Notes on Papers

Multi-Rate Deep Learning for Temporal Recommendation

MOTIVATION:

Try to capture both long-term (static) user preferences, and short term preferences.

IMPLEMENTATION:

They create a Multi-Rate TDSSM (Temporal Deep Semantic Structured Model) that uses FFNN to create item embedding (assumed static), and FFNN for user static features, in addition they use one or two RNN (w/LSTM) to model the users short term preferences. They suggest one model with a single RNN, and one model with two RNNs with different granularity (week and day). Also, pre-training is used.

Important point that user preferences change over time. E.g. during the world cup in football as opposed to after. BUT: the information about the users short term preferences are much more sparse, so hard to recommend only based on this. Therefore, it is good to supply with long term preferences.

RESULTS:

Both proposed models outperform the baseline models used for comparison. The model with two RNNs performed best.

The model creates an embedding of the user and the items and (***USER2VEC***). Cosine similarities is used to compare these vectors and decide if an item would be relevant for the user. They test how good the model is by asking it to rank a set containing the correct click-item and 9 random items.

Improved Recurrent Neural Networks for Session-based Recommendations

MOTIVATION:

Further improvements on RNNs for session-based recommendations. Builds on the work from Hidasi et al. (Session-based recommendations with recurrent neural networks).

IMPLEMENTATION:

Techniques used for improvement:

- Data augmentation via sequence preprocessing and embedding dropout.
- Model pre-training
- Distillation using privileged information to learn from small datasets.

Also propose a novel alternative model that reduces the time and space requirements for predictions by predicting item embeddings directly.

Uses the **RecSys 2015** dataset

Does not use the session parallel sequence-to-sequence training from the Hidasi et al.’s model. Processes each sequence separately, and are trained to predict the next item.

All prefixes of the original input sessions are treated as new training sequences. So the input session [23, 65, 39, 67] gives the following training sessions: ([23], V(65)), ([23, 65], V(39)), ([25, 65, 39], V(67)), where V(x) is the target value.

Dropout is also used in the form of randomly removing clicks from the sequences.

Pre-training is used. The most recent clicks are most relevant, older clicks might be out of date. First train on the entire dataset, then fine-tune the model on only the most recent clicks.

Instead of a 1-HOT output layer, uses an embedding output layer. Compare the output embedding with the embedding of the target item’s embedding. Less memory and calculations for prediction, but needs good embedding.

Evaluation of model size and batch prediction times are also done, since these are important metrics in a real recommender system

RESULTS:

Better to train on more recent data, instead of whole dataset. All their models performed better than the paper they are based on. M4 (worst performance) was faster than the others.

Next Basket Recommendation with Neural Networks

Next basket recommendation is the task of recommending new items to the user, based on items they have bought earlier (items in their shopping basket).

They use an embedding layer that concatenate the user preferences and the sequential item baskets available, this is sent through a hidden layer and out in a softmax output layer.

Embedded user representation vectors are created for all users, and similarly for all items, during training. Each basket is represented as the mean of all items in it. The embedding layer concatenates the user and basket representations to obtain a feature vector, this can be viewed as a representation of both the users general interest and local context.

Experiments on two **datasets**: *Tafeng* and *Beiren*, which are retail datasets.

They only use 1 or 2 of the last baskets for recommendation/testing, because most users only have a few baskets in the dataset. Weird, since they remove all users with less than 4 baskets. And the last basket of a user is used as the test data. So they could have used more baskets. Why use RNN if they are only inputing 1 basket?....

Their results shows that k=2 is better than k=1 on a sufficiently large dataset.

The papers mainly shows that NN can be a good fit for recommender systems. Some of the other papers go more into detail on how to create a better NN/RNN.

A Dynamic Recurrent Model for Next Basket Recommendation

They want to crate a model that captures both a dynamic representation of a user and global sequential features among baskets.

MOTIVATION: Matrix factorization is good collaborative filtering model, but does not utilize sequential actions. Different forms of Markov Chains uses sequential information, but only local (between two baskets e.g.). This is not good enough, a user might buy a laptop (first basket), some food (second) and accessories to the laptop (third), there is no relevance between two adjacent baskets here. Need a model which uses the full global sequence.

IMPLEMENTATION: RNN. Items in baskets are pooled into one basket vector. The basket vectors for a user are fed into the RNN, and out comes scores for all items.

Next Basket Recommendation with Neural Networks came out a year earlier, but the authors of this papers says that: “to the best of our knowledge, DREAM [their model] is the first approach that attempts to incorporate dynamic representation and global sequential behaviour for anhancing the performance of next basket recommendation.”

The latent representations of the items are pooled to represent the basket. Not clear how the latent representation of the item is found. (Seems like random vector representations are created for each item. Would it not be better to use 1-HOT or a more structured embedding? Maybe better performance with random vectors? Can easily choose size)

DATASETS: Ta-feng from RecSys and Tmall from Alibaba (was not able to access this site, but it is a much smaller dataset so probably not that interesting..).

Preprocessing: only keep items that were bought by at least k users (k=10 for ta-feng), and users that bought at least k items.

DTU Toolbox from NMF was used to implement the baseline models.

Max-Pooling was better than Avgerage-Pooling.

This paper does only use information about the buying sequences (baskets), no extra user embedding, and no fancy pre-training or other enhancements like they do in the other basket recommendation paper. So since they still achieve better results than other state-of-the-art models, it seems like RNN is a very good fit for sequential rec.sys. problems.

Their explanation for why max pooling is better: Max pooling is better than average pooling at representing the dependence between items.

Context-aware Sequential Recommendation

In this paper they present a novel (RNN) model that will be able to capture contextual/external information such as timer, location, weather, ...

RNN good at sequential prediction, but no sense of conetxt. Contextual models good at context but no sense of context. CA-RNN hopefully solves this.

DATASETS: Taobao and Movielens-1M. Both sets have timestamps. Taobao has rich contextual information. They removed users with less than 10 records and items with less than 3 records, to avoid data sparsity.

Based on timestamps they extract day of week, three ten-day periods in a month, and wheter it is a holiday or not (Taobao dataset). And in Movielens dataset they use day of week and hour (24) of the day.

Time intervals are discretized into days, more than 30 days are threated as the same bin.

RESULT: CA-RNN beats all other methods used for comparison. They also compared CA-RNN-input and CA-RNN-transition (only used input-context and transition-context respectively), and they beat eachother on one dataset each, implying that they are both important in different settings and both should be used.

Both contextual methods (FM, CARS2) and sequential methods (FPMC, HRM, RNN) were used for comparison. Sequential methods outperformed context methods, indicating that sequential information has more significant effects than contextual. RNN achieved the best performance among the compared methods, so it is probably good to use RNN in some forms for these tasks. CA-RNN also outperforms RNN on all dimensionalities (size of hidden layer (?)).

Session-based Recommendatios with Recurrent Neural Networks

Hidasi et al.

A ’plain’ RNN with GRU used to show that RNN can be better than MF and item-KNN for session rec.sys.

IMPLEMENTATION:

> Theano implementation can be found here: https://github.com/hidasib/GRU4Rec

Modelling Contextual Information in Session-Aware Recommender Systems with Neural Networks

Twardowski

Looks at using richer eventinformation from the session than just the item clicks. Item clicks are not very useful recommending on e.g. a news site. Also user interactions with the search enging contains very useful information that is often overlooked.

Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations

Hidasi et al

In the rec-sys session context it can be smart to utilize item features for the cold start problem, i.e. when a new item enters the pool.

They try to utilize alot of information about the item and event: images, textual description, itemID. They distribute the processing across multiple RNN networks, (one for image, one for itemID, ..), to let them specialize on different inputs.

They propose and evaluate 3 alternative optimizations to train their p-RNN to avoid different parts of the same network learning the same relations

They test their network on videorepresentation using the thumbnail and product recommendation using the text description. They compare against item-kNN.

They compare different architectures with 1-HOT and feature embedding inputs.

They also test different ways of training the networks where they only backpropogate through parts of the network in different fashions.

The parallell architecture worked best. ID only was not far behind when given enough hidden units. Parallell architectures (and RNN generally) had strongest improvement in MRR.