No. 27 Crowdsourcing

馬場 雪乃（国立情報学研究所）

2014年5月31日 ICDE 2014 勉強会

本セッション：クラウドソーシングを用い人間を組み込むシステムを作る際の費用・品質の問題を解決

- 本セッションでのクラウドソーシング：
  「お金を払うと
  正しい答えを返してくれる」仕組み
  - 費用削減のため、問い合わせ回数は減らしたい
  - 中身は人間なので、間違えることがある
- 論文ごとに異なるアプリケーションを想定
  - １本め：表のスキーママッチング
  - ２本め：文章中のエンティティペアの関係抽出
  - ３本め：アイテム選択

１本め概要：機械処理と人力の組み合わせによる表のスキーママッチング

- **“A Hybrid Machine-Crowdsourcing System for Matching Web Tables”**
  - 表の中の各列が何を表しているのか知りたい
    - 表のスキーママッチングに使う
  - 機械処理とクラウドソーシングの組み合わせ

赤枠：Film/Title
青枠：Film/Director

Table T_1: *Top Rated Movies*

Title	Directed By	Language
Les Misérables	T. Hooper	EN
Life of PI	A. Lee	EN
Inception	C. Nolan	EN

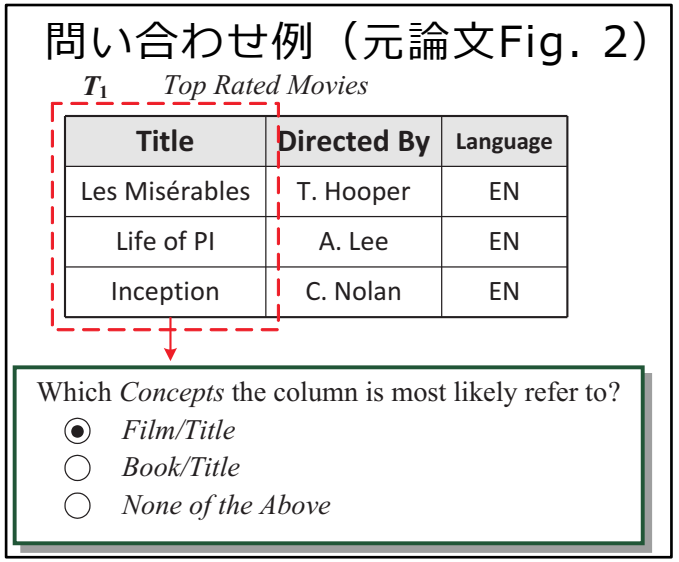
Table T_2: *Highest Weekend Box Office*

Movie	Director	Written By
Up	P. Docter	P. Docter
Avatar	J. Cameron	J. Cameron
The Pianist	R. Polanski	W. Szpilman

（元論文Fig.1）

提案：費用削減のため「機械では迷う列」「影響力が大きい列」を選んで人間に問い合わせ

- タスク：概念辞書から各列に適する概念を選ぶ
- 費用削減のため一部の列だけ人間に問い合わせ
- 列の選択方針：
  - 機械では迷う列
    - 例：”Film/Title”も”Book/Title”も両方正しそう
  - 影響力が大きい列
    - 問い合わせ結果を他の列の予測にも利用
  - 上記選択方針にしたがって、列集合の効用関数を設計



提案（続き）：効用関数の劣モジュラ性を利用、「人に聞くべき」列集合を貪欲法で得る

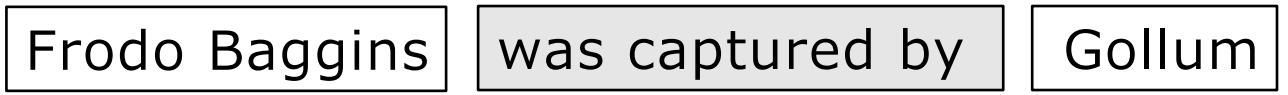
- 具体的な手順
  - まず機械処理で「列、概念」の関連度を算出
    - 列内の要素（例：”Avatar”）と
      概念辞書中の各概念のインスタンスの類似度を利用
  - 効用関数に従いK個の列を選択、人間に問い合わせ
  - 問い合わせ結果を用い関連度を更新、各列の概念を決定
- 効用関数は列集合に対して定義されている
  →効用関数を最大にするK個の列の選択はNP困難
  →効用関数に劣モジュラ性があるので貪欲法で近似的に解く
- 結果：2%・10%の列の問い合わせでaccuracy60%・75%

２本め概要：機械処理と人力の組み合わせによる文章中のエンティティペアの関係抽出

- **“Combining Information Extraction and Human Computing for Crowdsourced Knowledge Acquisition”**
  - 文章中に現れるエンティティ（例：人物）同士の関係を抽出したい
  - （前論文と同じく）機械処理とクラウドソーシングの組み合わせ

入力：文章

出力例：



提案：機械的に抽出した後、人間がフィルタリング。答えやすいように選択肢をばらつかせる等の工夫

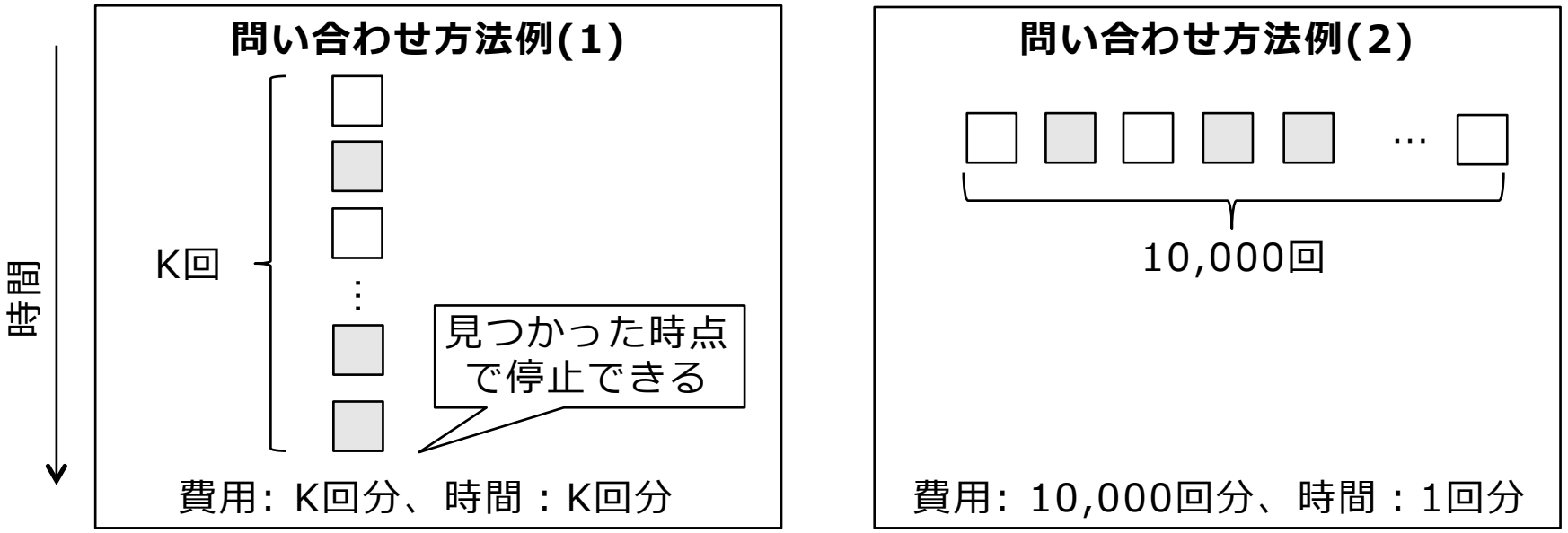
- 方針：
  - まずは機械的に抽出（Recall高め）
  - 人間に問い合わせて誤りを除去（Precision向上）
- 人間に問い合わせる際、答えやすくするために工夫
  - 似た関係が選択肢に同時に現れないようにする
  - 興味がありそうな人に聞く
- 報酬設計による品質管理：同じ問題を複数人に出題、回答が一致した場合はボーナスを支払う
- 結果：Precision 53%～85%、Recall 58%～98%

３本め概要：クラウドソーシングによるアイテム選択において費用と所要時間のバランスを取る

- **“Crowd-Powered Find Algorithms”**
  - クラウドソーシングを利用したアイテム選択
    - 例：10,000人の候補者から
      「英語が話せる人を10人」選びたい
    - 各アイテムが条件を満たすかどうかは
      人間に問い合わせないとわからない
  - 問い合わせ方法を工夫して、
    費用と所要時間のバランスを取る

背景：所要時間短縮のために並列化すると余分な費用が発生する

- 問い合わせ方法例(1)：1アイテムずつ順に問い合わせる。
  余分な費用は発生しないが時間が掛かる
- 問い合わせ方法例(2)：全アイテム並列問い合わせ。
  アイテム数分の費用が掛かるが1回分の時間で済む

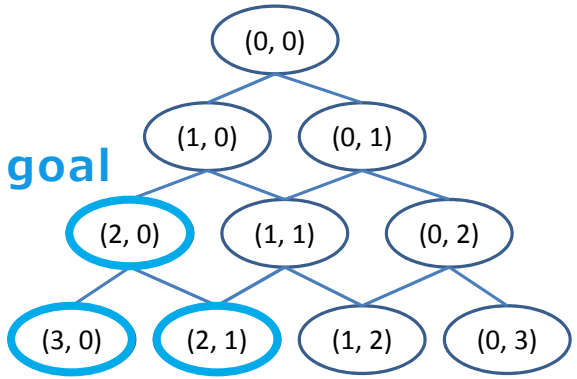


※アイテムの問い合わせ順は決まっているとする

提案（続き）：必ず正しい答えが返ってくる場合は、毎ステップ並列度を変えることで最適化

- 費用・所要時間最適化アルゴリズム：
  - 現在の状態からgoalへの最短パス数をKとする
  - K個の並列問い合わせを実行
- 人間が必ず正しい答えを返す場合はこれでOK

2個のアイテムを選びたいとき
（元論文Fig. 2）



（**n1**, **n2**）
n1: 「条件を満たす」ことがわかったアイテム数
n2: 「条件を満たさない」ことがわかったアイテム数

提案：誤った結果が返される可能性を考慮した費用・所要時間最適のアルゴリズムを提案

- 問い合わせ先は人間なので間違えることもある
  - ここがクラウドソーシングならでは
- 以下の設定で、
  費用・所要時間の最適化アルゴリズムを提案
  - 人間のエラー率は既知
  - 同じアイテムを複数人に問い合わせて多数決で統合

提案（続き）：各状態で「次に目的のアイテムを発見するのに必要な費用」を考える

- 各アイテムの状態を(n1, n2)で表現
  - n1: 「条件を満たす」と答えた人の数
  - n2: 「条件を満たさない」と答えた人の数
  - 奇数M人で多数決をとる：
    n1>(M+1)/2なら「条件を満たす」と確定
- 各状態の費用：Y(n1, n2)
  - 状態(n1, n2)から、次に「条件を満たす」アイテムを発見するまでに掛かる費用
  - エラー率から計算できる

提案（続き）：費用が少ないアイテムについて並列問い合わせ。各アイテムの問い合わせ回数も最適化

- アルゴリズムは毎ステップ以下を返す：
  - 問い合わせるアイテム、各アイテムの問い合わせ回数
- 費用最適化アルゴリズム：
  毎回、Yが最小のアイテムについて問い合わせる
- 費用・所要時間最適化アルゴリズム：
  - 毎回「必要数-現在の発見数」個だけ、
    Yが小さい方から順にアイテムを選ぶ
  - 各アイテムについて以下の並列度で問い合わせ：
    min(「『条件を満たす』と確定するのに必要な数」、
    「そのアイテムを諦めるのに必要な数」)