Combining Personalized Agents to Improve Content-Based Recommendations

Jason M. Adams

LTI Speaking Requirement Talk
30 June 2008

Joint work with Paul N. Bennett and Anthony Tomasic

Outline

- Background
- Stacked Agents Model
- Implementation
- Results
- Conclusions

Background: Recommender Systems

- Recommender systems suggest items to users that the user might like or purchase
- Benefits businesses and consumers
  - Mitigates information overload for user
  - Drives sales for businesses
- Amazon, Netflix, etc.

Background: Recommender Systems

- User interest: ratings
  - Explicit
    - Discrete rating for each item
    - Boolean rating (like, dislike)
  - Implicit
    - Clicks on a link
    - Time viewing an item
    - Item purchase history

Background: Types

- Content-based
  - Recommend items similar to items the user has already expressed interest in
- Collaborative filtering
  - Recommend items liked by users with similar tastes
- Hybrid
  - Combination of content-based and collaborative filtering

Background: Content-based

- Advantages
  - Can recommend items with very little initial feedback from the user (user cold start)
- Disadvantages
  - Assumes user wants similar items
  - Requires existence of similar items (item cold start)
  - Fails to find novel items

Background: Collaborative filtering

- Advantages
  - Finds novel recommendations
  - Fits intuitions about user preferences
- Disadvantages
  - Results are poor when few ratings have been given (user cold start)
  - Results are poor when an item has been rated by few people (item cold start)

Outline

- Background
- **Stacked Agents Model**
  - Content-based
  - Confidence
  - Stacked Agents
- Implementation
- Results
- Conclusions

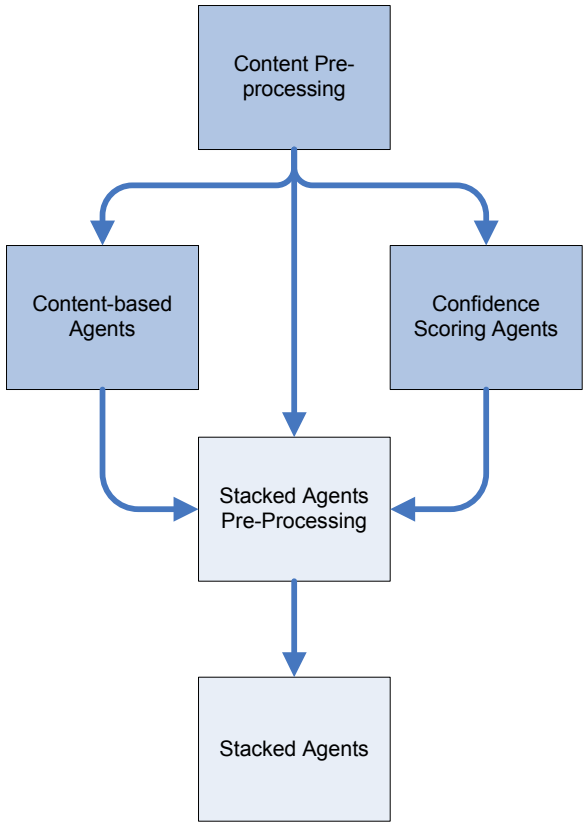
Stacked Agents Model

- The Problem: Data Sparsity
  - Users rate a limited number of items
  - Users could rate more items than they do
    - System only presents certain items to be rated
    - Users lose interest in entering ratings

- Our approach: Stacked Agents
  - Use predictions based on content of rated items to fill in the blanks (remove data sparsity).
  - Weight the predicted ratings based on how confident the system is that the user would rate the item
  - Run collaborative filtering algorithm on augmented data

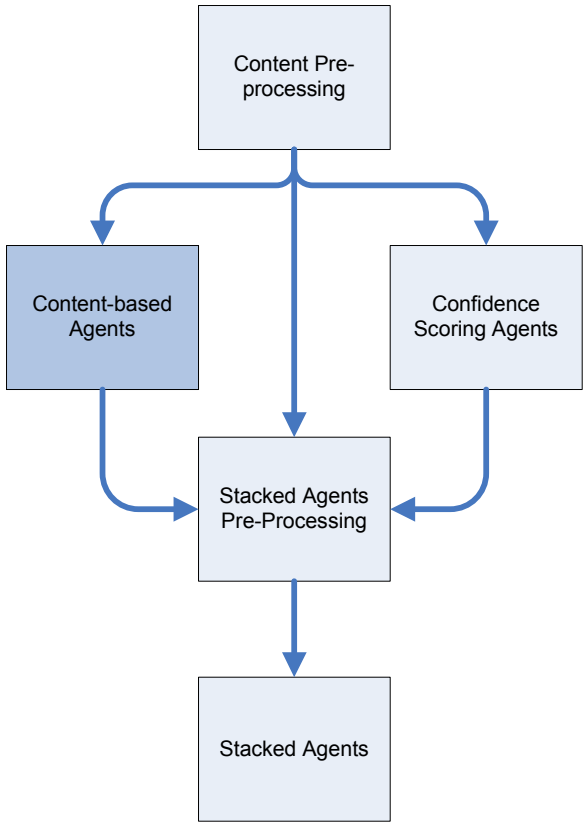
Stacked Agents Model

- Construct two initial models
  - Content-based recommender
    - Item-item similarity is used to predict ratings for items the user has not rated
  - Confidence model
    - Item-item similarity is used to predict whether the user would rate an item in the future



Stacked Agents Model

- Content-based agents
  - Each user is assigned an agent
  - Each agent builds a model
    - Explicit ratings by the user for each item
    - Features for each item rated
    - Weights assigned using tf-idf
  - Each agent predicts ratings for unrated items

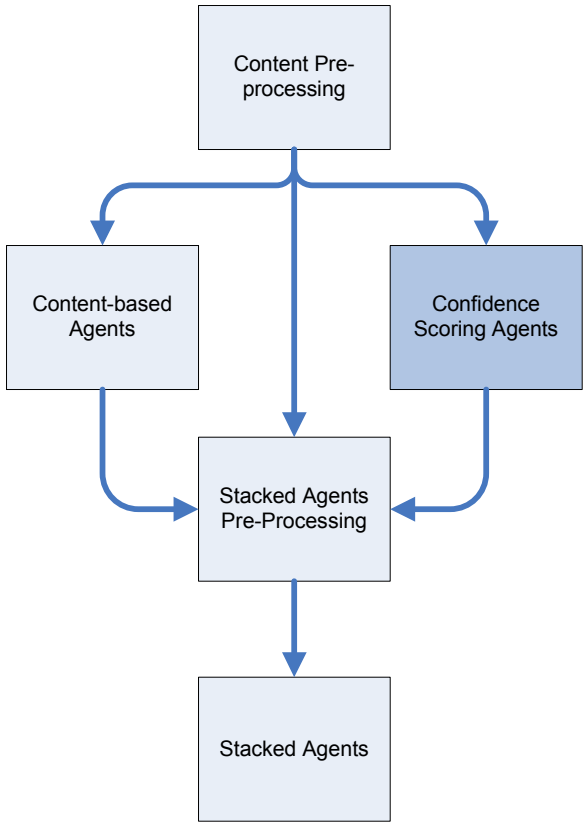


Stacked Agents Model

Item	Feature Vector	Labels	Predictions
1	$\vec{x}_1$	y_1	$\hat{y}_1$
2	$\vec{x}_2$	y_2	$\hat{y}_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
m-1	$\vec{x}_{m-1}$	y_{m-1}	$\hat{y}_{m-1}$
m	$\vec{x}_m$	y_m	$\hat{y}_m$

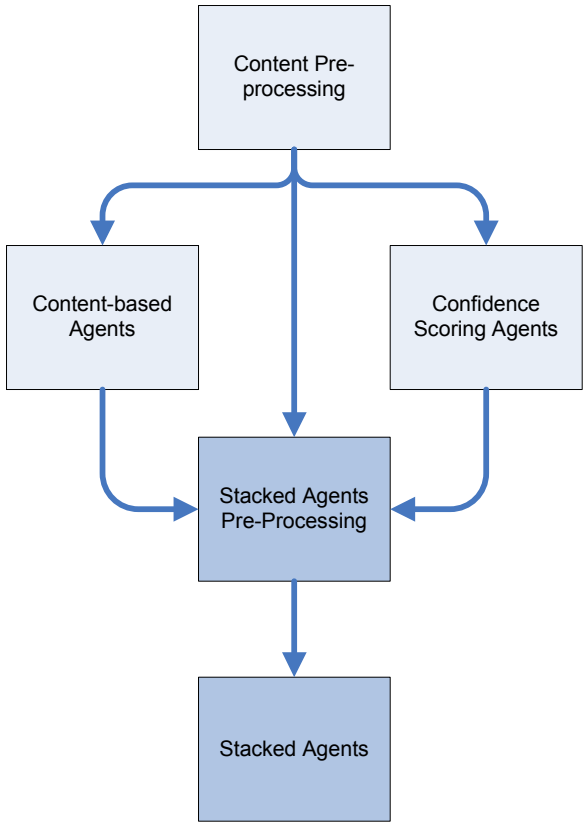
Stacked Agents Model

- Confidence Model
  - Classifier treats rated items as positive examples and unrated items as negative examples
  - Same features as content-based model
  - Drawbacks
    - Far more unrated items than rated
    - Unrated items are not missing at random



Stacked Agents Model

- Combine content-based and confidence models
  - Generate ratings predictions for all unrated items
  - Generate confidence predictions for all unrated items
  - Create agent to train new model and generate ratings predictions



Stacked Agents Model

- Combined predictions are computed:

$$\hat{r}_{u,v} = p_{u,v} \times q_{u,v}$$

- $q_{u,v}$ = confidence value
- $p_{u,v}$ = content-based prediction

Stacked Agents Model

- User profile model for stacked agent for user *i*.

Item	Feature Vector							Labels	Predictions
1	$\vec{x}_1$	$r_{1,2}$	$\dots$	$r_{1,i-1}$	$r_{1,i+1}$	$\dots$	$r_{1,n}$	y_1	$\hat{y}_1$
2	$\vec{x}_2$	$r_{2,2}$	$\dots$	$r_{2,i-1}$	$r_{2,i+1}$	$\dots$	$r_{2,n}$	y_2	$\hat{y}_2$
$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$	$\vdots$	$\dots$	$\vdots$	$\vdots$	$\vdots$
m-1	$\vec{x}_{m-1}$	$r_{m-1,2}$	$\dots$	$r_{m-1,i-1}$	$r_{m-1,i+1}$	$\dots$	$r_{m-1,n}$	y_{m-1}	$\hat{y}_{m-1}$
m	$\vec{x}_m$	$r_{m,2}$	$\dots$	$r_{m,i-1}$	$r_{m,i+1}$	$\dots$	$r_{m,n}$	y_m	$\hat{y}_m$
		$\underbrace{\qquad\qquad}_{\vec{r}_u}$							
	$\underbrace{\qquad\qquad}_{\vec{z}_u}$								

Outline

- Background
- Stacked Agents Model
- **Implementation**
- Results
- Conclusions

Implementation

- MovieLens data set
  - 6040 users
  - 3900 movies
  - 1 million ratings

  - http://grouplens.org

Implementation

Content features (from IMDB.com)

- Actors / Actresses
- Directors
- Producers
- Keywords
- Movie Length
- Plot summaries
- Genres
- MPAA Ratings
- Release Dates
- Production Companies

Implementation

- Used SVMLight for all models
- Linear kernel
  - Content-based
- Gaussian (rbf) kernel
  - Confidence model
  - Stacked Agents model
- Parameters tuned using 4-fold cross validation on training set

Implementation

- Delta space
  - Ratings were converted from scale [1, 5] to scale [-4, +4]
  - Rating subtracted from movie average
    - Ignoring the model's base user
    - Unique for each user
  - Improved results over flat ratings

Implementation

- Stacked pruning
  - Use content-features from content-based model when training the SVM?
  - Feature selection?
    - Threshold document frequency

Outline

- Background
- Stacked Agents Model
- Implementation
- **Results**
- Conclusions

Results

- Mean Absolute Error

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Results

- Normalized mean absolute error
  - For rating scale [1,5] divide by 1.6
  - Normalization factor computed by finding the MAE of random guesses
  - Allows comparison of data sets with different rating scales (Marlin, 2004)

Results

Content Features	Stacked Pruning	Content Model NMAE	Confidence Model Error	% Confidence in the Margin	Stacked Model NMAE	Improvement over Content
+	-	0.4405	0.0414	21.42%	0.456125	-3.547%
+	+	0.4405	0.0414	21.42%	0.455625	-3.434%
-	-	0.4405	0.0414	21.42%	0.433000	+1.703%
-	+	0.4405	0.0414	21.42%	0.433000	+1.703%

Outline

- Background
- Stacked Agents Model
- Implementation
- Results
- **Conclusions**

Conclusions

- Some contributions of this work:
  - Improve content-based recommendations with the stacked agents model
  - Delta space model improves SVM performance for CF

Conclusions

- Future Work:
  - Additional machine learning algorithms can be tried: random forests, kNN, etc
  - Additional feature selection techniques: Mutual information, Chi-squared, etc.
  - Issues of scaling on larger data sets

Questions?

Created using Wordle.net

References

- Marlin, B. Collaborative Filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004.