4 Experimentos com um acervo de vídeos

Neste capítulo descrevemos os experimentos realizados para testar as abordagens apresentadas. Primeiro, descrevemos as principais características de nosso conjunto de dados. Em seguida, os resultados dos experimentos são listados e discutidos.

4.1 Conjunto de dados

Em nossos experimentos utilizamos os dados do portal de vídeos da Globo.com [32]. Separamos os dados de consumo de vídeos nos catálogos do programa Auto Esporte [33] de janeiro a maio de 2011 com algo em torno de 77 mil registros de consumo de 396 vídeos por 3046 usuários, onde cada usuário viu pelo menos 18 vídeos.

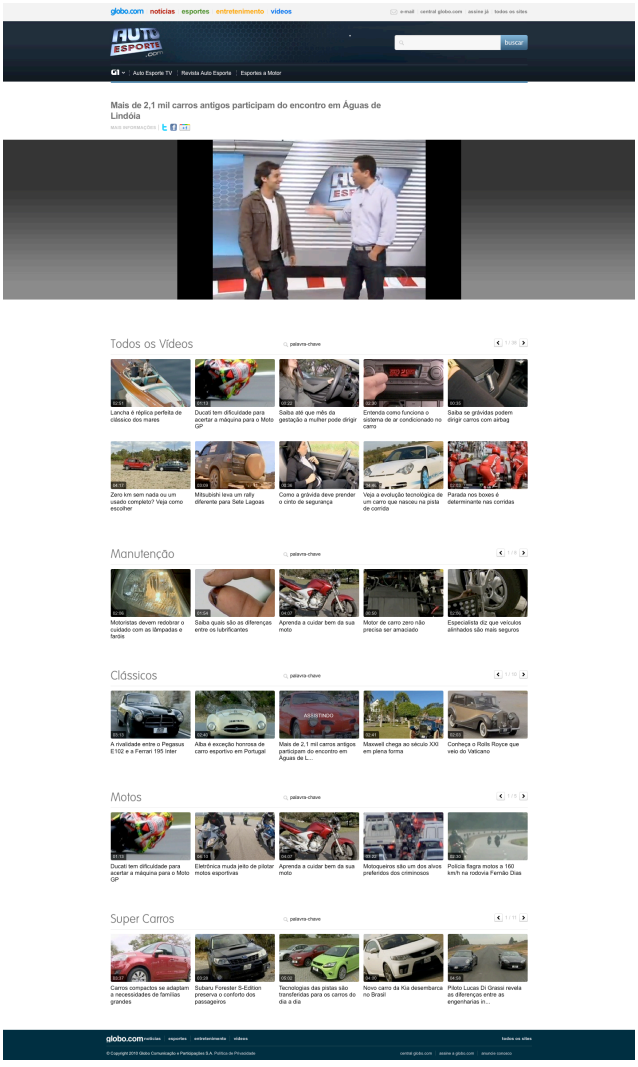

Figura 4.1 - Catálogo de vídeos do Auto Esporte

Os dados foram obtidos a partir dos servidores de borda da Globo.com responsáveis pela entrega dos arquivos de vídeo. A identificação do conteúdo destes arquivos é feita por um ID único que aparece no nome do arquivo.

`http://flashvideo.globo.com/h264/jornalismo/3//sptv/2011/08/09/EF_CGJ-RJ_T_`**`1591109`**`_mmp4.mp4`

Dado que a experiência de consumo no portal não é autenticada (ou logada), utilizamos o *cookie* do serviço Google Analytics (utilizado para contabilizar as métricas de acesso ao portal) [34] para a identificação dos usuários. Este *cookie* possui um parâmetro chamado *__utma* que faz a identificação de um visitante único e demora cerca de dois anos para expirar. A forma utilizada para a identificação dos usuários nos remete a dois problemas. O primeiro é que tratamos o mesmo usuário acessando os vídeos de máquinas ou *browsers* diferentes como sendo o mesmo usuário. O segundo é que se, por ventura, o usuário limpar o conteúdo cacheado de seu browser, seu próximo acesso será identificado como o de um outro usuário. Isso introduz ruídos no conjunto de dados que prejudicam os resultados das experimentações.

Considerando as avaliações dos usuários, os sistemas de recomendação costumam funcionar com diferentes tipos de dados de entrada. O mais conveniente é contar com o *feedback* explícito dos usuários, com os usuários indicando explicitamente seus interesses pelos produtos. No nosso caso, estes dados não estão disponíveis. Assim, tivemos que inferir as preferências dos usuários a partir de um *feedback* implícito, que reflete indiretamente a opinião do usuário através da observação de seu comportamento. Dentre os diferentes tipos de *feedback* implícito temos o histórico de compras em um *site* de comércio eletrônico, o histórico de navegação, padrões de busca ou até os movimentos do *mouse*. Por exemplo, um usuário que comprou diversos livros de um mesmo autor provavelmente gosta daquele autor.

Construímos o conjunto de dados de entrada a partir do *feedback* implícito dos usuários usando os registros de visualização dos vídeos no portal. Cada registro de visualização de vídeo em nosso conjunto de dados consiste de uma tupla "data e hora de acesso | identificador do vídeo | identificador do usuário". Não temos nenhum dado sobre o conteúdo do vídeo e características do

usuários. Assim, construímos nossa matriz de avaliações com "1" se um usuário requisitou um vídeo (avaliação positiva) e "-" se ele não viu. Essa matriz é uma matriz esparsa onde, aproximadamente, 94% de seus dados são desconhecidos.

	V_ID1	**V_ID2**	**V_ID3**	**V_ID4**	**V_ID5**	**V_ID6**	**V_ID7**	**V_ID8**	**V_ID9**	**V_ID10**
U_ID1	1	-	-	1	-	-	-	-	1	-
U_ID2	1	-	1	-	1	1	1	1	-	-
U_ID3	-	1	-	-	-	-	1	-	-	1
U_ID4	-	1	-	1	1	-	1	-	-	1
U_ID5	1	-	1	-	-	-	-	-	1	-

Tabela 4.1 - Matriz de visualizações de vídeos

Duas características desses dados implícitos devem ser observadas:

1. Não temos *feedback* negativo. Observando o comportamento dos usuários, podemos inferir que vídeos ele provavelmente gosta e, por isso, escolheu consumir. No entanto, não sabemos se um usuário viu um vídeo e não gostou. Por exemplo, um usuário que não assistiu um trecho de uma edição do programa, pode não tê-lo feito pelo simples fato de não saber da existência do mesmo ou porque o mesmo não estava disponível. Esta assimetria não existe quando estamos lidando com o *feedback* explícito dos usuários, quando os mesmos indicam claramente quais itens eles gostaram e quais não gostaram. Isto tem algumas implicações. Por exemplo, os sistemas que lidam com dados explícitos tendem a focar apenas na informação conhecida - os pares usuário-item para os quais conhecemos as avaliações. Assim, as relações usuário-item restantes, que geralmente constituem a maior parte dos dados, são tratadas como desconhecidas e omitidas da análise. No caso do processamento de dados implícitos, nos concentrar apenas nos dados conhecidos, o *feedback* positivo dos usuários, irá desfigurar a representação do perfil dos usuários. Isto significa que estamos interessados apenas em verificar se um usuário estaria interessado ou não em consumir um vídeo, e não se o mesmo gostou ou não do mesmo.

2. O *feedback* implícito é, por natureza, uma fonte ruidosa. Enquanto

rastreamos passivamente o comportamento dos usuários, podemos apenas supor quais são suas intensões e reais motivos de consumo. Por exemplo, podemos ver o comportamento de consumo para um indivíduo mas isso não necessariamente quer dizer que o mesmo gostou do conteúdo. Ele pode ter começado a consumir e trocado para uma outra aba do navegador ou mesmo saído de perto do ambiente de consumo. Ele pode também ter fechado o *player* de vídeo demonstrando total desinteresse pelo conteúdo que isso não será levado em consideração, dado que estamos extraindo este comportamento dos *logs* de acesso dos servidores de entrega de vídeos e que só temos a informação de quais arquivos de vídeo foram requisitados.

Separamos o conjunto de dados em dois arquivos, um contendo os dados de treino e outro os dados de teste contendo 5 visualizações de vídeo por usuário dentro do mesmo período de observação. Os conjuntos contendo os dados de treino e de teste são disjuntos. Um exemplo contendo algumas linhas do arquivo com os dados de treino é mostrado a seguir.

```
mac179045:auto_esporte brunofms$ tail -10 auto_esporte.2011.data
data e hora de acesso | identificador do vídeo | identificador do usuário
25/May/2011:23:31:46 -0300|1515732|100629313.946488314.1304379721.1306179211.1306369776.4
25/May/2011:23:16:22 -0300|1487030|100629313.452412031.1305484947.1305484947.1306349282.2
25/May/2011:23:41:19 -0300|1515726|156749805.1238557690.1306377303.1306377303.1306377303.1
25/May/2011:23:44:09 -0300|1515730|100629313.1215930022.1305570691.1306374138.1306376677.35
25/May/2011:23:49:05 -0300|1487028|156749805.1238557690.1306377303.1306377303.1306377303.1
25/May/2011:23:54:18 -0300|1486234|156749805.1428959616.1306374823.1306374823.1306374823.1
25/May/2011:23:54:51 -0300|1515726|100629313.1035999351.1300656444.1306361381.1306375927.20
25/May/2011:23:54:45 -0300|1481372|156749805.1238557690.1306377303.1306377303.1306377303.1
25/May/2011:23:56:31 -0300|1481373|156749805.1238557690.1306377303.1306377303.1306377303.1
25/May/2011:23:57:03 -0300|1515726|156749805.1242565478.1306182012.1306209390.1306292155.3
```

Tabela 4.2 - Formatação dos dados de entrada

Nosso objetivo é encontrar um vetor $p_i \in R^f$ para cada usuário *i*, e um vetor $q_j \in R^f$ para cada vídeo *j* que irá caracterizar a intenção dos usuários a consumir vídeos. Em outras palavras, as preferências são denotadas como o produto interno $\hat{e}(u_i, v_j) = p_i^T q_j$. Estes vetores visam mapear usuários e vídeos num espaço de variáveis latentes onde podem ser diretamente comparados de forma a prover as recomendações.

Nesses experimentos, separamos nosso conjunto de dados em três sub-conjuntos: o de treino com os dados de visualização usados para treinar os modelos de predição, o de validação para validar os parâmetros usados nos algoritmos de aprendizado destes modelos e um de teste, como usuários e vídeos sem registros de visualizações nos dois anteriores, usado para verificar e comparar a precisão destes modelos. Resumidamente, temos:

- dados de treino (53.167 visualizações)
- dados de validação (12.184 visualizações)
- dados de teste (12.184 visualizações), usado na comparação dos modelos

4.2 Métrica de Avaliação

A qualidade de um sistema de recomendação pode ser decidida a partir dos resultados das avaliações. O tipo de métrica utilizada depende do tipo de aplicação de filtragem colaborativa. De acordo com [61], as métricas de avaliação podem ser classificadas em diversas categorias. Para examinar a performance dos experimentos aqui propostos, focamos na categoria de precisão de predição e adotamos a raiz do erro médio quadrático (*RMSE*):

$$RMSE = \sqrt{\frac{1}{n}\sum_{\{i,j\}}(\hat{e}(u_i, v_j) - r_{ij})^2}$$

onde n é o número total de observações, $\hat{e}(u_i, v_j)$ é a predição de intenção do usuário i em visualizar o vídeo j, e r_{ij} é a observação de fato. O *RMSE* amplifica a contribuição dos erros absolutos entre as predições e as observações de consumo.

Além de ser bem conhecido e um número simples de se calcular, o *RMSE* tem uma outra propriedade vantajosa que é o fato dele amplificar erros gerados por falso positivos ou falso negativos. E qualquer sistema de recomendação deve saber lidar bem com isso. No entanto, a precisão da predição não resolve diversos outros aspectos importantes de se considerar ao fazer, ou receber, uma recomendação [68].

4.3 Experimentos

O método aqui proposto possui diversos parâmetros a serem ajustados. Dentre estes parâmetros, experimentamos diferentes números de variáveis latentes, diferentes taxas de aprendizado para as variáveis latentes e as constantes de vieses, diferentes parâmetros de regularização. Também testamos diferentes modelos de predição.

Visando o desenvolvimento de um protótipo rápido para os testes, a maior parte do código foi escrita em *Python*, usando a biblioteca *numpy* para lidar com objetos tipo *array* muito grandes e para os cálculos mais pesados, dentre outras funcionalidades.

Inicializamos todos os valores da matrizes de preferências P e de características Q randomicamente seguindo um modelo de distribuição normal. Uma variável latente é treinada até que a melhoria no *RMSE* seja inferior a 0.0001, ou que um número mínimo de 20 passos de treino e máximo de 50 seja feito no conjunto de dados.

Quantidade de Passos de Treino

Neste experimento, visamos encontrar uma quantidade de passos de treino razoável para o conjunto de dados em questão. Usamos uma taxa de aprendizado de 0.001 e 20 variáveis latentes. Vale notar que a partir do vigésimo quarto passo passamos a ter uma condição de *overfitting*.

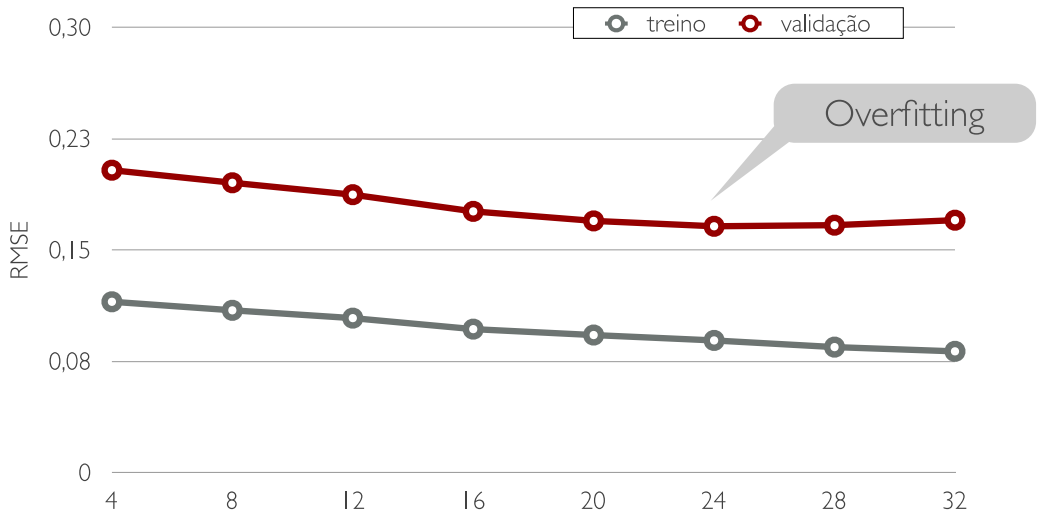

Figura 4.2 - Quantidade de passos de treino vs. *RMSE*

	16 passos	20 passos	24 passos	28 passos	32 passos
Treino	0.0965	0.0926	0.0889	0.0844	0.0816
Validação	0.1759	0.1695	0.1658	0.1666	0.1700

Tabela 4.3 - Quantidade de passos de treino vs. *RMSE*

Número de Variáveis Latentes

A escolha da quantidade de variáveis latentes ótima *f* é crítica para uma boa predição das intenções de visualização. Estamos interessados em um valor de *f* que seja grande o bastante para capturar todas as características das interações presentes na matriz e pequena para comportar um tempo de construção do modelo que seja razoável. Encontramos um valor razoável para *f* através de sucessivas tentativas com diferentes valores experimentais.

Para este experimento, ajustamos o número mínimo de passos de treino em 20 e o máximo em 50, e usamos uma taxa de aprendizado de 0.001.

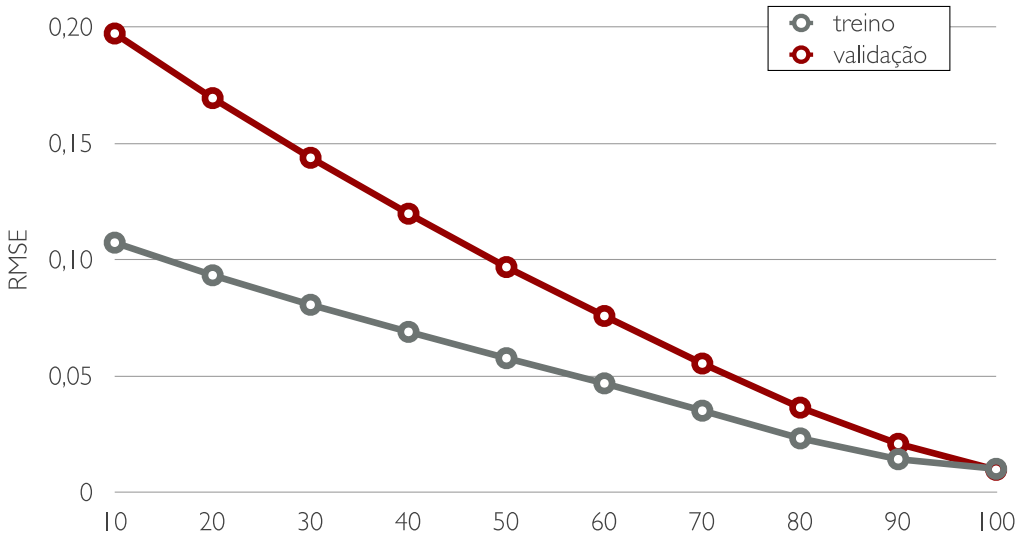

Figura 4.3 - Quantidade de variáveis latentes vs. *RMSE*

	10 variáveis	30 variáveis	50 variáveis	70 variáveis	90 variáveis
Treino	0.1072	0.0805	0.0576	0.0350	0.0142
Teste	0.1971	0.1437	0.0967	0.0552	0.0208

Tabela 4.4 - Quantidade de variáveis latentes vs. *RMSE*

Pegando duas das variáveis latentes resultantes do processo de fatoração matricial, vemos na Figura 4.4 que os vídeos ficam localizado no lugar apropriado com base nas duas dimensões destes vetores latentes. Estes lugares no

gráfico revelam o conteúdo dos filmes que por ali se localizam.

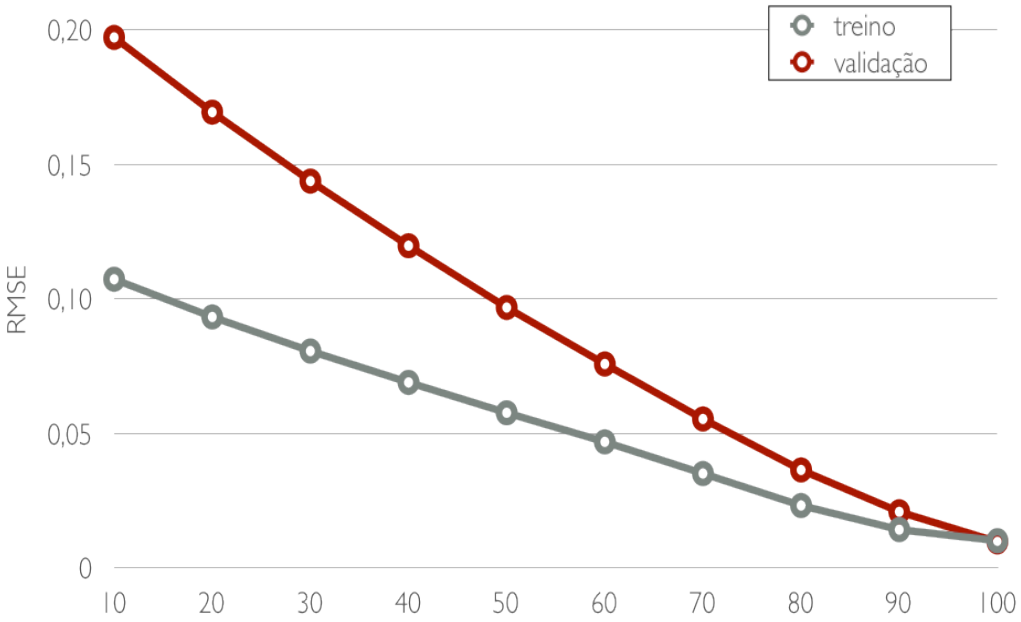

Figura 4.4 - Variável latente 1 vs. variável latente 2

Neste exemplo, podemos identificar claramente 2 categorias: vídeos referentes a carros classificados como raridades ou de luxo, e vídeos relacionados a mecânica de automóveis com informações e tutorias.

Taxa de Aprendizado

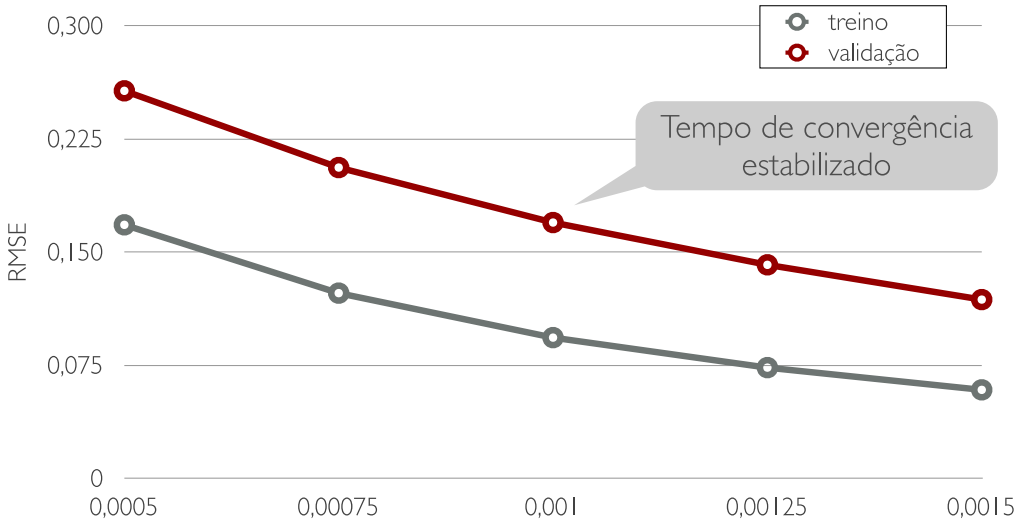

Figura 4.5 - Efeito da taxa de aprendizado

	L = 0.0005	L = 0.00075	L = 0.001	L = 0.00125	L = 0.0015
Treino	0.1679	0.1226	0.0932	0.0733	0.0585
Validação	0.2567	0.2059	0.1694	0.1414	0.1184

Tabela 4.5 - Efeito da taxa de aprendizado

Neste experimento, variamos a taxa de aprendizado com valores em torno de 0,001. Todas as observações foram feitas usando 20 variáveis latentes e um fator de regularização com valor fixo em 0,02.

A Figura 4.5 mostra os valores de *RMSE* para os dados de treino e validação. Quanto menor a taxa de aprendizado, maior o tempo de convergência durante o treino. No entanto, aumentar muito a taxa de aprendizado, pode nos levar rapidamente a uma condição de *overfitting*.

Fator de Regularização

Neste experimento, variamos a taxa de aprendizado com valores entre 0,015 e 0,025. Todas as observações foram feitas usando 20 variáveis latentes. A taxa de aprendizagem utilizada foi de 0,001. O gráfico a seguir mostra os resultados.

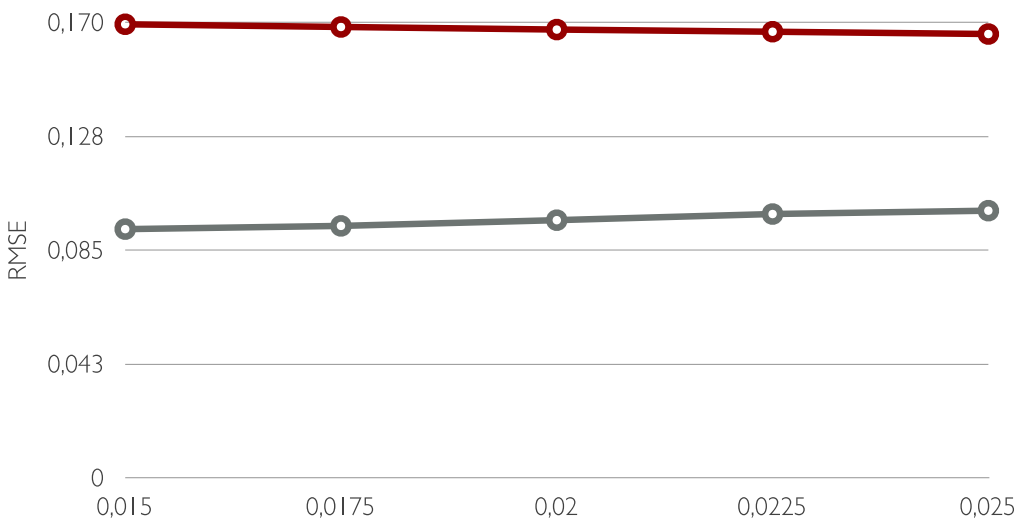

Figura 4.6 - Efeito da regularização

	R = 0.015	R = 0.0175	R = 0.02	R = 0.0225	R = 0.025
Treino	0.0928	0.0940	0.0961	0.0984	0.0997
Validação	0.1692	0.1682	0.1673	0.1665	0.1656

Tabela 4.6 - Efeito da regularização

Os resultados mostram claramente que baixos valores para o parâmetro de regularização resultam, mais uma vez, numa condição de *overfitting* nos dados de treino e que o *RMSE* diminui a medida que aumentamos seu valor.

Comparação dos Modelos

Tentamos diferentes implementações e parametrizações de fatoração matricial. A Figura 4.7 mostra como diferentes modelos e quantidades de variáveis latentes, assim como a evolução dos modelos - fatoração pura, adição de vieses dos vídeos e usuários, adição dos efeitos temporais - afetam o *RMSE*.

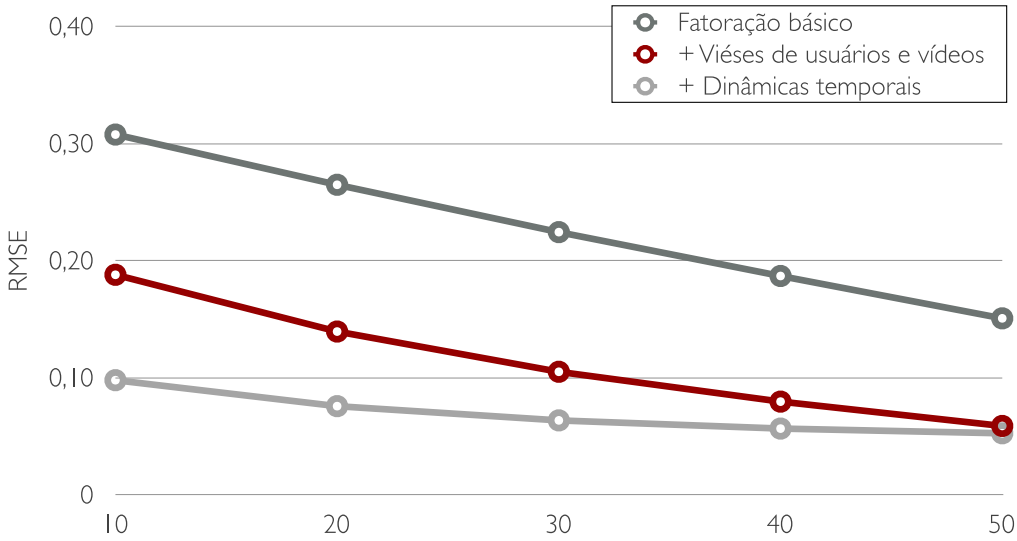

Figura 4.7 - Comparação dos modelos de fatoração

	10 variáveis	20 variáveis	30 variáveis	40 variáveis	50 variáveis
fatoração simples	0.3076	0.2647	0.2242	0.1868	0.1506
+ viés	0.1878	0.1393	0.1049	0.0794	0.0586
+ efeitos temporais	0.0978	0.0756	0.0634	0.0564	0.0524

Tabela 4.7 - Comparação dos modelos de fatoração

Um dos principais desafios destes experimentos foi encontrar uma combinação de parâmetros que desempenhasse bem em cada caso. Esta tarefa nos consumiu grande parte do tempo do projeto.

Para a construção dos nossos preditores ficamos com modelos de fatoração matricial incremental com regularização com uma taxa de aprendizado

de 0,001 e fator de regularização em 0,02. Na inclusão dos vieses dos itens (vídeos) e usuários, usamos uma taxa de aprendizado de 0,0001 com fator de regularização de 0,022 nas equações de atualização. Na adição das dinâmicas temporais dos itens, usamos uma taxa de aprendizado de 0,00003 com fator de regularização em 0,02 nas equações de atualização.

Dentro do período de tempo no qual os registros foram capturados, extraímos 22 fragmentos temporais, ou seja, 1 fragmento para cada semana. Isso, para poder entender o impacto que as dinâmicas temporais desses acessos têm no resultado final das predições.

Os resultados da Figura 4.7 mostram a evolução do *RMSE* para cada um dos modelos individualmente. Vemos que a precisão melhora à medida em que aumentamos a dimensão (denotada pela quantidade de variáveis latentes) do modelo. Além disso, pode-se observar que, quanto mais refinado é o modelo, com entradas de dados oriundas de outras observações, melhor o resultado.

4.4 Recomendando Vídeos

Para recomendar novos vídeos para os usuários, inicialmente usamos os modelos apresentados para capturar as relações latentes entre os usuários e os vídeos e então computar a predição de intenção de um usuário assistir a um dado vídeo. Em seguida, usamos estas predições para gerar uma lista com os *N* vídeos a serem recomendados para os usuários.

Começamos com a matriz de avaliações *R*, a qual é bastante esparsa. Para capturar os relacionamentos latentes removemos esta esparsidade extrapolando as avaliações dadas pelos usuários para alguns itens para o resto da matriz. Para isso, decompomos a matriz *R* nas matrizes *P* e *Q*, indicando o peso das interações usuário-vídeo na predição das intenções, nos vetores b_i e b_j com os vieses de usuários e itens, respectivamente e nas matrizes *M* e *D*, com o peso dos efeitos temporais nas intenções de consumo e usamos estes componentes resultantes para computar a predição de intenção de qualquer usuário *i* ver um vídeo *j*:

$$r'_{ij}(t) = \sum_{k=1}^{f} p_{ik} + q_{kj} + b_i + b_j + b_{j,Bin(t)}$$

Para recomendar vídeos para um dado usuário *i*, ordenamos as predições de intenção de consumo de vídeos para aquele usuário, isto é, os elementos da linha *i* da matriz *R*. Os vídeos cujas predições de intenção de visualização estão mais próximas de 1 e que ainda não tiverem sido consumidos pelo usuário em questão serão os recomendados.

```
User 100629313.2038689089.1296840090.1299264125.1299717968.27:
(0.0002619, 'Veja as diferenças dos barulhos de cada motor', '2011-01-30')
(0.0002840, 'Objetos soltos no carro podem causar graves acidentes', '2010-12-19')
(0.0004117, 'Chave, cartão ou controle remoto?', '2009-01-18')
(0.0005802, 'Veja o que levar no kit de emergência para o seu carro', '2010-10-31')
(0.0007359, 'Rodas e aros', '2009-07-12')
```

Tabela 4.8 - Exemplo de recomendações para um usuário