Motivation
- Our objective: create a model that learns to extract revenue information from richly formatted financial 10-Q filings
- Complicated task because attributes and relations are expressed in a combination of textual, structural, tabular and visual signals

Dataset
- Get Data from SEC
  - PostgreSQL
  - 2,007 reports
    - HTML format
    - Text spans: ~60 million
- Structural model
  - Text extraction (Stanford)

Defining training candidates
- Candidate - (Date, Revenue) pair
- Potential # candidates - 60 mm * 40 mm
- Total: 3.5e+15

Hard-filtering – limit # of candidates
- Logical, tabular, format, content, linguistic, RegEx rules to limit # of candidates
- Too many so we filter
  - Total 17,217 training samples

Weak supervision
- Manual labeling is unfeasible, so we use data programming (Snorkel)
- Labeling functions (LFs) evaluate the relation between the mentions of each candidate
  - Example labeling functions (not have 14 in total)

GAN for label generation
- Apply LFs to unlabeled data, resulting in a label matrix \( \mathbf{A} \). Then encode generative model \( p(\mathbf{A}, \mathbf{Y}) \) using three factor types:
  - Labeling propensity, accuracy, and pairwise correlations of LFs

Creating weak labels
- GAN marginal probabilities
- Custom marginal probabilities

Feature matrix
- 107,000 features: One-hot vectors of surrounding words, style features (font, size, bold, caps, etc.), NLP structure, length, lemma sequences, row and column headers, table characteristics, page location, object hierarchy, HTML tags, and others

Results
- Chosen model: Shallow [1000, 100, 300, 10]
- Neural network with a soft cross entropy loss function using Adam optimizer

Overfitting?
- Train
- Dev

Conclusions
- Through a combination of hard filters and weak supervision, our model was able to pinpoint a handful of revenue + date candidates out of billions of potential pairs
- F3 performance was slightly below 40%, with a baseline performance of 0% for the broad dataset and 20% for the filtered one
- Although performance was decent, more than half of the model’s performance was in the hard-filtering portion (20%).

Expansions
- Use fewer hard filters and train model with more candidates (over 1mm ideally)
- To avoid computation unfeasibility, trim features from 107,000 to less than 10,000
- Overall, creating a structural model is computationally inefficient for a task like this. Table and page extraction based on heuristics paired with a machine learning model would be more efficient

References

Team members
- Andrea Aguirre, andreaar@stanford.edu
- Roberto Seminario, rseminar@stanford.edu