# An MDP-Based Recommender System

Autores: Guy Shani, Ronen I. Brafman, David Heckerman

Gabriel della Maggiora
Pontificia Universidad Católica

October 29, 2015

# Overview

# Motivación

la principal motivación de los autores es que ellos ven la recomendación no como un hecho individual sino como un proceso.

# Solución?

## MDPs

Pero no de la manera tradicional pues, en espacios con muchas acciones tardaría mucho en converger para una política óptima para algun usuario.

# Solución?

## N-Gram MDPs

# Idea de implementación

La idea de los N-Gram models, es intentar predecir la siguiente acción dado los n-1 eventos anteriores.

# Modelo

Recordemos que un MDP es una tupla $< S, A, T, R >$ donde

- S es el conjunto de estados, en donde cada estado representa la información relevante que tenemos sobre el usuario. Esta información corresponde a todas las acciones previas hechas por usuarios en la forma de un conjunto ordenado de secuencias de acciones. Por ende, el conjunto de estados contiene todas las posibles secuencias de acciones que puede hacer un usuario.
- A es un conjunto de las posibles acciones del sistema. Este conjunto de acciones se relaciona con la política. Diremos que una política $\pi$ es una funcion $\pi : S \rightarrow A$, en donde para $s \in S$ indica que $\pi(s) = a$ para $a \in A$.

# Modelo

- T una función de transición. Donde $T(s, a, s')$ es la probabilidad de terminar en el estado $s'$ si ejecutamos $a$ desde $s$.
- R una función de recompensa. Donde $R(s, a)$, asigna un valor escalar por cada accion $a$ ejecutada en $s$. Para efectos del problema, asumiremos recompensa negativa para cualquier estado no terminal y recompensa positiva por terminar en el objetivo.

# Inicialización Funcion de Transición

A diferencia de los MDPs tradicionales se debe contar con una función de transición ya aprendida. Utilizamos los siguientes supuestos. Para la cadena de markov para calcular los valores iniciales, se calculan las probabilidades de transición según el set de entrenamiento.

- Una recomendación aumenta la probabilidad que un usuario compre un item. Esta probilidad es proporcional a que el usuario compre el item en ausencia de una recomendacion denotamos esto con constante $\alpha > 1$

$$T(s, a, s') = \alpha T(s, s')$$

- La proabildad de que un usuario no compre un item que no fue recomendado es más baja que la probabildad que lo compre en ausencia de una recomendación, denotamos la constante de proporcionalidad como $\beta < 1$

$$T(s, a, s') = \beta T(s, s'), a(s) \neq s'$$

# Reinforcement Learning al Rescate

# Policy iteration

La idea del algoritmo de optimización es la siguiente, supongamos que el sistema está en el estado $S_0$, entonces dada una política inicial arbitraria, el sistema decide ejecutar $a_0$. En respuesta, el usuario elige alguna accion lo que lleva al sistema al estado $S_1$ y recompenza al agente con $r_0$. Es intuitivo querer encontrar la política $\pi$ tal que se maximize la recompenza.

# Policy iteration

Definimos que el valor de un estado $s$ bajo la política $\pi$, $V^{\pi}(s)$, como la recompenza acoumulada esperada, cuando el agente comienza en $s$ y sigue $\pi$ después. Entonces tenemos que:

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^{\pi}(s')$$

Donde $s' \in S$ es el siguiente estado alcanzado, cuando el agente toma la acción $\pi(s)$ en $s$. Entonces, para calcular el $\pi^*$ óptimo se reduce al siguiente problema de optimización:

$$V^{\pi^*}(s) = max_{\pi} E(\sum_{t=0}^{\infty} \gamma^t R_t)$$

# Policy iteration

$$V^{\pi^*}(s) = max_a R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{\pi^*}(s')$$

$$\pi^*(s) = argmax_a R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{\pi^*}(s')$$

# Recomendación adaptativa

Con el modelo dado y una política inicial arbitraria, el agente itera sobre los siguientes pasos en cada ejecución.

- Evaluación de la política: se computa $V^{\pi}$ como lo señalamos anteriormente.
- Mejora de la política: se utiliza $V^{\pi}$ para mejorar la política.

# Dataset

Se utilizó el set de datos Mitos, que contenía dos set de datos uno que eran las compras y otro que eran los distintos historiales de navegación de los usuarios. Se quitaron los items que fueron comprados o buscados menos de 100 veces y usuarios que solo compraron un solo item. La evaluacióm consistía en que para cada secuencia de usuario $t_1, ..., t_n$, se generaron los siguientes casos de pruebas $< t_1 >, < t_1, t_2 >, ..., < t_{n-k}, t_{n-k+1}, ..., t_{n-1} >$ Para cada caso, se utilizaron modelos para predecir la distribuición de probabilidad.

# Métrica de Evaluacion

- Recommendation Score: una recomendacion es correcta si está dentro de los top $m$ items recomendados, y se calcula el porcentage de acierto versus todas las recomendaciones hechas.
- Exponential Decay Score: Le da mayor puntaje a cada recomendación dependiendo de la posición mientras más abajo menor el puntaje y decae exponencialmente.

# Resultados

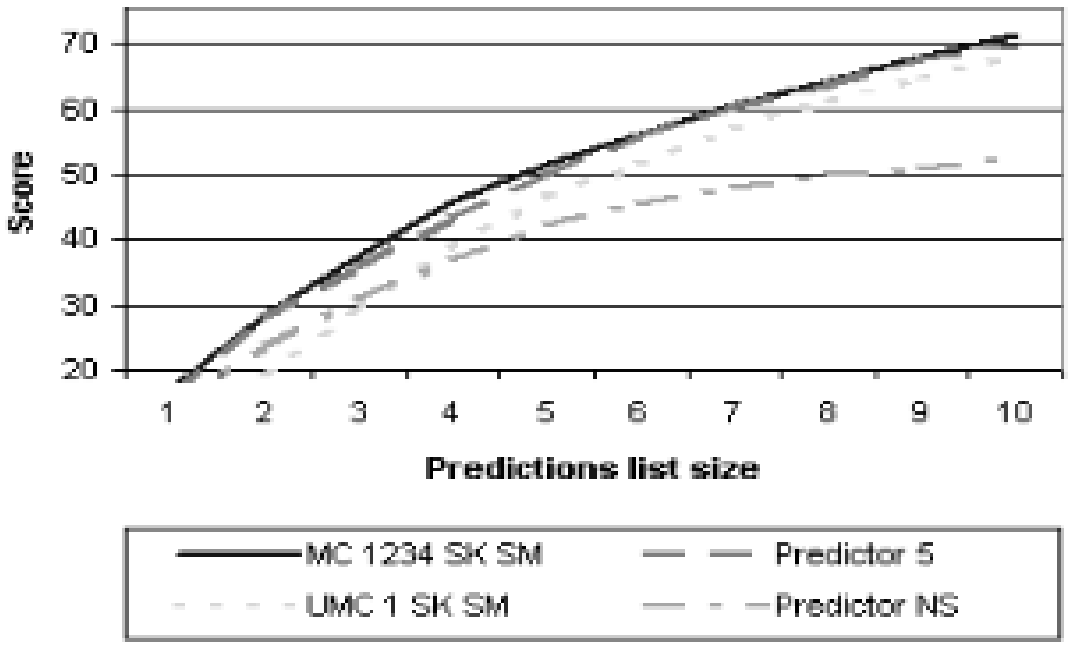


Figure: Recommendation Score Browsing

# Resultados

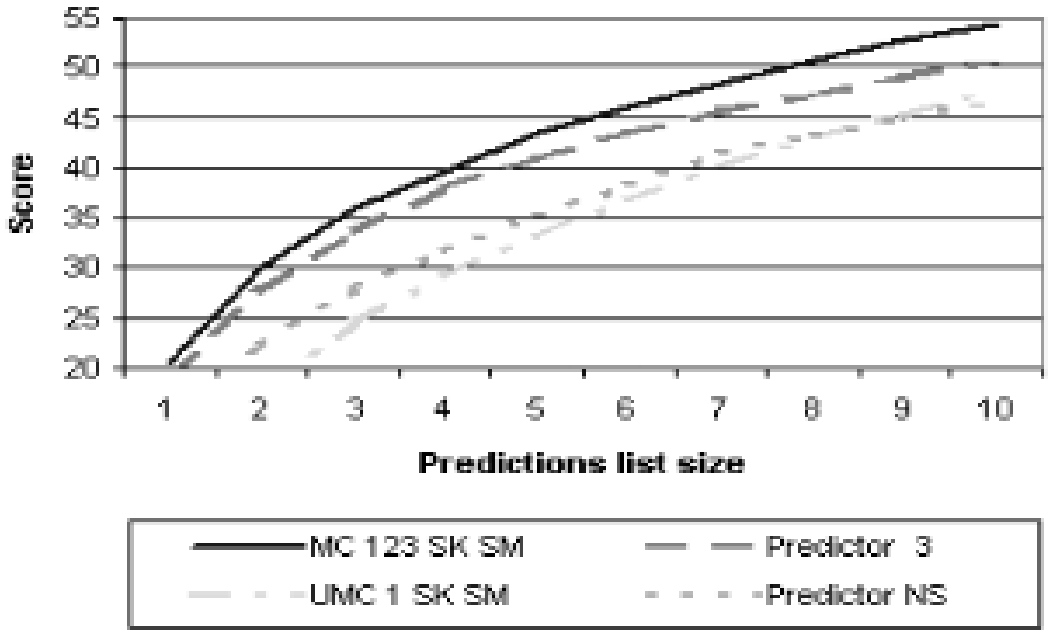


Figure: Recommendation Score Transactions

# Resultados

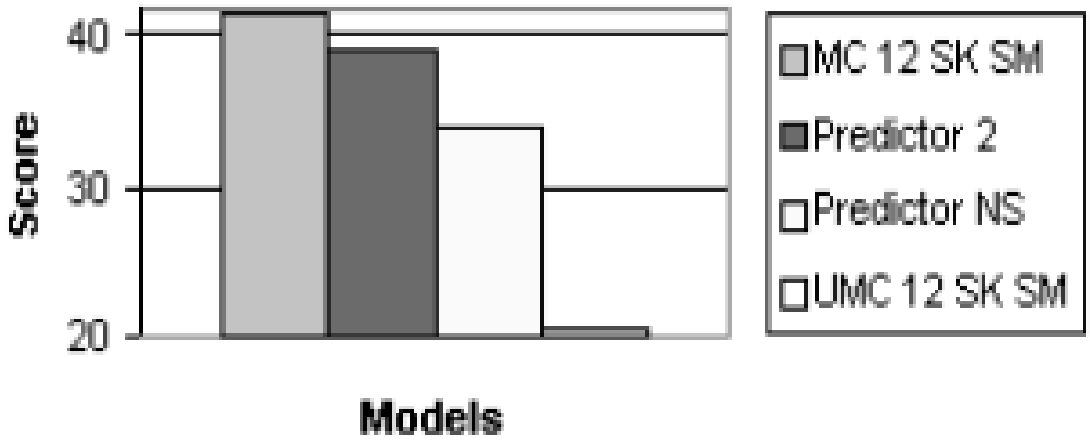


Figure: Exponential decay Transactions

# Resultados

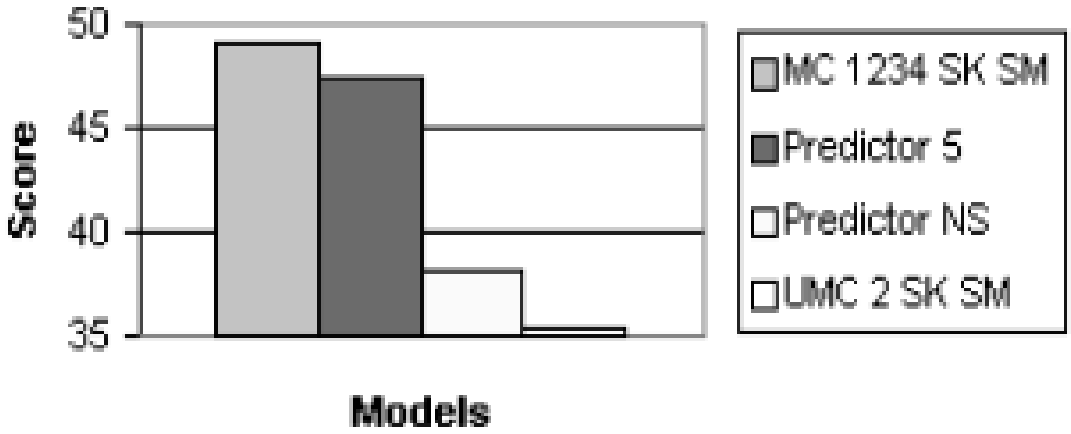


Figure: Exponential decay browsing

# Conclusiones y Trabajo a futuro

- Contribución conceptual para aportar en esta nueva visión de los sistemas recomendadores.
- Contribución técnica en donde se muestra como aplicar el concepto de modelos n-gram a sistemas recomendadores.