Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

# Master's Thesis Nr. 265

Systems Group, Department of Computer Science, ETH Zurich

in collaboration with

Digitec Galaxus AG

Applying the GAN Framework to Recommender Systems

by

Mohammed Ajil

Supervised by

Bojan Karlas, Ce Zhang

September 13, 2019

Informatik
Computer Science

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

# Applying the GAN Framework to Recommender Systems

Master Thesis

Mohammed Ajil

September 13, 2019

Advisors: Prof. Ce Zhang, Bojan Karlas

Department of Computer Science, ETH Zürich

**Abstract**

This example thesis briefly shows the main features of our thesis style, and how to use it for your purposes.

# Contents

Chapter 1

# Introduction

- Recommender System are a field in machine learning that get more and more attention
- In principle the goal of a recommender system is to recommend items (in the case of an e-commerce system products) to users
- The idea is that the system helps the user navigate the mass of products available and to show the user what is relevant
- A recomender system can be tuned to optimize for different metrics such as click through rate, conversion rate etc.
- In the past there have been mainly to approaches to recommender systems user based and item based
- In the last years there have been more and more proposals for session based approaches (refer to paper that does the survey)
- Specifically GRU4Rec has gained attention as a RNN based approach to solving this problem
- In a session based setting we try to model the sequence of events a user makes when he browses the product catalog instead of items or users themselves.
- This has the following advantages: Usually we have a lot of users that only visit the site once or twice a year, these users are very difficult to model. The same goes for products, there are a lot of products that have limited attention from the userbase, therefore it is difficult to get a useful representation for products like this.
- In a session based setting we model the sequence of events, which is more useful since there is a lot more data, and sessions are comparable even if we do not know which user is online.

- Session based approaches work in both settings where we dont know the user and where we know the user
- Item based are for unkown users and user based for known users
- The goal of the thesis is twofold:
  - First we want to explore the capabilities of such a system in a production setting with real world data. Are we really better than simple approaches like "often bought together"?
  - Also we want to compare it to a more sophisticated system that is a blackbox and consumed as a service provided by Google.
  - Third, since this is an RNN approach we want to find out if we can improve the system by applying the GAN framework in form of professor forcing
  - Fourth, the system inherently produces embeddings for products and users, we want to find out if we can improve the overall performance if we use a pretrained product embedding which captures the semantic similarity of products

Chapter 2

# Background

## 2.1 Recommendation Systems

### 2.1.1 Problem Statement

Generally speaking recommendation systems are concerned with recommending items to be of use to users. The recommendations should help the users decide which items to buy, music to listen to, what news articles to read etc. Virtually any decision-process can be made easier for users by providing recommendations. Typically recommendation systems are used by online services, famously for example by Spotify for music [1] and YouTube for videos [2]. By using these online service the users generate data that the service can use to improve the recommendations, for example rating videos gives the operator of the service a datapoint on how much a video is liked by a specific user. However also retailers are known to use recommendation systems, based on data generated by customer loyalty programs such as Migros Cumulus [5] retailers tailor coupons or other offers to their customers. In principle data of users interacting with items (viewing, buying, rating etc.) serves as the basis for a recommendation system. Depending on the use-case this data is utilized to assign scores to items depending on the user. The semantic meaning of a score depends on the objective of a recommendation system. For example a system which recommends products to be bought might assign a "probability of purchase", whereas a system which recommends videos might predict the probability of a user watching a video to the end.
Bringing this all together we can define a general recommendation system with the following function:

$$s = f(i, u, h_u)$$

Where $s$ is the assigned score to item $i$ for the user $u$ and the users history $u_h$. The users history represents all the previous interactions with items.

Essentially when we design a recommendation engine we want to learn the function $f$. To assess the quality of the learned function we need to know the "true" score for the specific item and user, this can be done in different ways. Usually during training we will hold off later interactions with items, and test the learned functions on those. However as soon as the users sees recommendations, we essentially change the reality, i.e. we don't know what the user saw as recommendations, if any, in the user history, therefore it makes sense to also assess the performance of a recommendation engine in production.

### 2.1.2 Properties of Recommendation Systems

In the following we will look at different properties of recommendation systems. Typically recommendation systems have a combination of the following properties.

**User-based** User-based recommendation system base the information used to make recommendations mainly on properties of the user, such as gender, age, etc. Also information from the history of the user can be used, for example what the user has bought or read before. Essentially that means when we try to generate recommendations for a specific user we try to find similar users and recommend items that the similar users liked.

**Item-based** Item-based recommendation system use mostly information about the item to produce recommendations, such as item type (e.g. genre of a movie). However as the name suggests, the basis for a recommendation is always a specific item, for example a product a user is looking at online. This item the user is currently interacting with is referred to as the "active item". Usually item-based recommendation systems then try to find similar items to the currently active item. The method of finding similar items can be arbitrarily complex, identifying similar items is its own field of research. What is also often done is combine this approach with a user-based component, where the similar items are sorted or filtered based on the user. An example of this might be the following:

- Steve is a male looking at black shirts.
- When extracting similar products we find a range of black shirts, also containing womens shirts.
- If we would recommend items by popularity the womens shirts would appear as the first recommendation, since women typically buy more shirts.
- Instead of directly displaying the recommendations we filter out the womens shirts that is found in this selection.

**Session-based** Session-based recommendation systems are a rather new form of such a system. A session is a sequence of interactions from a user with one or several items. Depending on the use-case and setting this can be defined differently. Usually online-services define a session as the sequence of interactions a user produces on the site until closing the browser window. Obviously sessions can have variable lengths, therefore it is difficult to directly feed that information into a recommendation system. However with the rise of Recurrent Neural Networks (c.f. 2.2.1) a powerful tool for handling variable length sequence data becomes available. Session-based recommendation systems use RNNs to model the sequence data generated by sessions to achieve two things. First by using RNNs we can extract a fixed dimensional representation of a specific session, this allows us to compare different sessions. Second by using the fixed representation of a session we can try to identify the intent a user has in a specific session, based on this recommendations can be made to fulfill the users intent. An intent can be defined as the goal the user has in a specific session. In the example of an online-shop there are a few different, well-known intents identified by analysts such as: Browsing for inspiration, searching for a specific product, buying a specific product, researching products etc. Also these intents exist in different contexts, in the case of an online-shop the contexts can be different product types (mobile phone, couch, dining table) etc. From the above explanation it is intuitive to see why these systems are more desired by operators of online services, since the recommendations can be targeted much more specific to the user and his intent, instead of just general information of the user and the active item.

**Collaborative** Collaborative recommendation-systems mostly use interaction data to generate recommendations. For example we use the clicks on products as a data source and then predict which products the user will click next. However the collaborative aspect comes from the fact that we source other users interactions as a basis for the recommendations. In principle we view different users as versions of possible behaviour of a user, the more interactions two users have in common, the more similar they are assumed to be. Therefore we can extend the behaviour (i.e. product views) of a user by looking at what similar users have done on the same active item.

**Content-based** In contrast to collaborative recommendation systems, content-based systems heavily rely on so called content information. Content information refers to the actual content of different items. The name stems from early recommendation systems which mainly focussed on recommending media. The idea is to actually analyze the content of an item, be it the images in a movie, the soundwaves of a song or the text in a book. The assumption is that a user likes to see items that are similar to each other (e.g. a user who mostly likes action films). However this method can also be used in different contexts, such as online-shopping, where the ”content” of an item might be its textual description or an image of the actual product.

### 2.1.3 Well-Known Examples

- Describe a classical user based approach
- Describe a classical item based approach

## 2.2 Concepts

### 2.2.1 Recurrent Neural Networks

- Explain the concept of neural Networks
- What are the problems (vanishing gradients etc)
- Explain what a GRU is and why does it handle vanishing gradient better than simple RNN cells?
- How does a layered RNN work?

### 2.2.2 Generative Adversarial Network Framework

- Explain the way GANs work
- Loss function is learnable
- The discriminator is part of the loss function
- Especially good if we want generate stuff

### 2.2.3 Teacher Forcing

- RNN have two modes: Teacher forced and Free running
- When training we always use the ground truth as input
- This generates two different distributions of hidden states, which in turn generates two different distributions for the output
- This is a problem since this makes RNNs in production less predictable

## 2.3 Previous Work

### 2.3.1 Professor Forcing

- Here we want to mitigate the teacher forcing problem, by applying the gan framework
- Add graphic showing how it should work
- Describe the model as we did in the slides, show the loss function

[3]

### 2.3.2 Meta-Prod2Vec

- This is a model that allows us to capture the semantic similarity between products
- Explain how it works by relating to word2vec (refer to word2vec paper)

[6]

### 2.3.3 Hierarchical RNNs for personalized Recommendations

- This is the starting point for the thesis
- Which components does the system have?
- Show the graphic from the paper

[4]

## 2.4 KPIs

### 2.4.1 Click-Through-Rate

### 2.4.2 Conversion Rate

- Describe different KPIs
- What do they meaure, how to optimize for it

Chapter 3

# Dataset

## 3.1 Data Collection

### 3.1.1 Data Generation Mechanism

- Each click the user makes on the site is tracked
- An event is generated containing the information and context of the click of the user
- This event is sent to a message queue
- From the message queue there is a subscriber that archives this event in Google BigQuery
- The most important fields tracked are: UserId, SessionId, ProductId, Timestamp, LastLoggedInUserId
- Google BigQuery is a Data Warehousing solution consisting of append-only tables

### 3.1.2 Data Storage

- Data is stored in BigQuery in a denormalized form
- One row describes one event

## 3.2 Data Extraction

- First we extract the events on ProductDetail Pages from BigQuery
- This is done in shards, where each shard approximately corresponds to one day of events

- We only use the data where we know which user produced the event
- The events are filtered first by removing all events generated by known bot user agents
- In a second step we merge the fields UserId and LastLoggedInUserId, if the latter is set we know that the user ″passively″ logged out of the shop (long time without a session)
- Then the events are grouped by sessions
- Then the sessions are grouped by users
- After that we filter out users that have too few sessions
- Then we filter products that have too few events
- After that we have a dataset that contains all users with enough sessions and all events on products with enough events
- Show the structure of one user in the json

## 3.3 User Parallel Batches

- Show how the user parallel batches should look like
- Explain how we transform the json dataset to these user parallel batches

Chapter 4

# System Overview

## 4.1 Model Architecture

- Describe Model Architecture
- Describe different Components of Model Architecture
- Describe different variants of the model (with/without pf, with/without embeddings)

## 4.2 Product Embedding

- Describe the Architecture of MetaProd2Vec
- Write about the features used, what are the transformations etc.
- Analyze the embedding like we did when we first started the experiment (Which products are close/far away inside a producttype, similar brands etc)

## 4.3 Implementation

- Describe Class Diagram
- Describe prediction mode/training mode

## 4.4 Production Setup

- Describe the flow of a recommendation through the system for the different cases

- Probably a sequence diagram makes most sense
- I.e. show how SOFI etc. plays into this whole thing

Chapter 5

# Experiments

For each of the model variants evaluate the experiments offline and online.

## 5.1 Offline

### 5.1.1 Experiment Setup

- Describe the last session out split (test and eval)
- Describe how we train the model and how we evaluate it

### 5.1.2 Measurements

- Here we want to confirm a few things: user rnn helps, embeddings helps, profforcing helps
- The evaluation set is always the same
- Show the relevant KPIs (described in the chapter before)

## 5.2 Online/Production Experiment Setup

### 5.2.1 Experiment Setup

- Describe the online shop
- Where do we have recommendations, general description
- Explain SOFI
- Where do we put our recommendation system? Against what does it compete?

### 5.2.2 Measurements

- Show results for AutoML, Often Bought Together and our system in the different variants
- Show the relevant KPIs

Chapter 6

# Results

- bring everything together
- compare best performing model to automl from google

Appendix A

# Dummy Appendix

You can defer lengthy calculations that would otherwise only interrupt the flow of your thesis to an appendix.

# Bibliography

[1] Erik Bernhardsson. Music discovery at Spotify. https://www.slideshare.net/erikbern/music-recommendations-mlconf-2014, 2014.

[2] Paul Covington, Jay Adams, and Emre Sargin. Deep Neural Networks for YouTube Recommendations. https://ai.google/research/pubs/pub45530, 2016.

[3] Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor Forcing: A New Algorithm for Training Recurrent Networks. http://arxiv.org/abs/1610.09038, 2016.

[4] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. http://arxiv.org/abs/1706.04148, 2017.

[5] Marcel Urech. Migros CIO Interview. https://www.netzwoche.ch/news/2017-06-29/martin-haas-ueber-cumulus-twint-und-die-migros-it, 2017.

[6] Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-Prod2Vec - Product Embeddings Using Side-Information for Recommendation. http://arxiv.org/abs/1607.07326, 2016.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

# Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

---

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

**Title of work** (in block letters):

**Authored by** (in block letters):
*For papers written by groups the names of all authors are required.*

| **Name(s):** | **First name(s):** |
|---|---|
| | |
| | |
| | |
| | |
| | |

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

| **Place, date** | **Signature(s)** |
|---|---|
| | |
| | |
| | |
| | |
| | |

*For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.*