAL: Only remind/notify users for tasks that are not yet completed

i.e., not complete by t_n



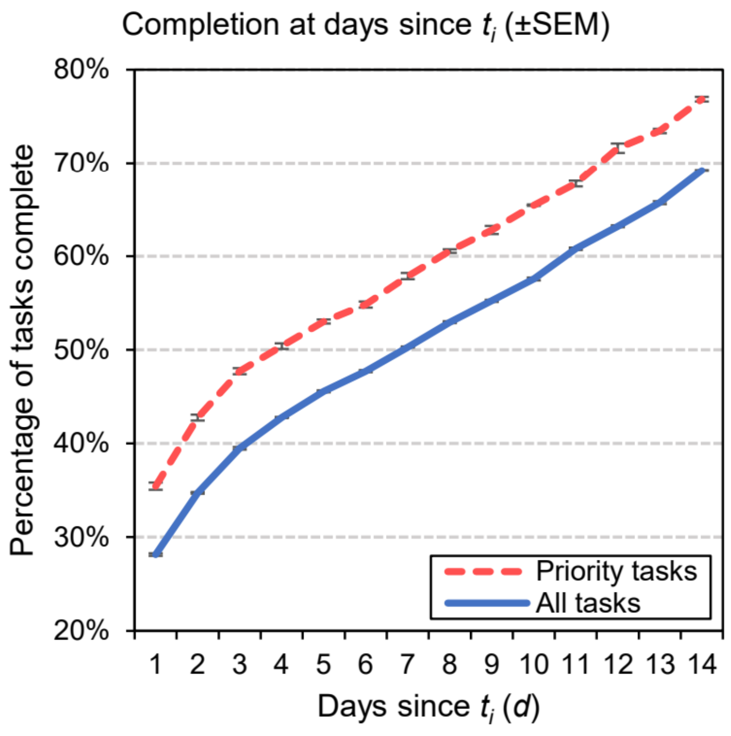
- Label distribution: 1.53M positive (51%) and 1.47M negative (49%)

- Task completion is time dependent (i.e., more tasks get done over time)

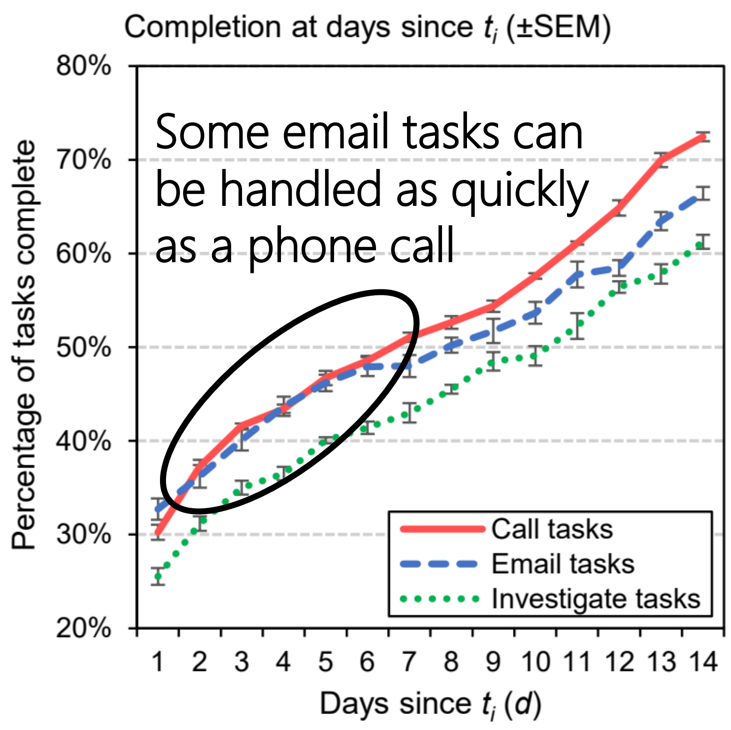
Temporal Dynamics

Task Completion Over Time

- Compute fraction of tasks completed at t_n, all tasks and per task type
  - Task type by priority (high-pri language) and by activity (call, email, investigate)



High priority tasks are completed faster



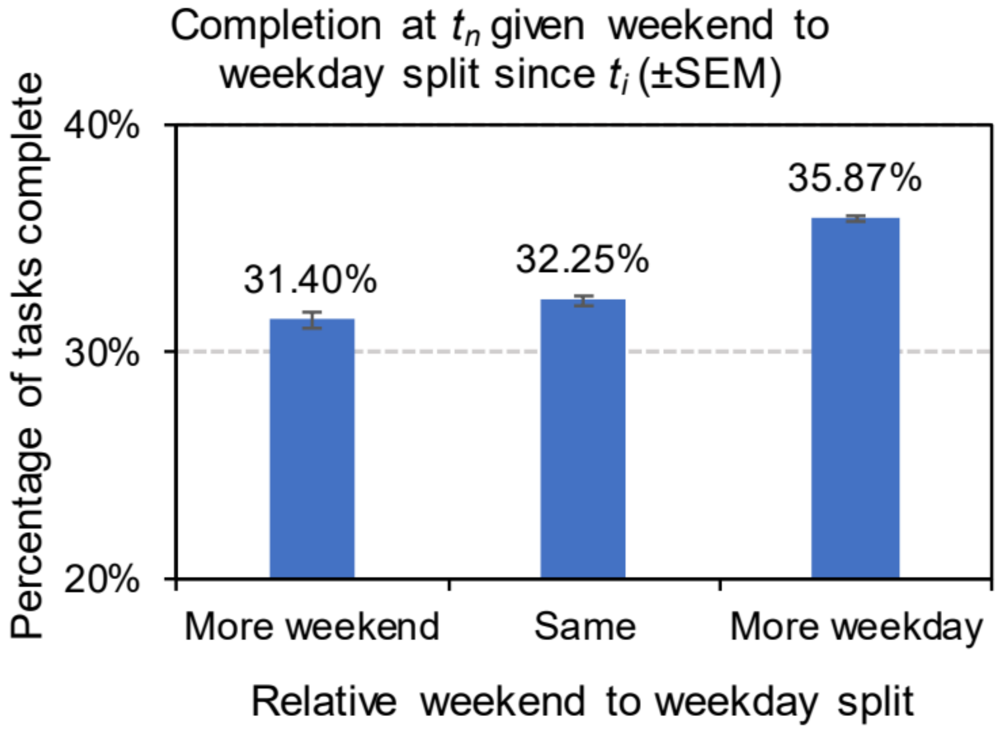
Relative completion timing: Call < Email < Investigate
Connected to avg relative complexity

Weekend vs. Weekday

- Studied differences in number of weekend days and weekdays between commitment made (t_i) and notification time (t_n)

- Focus on d=2 to control for confounds

- Three groups:
    1. **More weekend** (2 weekend, 0 weekday)
    2. **Same** (1 weekend, 1 weekday)
    3. **More weekday** (0 weekend, 2 weekday)



- Task completion % higher when there are more weekdays

Detecting Task Completion

Methods

- Train binary classifiers to detect completion of pending task by notification time (t_n) using many signals

- Use completion labels from "Complete" clicks as ground truth

- Five feature classes:
  - **Time:** time elapsed since task created, #weekend days, #weekdays
  - **Commitment:** n-grams, verbs, priority, due date, is conditional, intent, etc.
  - **Email:** subject n-grams (no email body), is reply, number of recipients, etc.
  - **Notifications:** logged Cortana notifications (16% of tasks), num notifications, etc.
  - **User:** >1 commitments (38% of users), historic tasks, completion time/rates, etc.

Learning Algorithms

- Logistic Regression
  - + Compact, interpretable models
  - + Used previously for task modeling on email*

- Gradient Boosting Decision Trees
  - + Efficiency, accuracy, robustness to missing/noisy data, interpretability
  - + LightGBM (used here) optimized for speed and low memory consumption

- Neural Networks – bi-directional RNN with GRU and attention
  - + State-of-the-art NLU performance

* Corston-Oliver, S., Ringger, E., Gamon, M., & Campbell, R. (2004). Task-focused summarization of email. In *Text Summarization Branches Out* (pp. 43-50).

Evaluation

- Split 3M commitments into training (2.9M), validation (50k), testing (50k)
- Stratified commitments by user (user only in one of train/valid/test)
- Tuned model hyperparameters on validation set
- Computed accuracy, F1, precision-recall
- Sig: Two-tailed t-tests with bootstrap sampling (n=10)

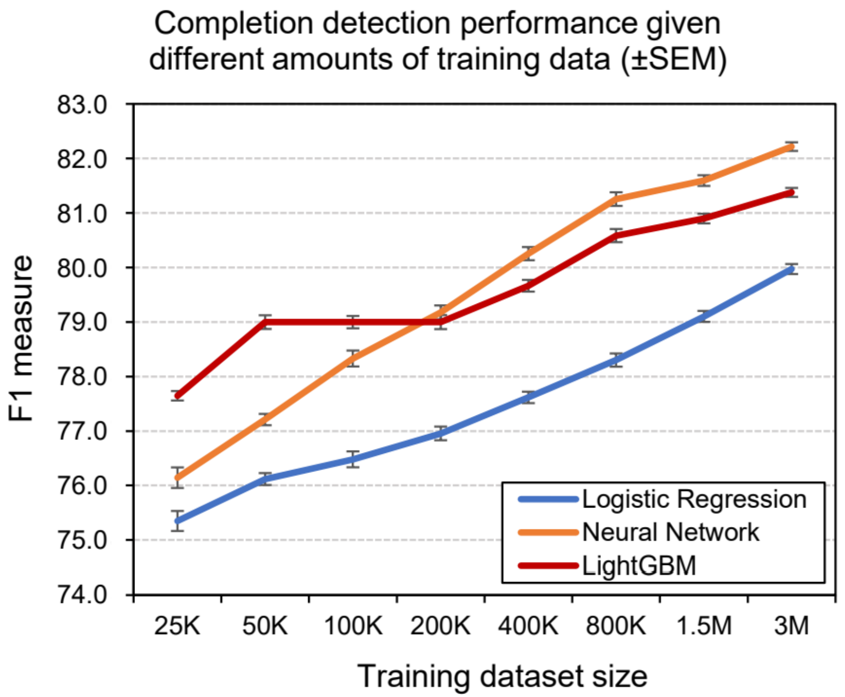
Findings

- Overall
  - LR model performs worst
  - LightGBM and NN perform similarly
  - LightGBM simpler, more interpretable, faster to train
  - NN can better encode text (not needed)
- Effect of data volume
  - Vary training set from 25K to 3M
  - LR model performs worst at all data points
  - LightGBM and NN outperform LR
  - LightGBM better for less data (≤100K)
  - NN better for more data (≥200K)

Overall model performance

All paired differences in F1 significant at $p < .01$

Model	Precision	Recall	F1	Accuracy
Logistic Regression	87.17	73.87	79.97	81.11
LightGBM	78.92	83.90	81.37	80.48
Neural Network	87.67	77.40	82.21	83.00



Findings

- Effect of features used
  - Used LightGBM (faster, etc.)
  - Two complementary strategies
    - Dropped feature classes, one-by-one
    - Trained on one feature class at a time
  - Ablation Findings
    - Removing Time/Email/Notification has little effect
      - Substitutable with other features (notifications)
    - Removing Commitment Text has little effect
      - Features captured elsewhere (verbs, etc.)
  - One-Class Findings:
    - Commitment features most important
    - User features are also strong
      - Personalization or user segmentation (?)

Removing one feature class at a time

Note: Differences in F1 vs. All Features significant at * p <.05 and ** p < .01

Model	F1	% Δ	Acc	% Δ
All feature classes	81.37	—	80.48	—
— Commitment**	69.38	−14.75%	69.60	−13.52%
— User**	75.03	−7.80%	74.96	−6.86%
— Time**	80.66	−0.86%	79.62	−1.07%
— Email*	80.95	−0.52%	80.02	−0.58%
— Commitment Text	81.13	−0.29%	80.19	−0.37%
— Notifications	81.44	+0.08%	80.53	+0.06%

Training on one class at a time

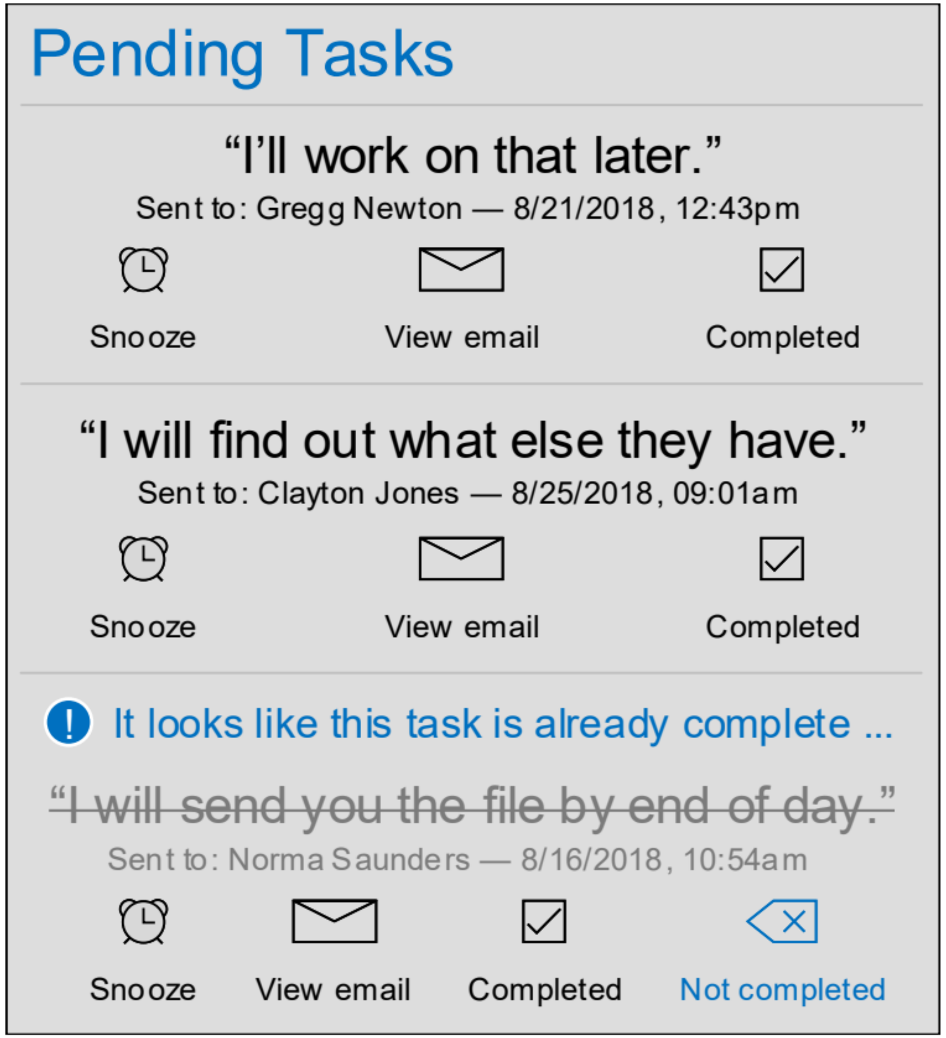
Note: Differences in F1 vs. All Features significant at * p <.05 and ** p < .01

Model	F1	% Δ	Acc	% Δ
All feature classes	81.37	—	80.48	—
Commitment Only**	71.01	−12.72%	71.06	−11.70%
User Only**	66.74	−17.98%	71.61	−11.02%
Email Only**	64.37	−20.89%	62.67	−22.13%
Commitment Text Only**	60.20	−26.02%	61.18	−23.98%
Time Only**	59.26	−27.17%	59.84	−25.64%
Notifications Only**	28.45	−65.04%	54.55	−32.22%

Discussion

- Accurately detect completion, although focused on one (notifications) scenario
- Need to understand how users respond
  - Incl. UX designed to help not hinder users
- Measured independently, on all users
  - Likely used in a pipeline, on user segments
- Task <u>progression</u> is important
  - More general problem than task completion

"Auto-deprecation" experience from Slide 3



Summary and Takeaways

- Detecting task completion important challenge in intelligent systems
  - Help users focus on what needs their attention (vs. what has been done)

- Showed strong performance (~83%) for one scenario (notifications)

- Need to explore more sophisticated ML, richer signal collection, expand to other scenarios and task types, etc.

- Need to work with users to understand the impact of completion detection
  - Esp. when the experience is visibly altered by the task completion inference