Session-Based Recommendations with Recurrent Neural Networks

Hidasi, Karatzoglou, Baltrunas, Tikk

Patricio Cerda - Bastian Mavrakis

IIC3633 - Sistemas Recomendadores
2018-2

Acerca del paper

- Autores:
  - Hidasi, Tikk: Gravity R&D, Hungría
  - Baltrunas: Netflix
  - Karatzoglou: Telefónica Research

- Paper presentado en ICLR 2016

- Trabajo follow-up presentado en RecSys ‘16

Sesiones

Wikipedia

"Temporary, interactive information interchange between user and a computer system"

e-commerce

plataformas de consumo (como Netflix)

clicks que realiza el usuario para navegar a ciertos ítems de su interés

Información latente: muy útil para caracterizar el perfil del usuario

Aprovechar esto lleva a mejores recomendaciones

e.g.: ítems vistos hace tiempo son claves para recomendaciones en el futuro

Trabajo relacionado: session-based recommendation

- Item-to-Item: simple y efectivo
  - Matriz de similaridad: co-ocurrencia de ítems en base a clicks
  - Usa solamente el último click como *prior*

- Markov Decision Process: < estados, acciones, recompensas, transiciones >
  - Acciones = Recomendaciones
  - Se determinan según la transición más probable
  - Espacio de estados inmanejable si se incluye todas las posibles sesiones

- General Factorization Framework *(Hidasi, Tikk)*:
  - Usa info. pasada, pero no considera el orden de la secuencia

Deep Learning

Auge reciente de la técnica:

- Cantidad de datos disponibles + alta capacidad de cómputo ⟹ buen funcionamiento

- Desde el 2012 (*Krizhevsky et al*): resultados impresionantes ⟹ cambio paradigma

- Arquitectura para trabajar sobre datos secuenciales: Recurrent Neural Networks (*Hochreiter, Schmidhuber*)

Trabajo relacionado: Deep Learning en RecSys

- **Restricted Boltzmann Machines** (*Salakhutdinov et al.*): de los mejores modelos de collaborative filtering. Modela interacción usuario-item.

- **Convolutional Neural Networks** (*Van den Oord et al, Wang et al.*): modelos profundos obtienen features desde datos multimodales, para luego utilizarlas en modelos tradicionales.

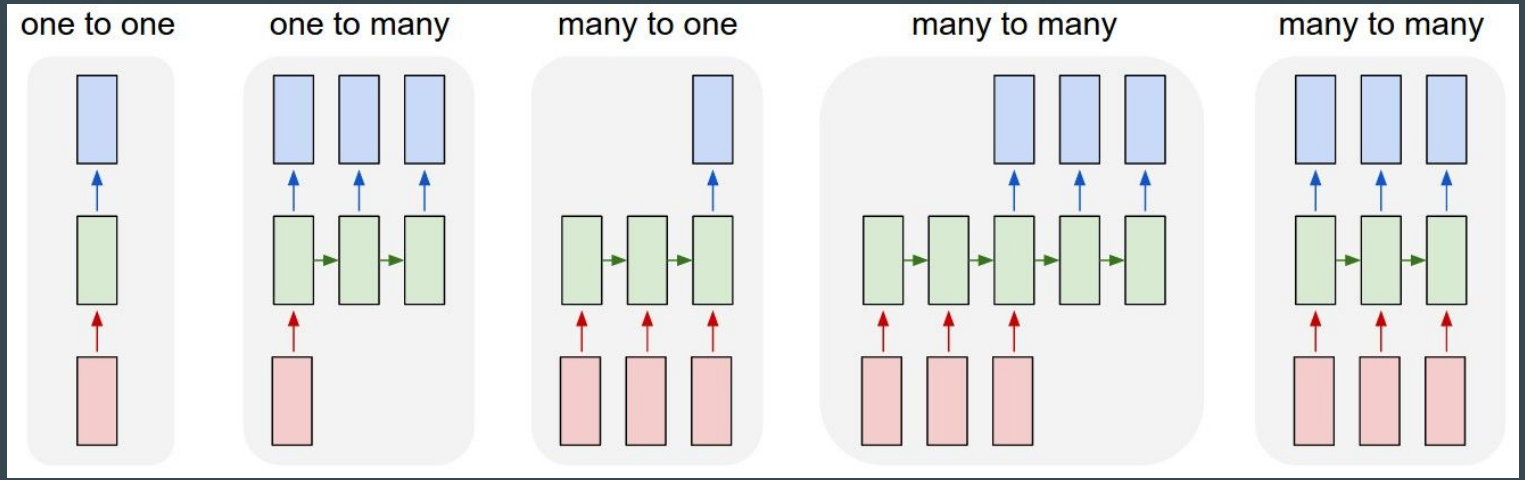
Contribución del trabajo

- Aplicar RNNs para recomendar en base a sesiones

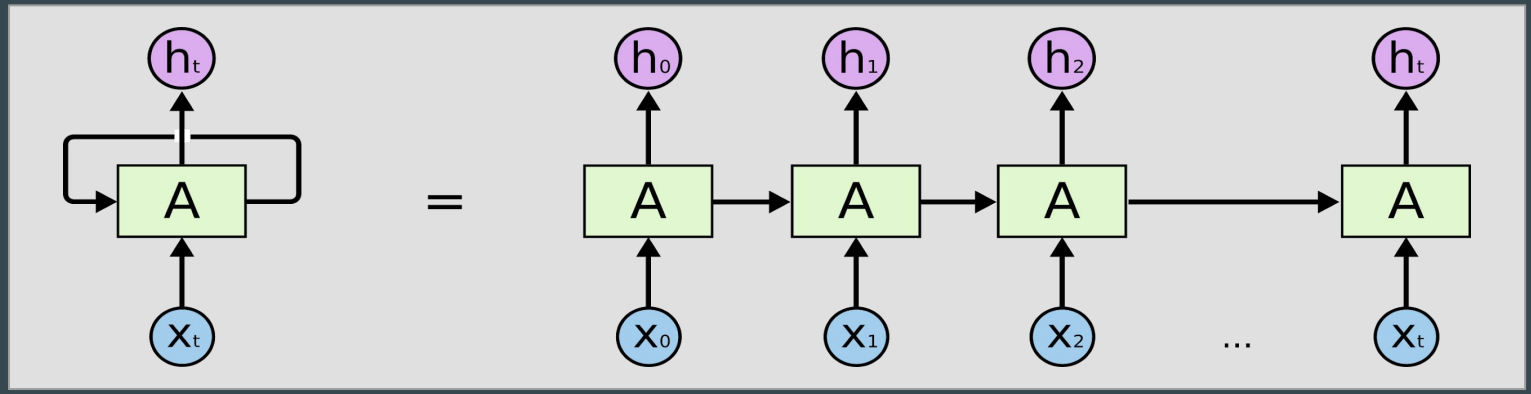
- Adaptar arquitectura RNN para lograr esto:
  - Nuevo método de **batch training: sesiones en paralelo**
  - Nueva **función de pérdida por ranking**, con sampling optimizado

- Resultados muestran desempeño superior a técnicas estándar

Recurrent Neural Networks

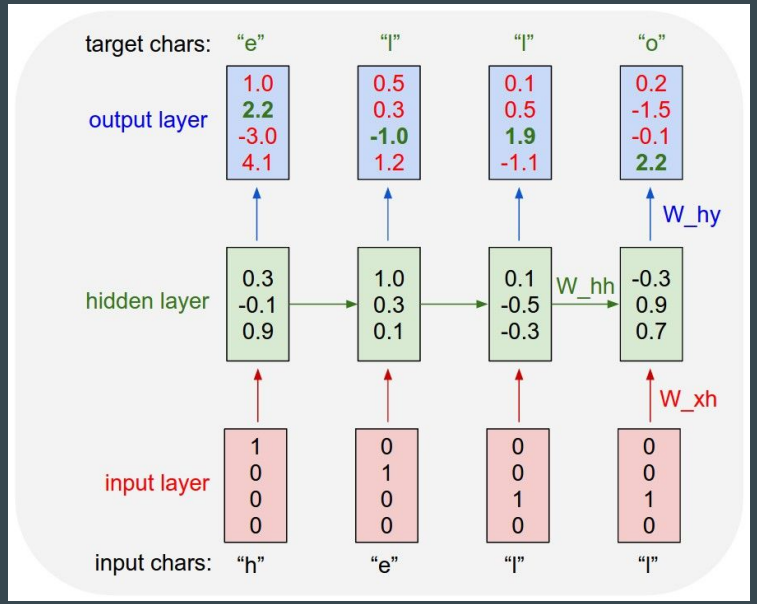


Recurrent Neural Networks



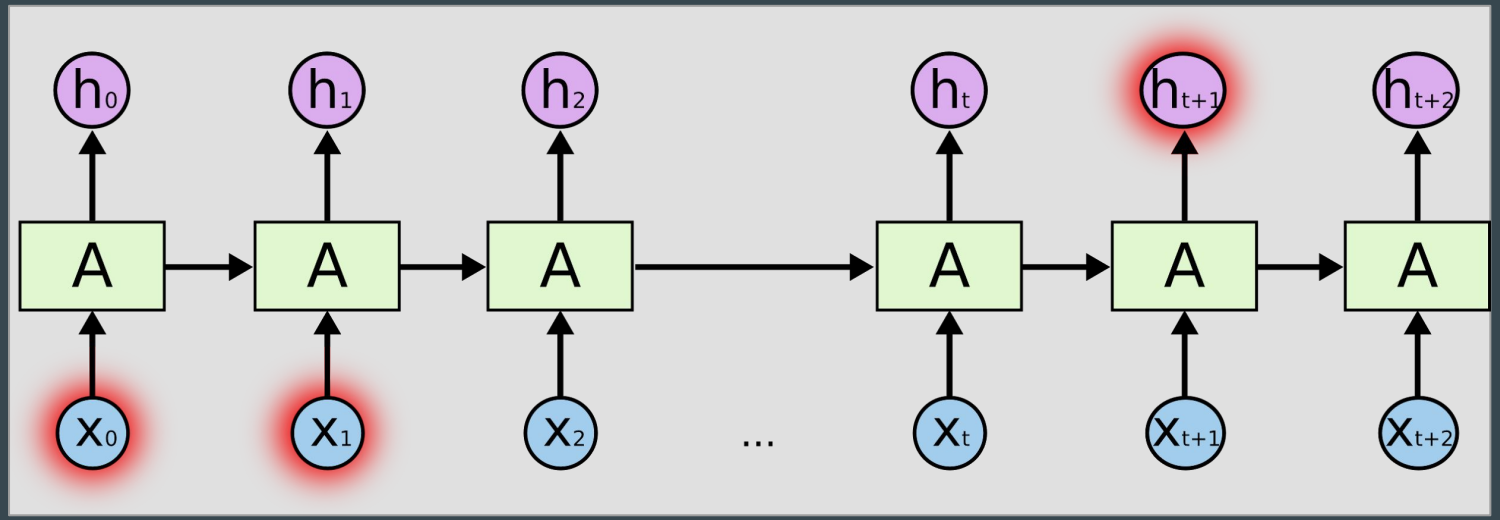
$$\mathbf{h_t} = g(W\mathbf{x_t} + U\mathbf{h_{t-1}})$$

Recurrent Neural Networks



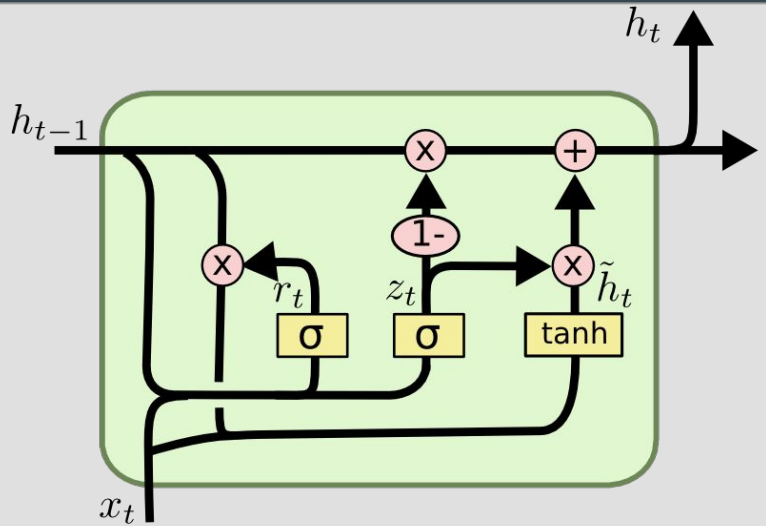
El problema de la dependencia temporal larga

Falla por exploding o vanishing gradient, aún cuando en la teoría es posible que la red aprenda esta dependencia.



Gated Recurrent Units

Permite recordar y olvidar, selectivamente, por intervalos indeterminados de tiempo.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

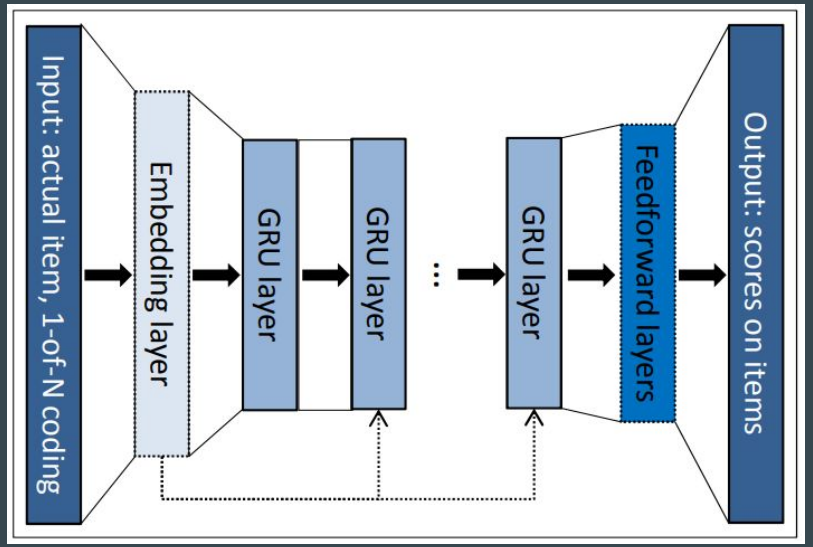
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Modelo RNN propuesto

Input al modelo: estado de la sesión, con one-hot encoding

Se probó un encoding que incluya todos los eventos hasta ese punto, pero se desempeñaba peor. Se especula que basta con el estado oculto de la RNN para llevar cuenta de la info. agregada de la sesión

La predicción del click en t+1 es la recomendación

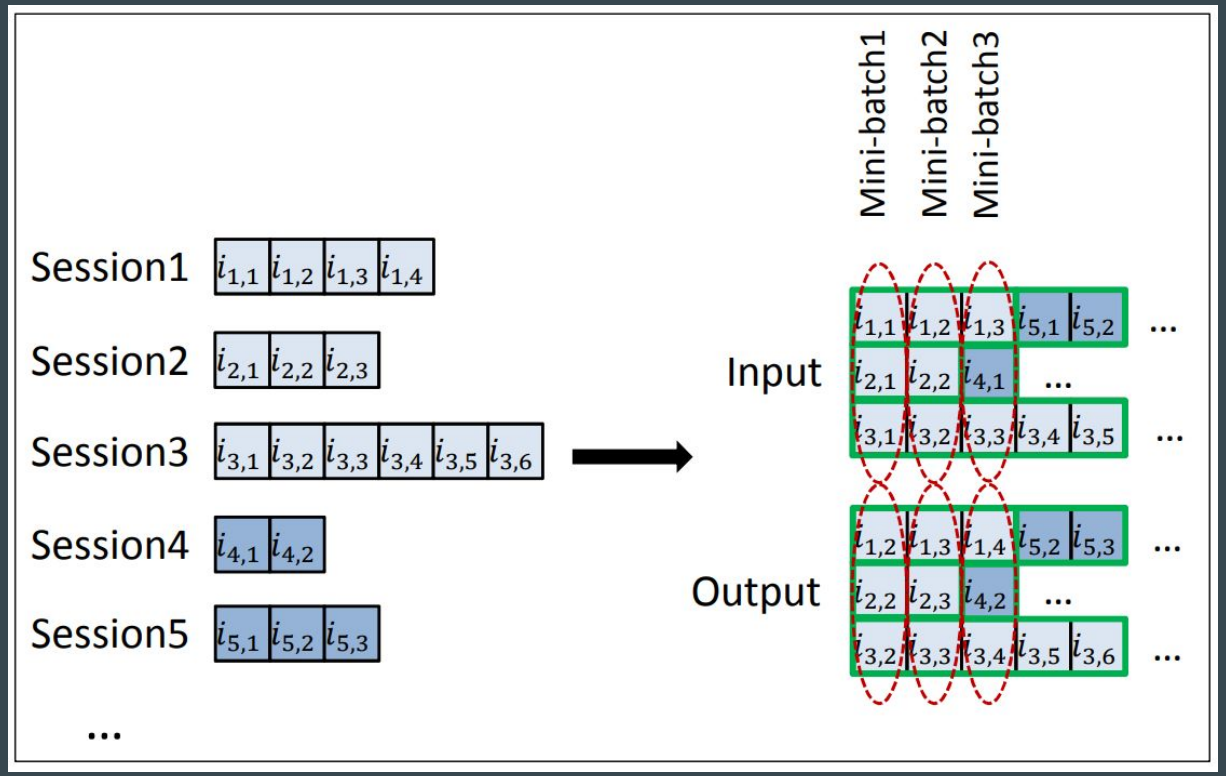


Modificaciones

Una RNN no está pensada para recomendar. Cambios para que el modelo funcione:

1. **Mini-batches de sesiones en paralelo**
    a. Se crea orden para las sesiones
    b. Para las n primeras sesiones, se toma sus primeros eventos como datos de training, y el segundo evento como la predicción correcta
    c. Se repite este paso con el siguiente grupo de eventos
    d. Si una sesión termina, se reemplaza en el mini-batch por la siguiente
    e. Se asume independencia. Si ocurre d), se resetea el estado oculto de la RNN.
2. Popularity sampling
3. Ranking Loss

Mini-batches de sesiones en paralelo



Modificaciones

Una RNN no está pensada para recomendar. Cambios para que el modelo funcione:

1. Mini-batches de sesiones en paralelo
2. **Popularity sampling:** Entrenamiento más eficiente
    a. Necesario para el cálculo de la loss function
    b. Sacar ejemplos negativos desde el resto del mini-batch
    c. Se omite etapa dedicada de sampling
    d. Probabilidad de aparecer es proporcional a la popularidad del ítem
3. Ranking Loss

Modificaciones

Una RNN no está pensada para recomendar. Cambios para que el modelo funcione:

1. Mini-batches de sesiones en paralelo
2. Popularity sampling
3. **Ranking Loss:**
   a. Funciones pointwise: inestables en la práctica
   b. Pairwise: BPR y TOP1

$$L_s = \frac{1}{N_S} \cdot \sum_{j=1}^{N_S} \sigma\left(\hat{r}_{s,j} - \hat{r}_{s,i}\right) + \sigma\left(\hat{r}_{s,j}^2\right)$$

Función de pérdida TOP1

L_s : pérdida para la sesión s

N_S : sample size sesion s

$\hat{r}_{s,j}$: score sesión s sobre ítem negativo j

$\hat{r}_{s,i}$: score sesión s sobre ítem relevante i

Experimentos

Se utilizaron 2 datasets:

1. RecSys Challenge 2015: Secuencia de clicks en un sitio de e-commerce. 7 millones de sesiones y 31 millones de eventos
2. OTT Video Service Platform: Secuencia de vistas a videos de la plataforma. 3 millones de sesiones y 13 millones de eventos

Experimentos

Métricas utilizadas:

1. Recall@20: Proporción de casos en donde el ítem deseado se encuentra dentro de los primeros 20 elementos recomendados.
2. MRR@20: Promedio de rangos recíprocos de los ítems deseados. El rango recíproco es 0 si un ítem no se encuentra entre los 20 primeros

Experimentos

Definición de baselines:

- POP
- S-POP
- Item-KNN
- BPR-MF

Table 1: Recall@20 and MRR@20 using the baseline methods

Baseline	RSC15		VIDEO	
	Recall@20	MRR@20	Recall@20	MRR@20
POP	0.0050	0.0012	0.0499	0.0117
S-POP	0.2672	0.1775	0.1301	0.0863
Item-KNN	0.5065	0.2048	0.5508	0.3381
BPR-MF	0.2574	0.0618	0.0692	0.0374

Experimentos: Parámetros y Optimización

- Hiperparámetros: 100 experimentos aleatorios
  - La mejor combinación fue luego optimizada a nivel de cada parámetro
- Número de hidden units: 100 y 1000
- GRU > LSTM y vanilla
- Matrices inicializadas random uniforme
- Optimizador: Adagrad
- Arquitectura: una capa de GRUs, sin feedforward

Table 2: Best parametrizations for datasets/loss functions

Dataset	Loss	Mini-batch	Dropout	Learning rate	Momentum
RSC15	TOP1	50	0.5	0.01	0
RSC15	BPR	50	0.2	0.05	0.2
RSC15	Cross-entropy	500	0	0.01	0
VIDEO	TOP1	50	0.4	0.05	0
VIDEO	BPR	50	0.3	0.1	0
VIDEO	Cross-entropy	200	0.1	0.05	0.3

Resultados

- El modelo le gana ampliamente a la mejor baseline
- Valida RNNs como herramienta para recomendar en sesiones
- Implementación en Theano, código en GitHub
- Mejor función de pérdida: TOP1, luego BPR

Table 3: Recall@20 and MRR@20 for different types of a single layer of GRU, compared to the best baseline (item-KNN). Best results per dataset are highlighted.

Loss / #Units	RSC15		VIDEO	
	Recall@20	MRR@20	Recall@20	MRR@20
TOP1 100	0.5853 (+15.55%)	0.2305 (+12.58%)	0.6141 (+11.50%)	0.3511 (+3.84%)
BPR 100	0.6069 (+19.82%)	0.2407 (+17.54%)	0.5999 (+8.92%)	0.3260 (-3.56%)
Cross-entropy 100	0.6074 (+19.91%)	0.2430 (+18.65%)	0.6372 (+15.69%)	0.3720 (+10.04%)
TOP1 1000	0.6206 (+22.53%)	**0.2693 (+31.49%)**	**0.6624 (+20.27%)**	**0.3891 (+15.08%)**
BPR 1000	**0.6322 (+24.82%)**	0.2467 (+20.47%)	0.6311 (+14.58%)	0.3136 (-7.23%)
Cross-entropy 1000	0.5777 (+14.06%)	0.2153 (+5.16%)	–	–

Referencias

1. Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, Domonkos Tikk: **Session-based recommendations with recurrent neural networks.** ICLR, 2016
2. Linden, G., Smith, B., and York, J. Amazon. com recommendations: **Item-to-item collaborative filtering**. Internet Computing, IEEE, 7(1):76–80, 2003
3. Shani, Guy, Brafman, Ronen I, and Heckerman, David. **An mdp-based recommender system**. In Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence, pp. 453–460, 2002.
4. Hidasi, Balazs and Tikk, Domonkos. **General factorization framework for context-aware recommendations.** Data Mining and Knowledge Discovery, pp. 1–30, 2015. ISSN 1384-5810. doi: 10.1007/ s10618-015-0417-y
5. Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. **BPR: Bayesian personalized ranking from implicit feedback.** In UAI'09: 25th Conf. on Uncertainty in Artificial Intelligence, pp. 452–461, 2009. ISBN 978-0-9749039-5-8
6. Krizhevsky, A., Sutskever, I. & Hinton, G. **ImageNet classification with deep convolutional neural networks.** In Proc. Advances in Neural Information Processing Systems 25 1090–1098(2012).
7. Hochreiter, S. & Schmidhuber, J. **Long short-term memory.** Neural Comput. 9,1735–1780 (1997).
8. Salakhutdinov, Ruslan, Mnih, Andriy, and Hinton, Geoffrey. **Restricted boltzmann machines for collaborative filtering**. In Proceedings of the 24th international conference on Machine learning, pp. 791–798. ACM, 2007.
9. Van den Oord, Aaron, Dieleman, Sander, and Schrauwen, Benjamin. **Deep content-based music recommendation**. In Advances in Neural Information Processing Systems, 2013.
10. Wang, Hao, Wang, Naiyan, and Yeung, Dit-Yan. **Collaborative deep learning for recommender systems**. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, pp. 1235–1244, New York, NY, USA, 2015. ACM
11. Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua (2014). "**Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation**"
12. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
13. http://karpathy.github.io/2015/05/21/rnn-effectiveness/