INDUSTRIAL TRAINING

(PHASE-I)

Undergone at

LMS SOLUTIONS Pvt. Ltd.

Indore, M.P

A PRESENTATION REPORT

On

Machine Learning: Hybrid Recommender Systems

Submitted by

Tanmay Agrawal
RA1611008010702

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

June, 2018

BONAFIDE CERTIFICATE

Certified that the report on **Machine Learning: Hybrid Recommender Systems** is a proof of successful completion of Industrial Training Phase–I programme undergone by **Tanmay Agrawal** (Register no. **RA1611008010702**) in the company **LMS Solutions Pvt. Ltd.** located at **Indore, M.P** during the period **1.06.2018** to **30.6.2018**.

Date **Signature of the**

17.09.2018 **Industrial Training In-charge**

DECLARATION

I hereby declare that the presentation report submitted titled "**Machine Learning: Hybrid Recommender Systems**", is a record of my industrial training programme which I had undergone in the company **LMS Solutions Pvt. Ltd.** located at **Indore, M.P** during the end of the fourth semester between the period **01.06.2018** to **30.06.2018**.

Date : 17.09.2018

Name : Tanmay Agrawal

Register Number : RA1611008010702

Signature of the Student

RUBRICS FOR EVALUATION

S. No	Marks Split up	Maximum marks	Marks Obtained
1	Report Preparation	50	
2	Presentation	25	
3	Quiz and Viva	25	
	Total	100	

Signature of the Staff

TABLE OF CONTENTS

Consulting-Technology-Outsourcing

Dedicated to Deliver

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Tanmay Agrawal** a student of **SRM Institute of Science and Technology Kattankulathur, Chennai.** has completed 1-month internship program (starting from June 1, 2018) at LMS Solutions (India) Pvt. Ltd. He learned Machine learning and artificial intelligence from our engineering team during the mentioned period. We found him hardworking and inquisitive.

We wish him every success in life.

Sincerely,
For LMS Solutions (India) Pvt. Ltd.

Abdul Hussain Arif
Director

June 30, 2018

ABOUT THE COMPANY

LMS is a rapidly growing IT company with extensive experience in designing and developing cutting edge custom software solution. We are a group of highly motivated IT professionals with ability to innovate and a strong desire to excel. LMS is working to provide solutions, which are delivered with reliability and timeliness, flexibility and low cost. Customer satisfaction is our primary aim.

LMS team has been closely working with clients across verticals to not only help them run their businesses seamlessly over the years but also to make them Next Gen ready. LMS has successfully demonstrated their technical expertise integrated with high quality domain knowledge to deliver prestigious projects on Health Care, Finance, Retail, Real Estate, Telecom and Social Networking Verticals. They are among the earliest to empower their esteemed clients with the latest trends of Business Intelligence, Data Mining and Analytics.

INTERNSHIPS AT LMS

Summer Internship assignments at LMS last 4-8 weeks. Interns have the chance to be a part of the unique culture and work environment at LMS. Interns are offered a suitable level of challenge, a variety of work projects (often in a team environment), along with the coaching and support from mentors and professionals who help students learn what it takes to become successful in the field. LMS has a well-structured internship program that sets their interns up for success. Students have the opportunity to utilize the knowledge they have gained in the classroom and see how their studies translate into a professional setting. It's no surprise that an internship helps bridge the gap between being a successful student and an invaluable member of the LMS team.

DAILY DIARY

Day 1	Introduction to python
Day 2	Installing Tensorflow
Day 3	Leave
Day 4	Learning NumPy
Day 5	Learning Pandas
Day 6	Weekend leave
Day 7	Weekend leave
Day 8	Finding out about different ML algorithms.
Day 9	Recommender System reading
Day 10	Leave
Day 11	Studying Collaborative Filtering
Day 12	Continued
Day 13	Weekend leave
Day 14	Weekend leave
Day 15	Studying Content based Filtering
Day 16	Continued
Day 17	Leave
Day 18	Research on different cases of recommender systems.
Day 19	Research on different cases of recommender systems.
Day 20	Weekend leave
Day 21	Weekend leave
Day 22	Using python to implement recommender system code for the client website.

Day 23	continued
Day 24	Leave
Day 25	Tweaking the code to make improvements
Day 26	Project continued
Day 27	Weekend leave
Day 28	Weekend leave
Day 29	Final Changes
Day 30	Committing the code

WORK DONE

Machine Learning: Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

SOFTWARE USED*:*

1. **ANACONDA**:- **Anaconda** is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management system *conda.* The Anaconda distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and macOS[. Anaconda distribution comes with more than 1,000 data packages as well as the Conda package and virtual environment manager, called Anaconda Navigator. The open source data packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. You can also make your own custom packages using the conda build command, and you can share them with others by uploading them to Anaconda Cloud. PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda PythonMozillapipe.

LANGUAGES/LIBRARIES USED:-

1. **Python**:- Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a

design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software of Python Software Foundation.

2. **Pandas**:- In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

3. **NumPy**:- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

4. **TensorFlow**:- TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google, often replacing its closed-source predecessor, DistBelief. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open source license on November 9, 2015.

PROJECT HANDLED

Project comprises of implementing Recommender system using python libraries, for an education and mentorship website.

MODULE1: Installing TensorFlow and dependencies

1) Install Python and the TensorFlow package dependencies:
    a) sudo apt install python-dev python-pip # or python3-dev python3-pip

2) Install the TensorFlow pip package dependencies (if using a virtual environment, omit the --user argument):
    a) pip install -U --user pip six numpy wheel mock
    b) pip install -U --user keras_applications==1.0.5 --no-deps
    c) pip install -U --user keras_preprocessing==1.0.3 --no-deps

3) Install Bazel:
    Install Bazel, the build tool used to compile TensorFlow. Add the location of the Bazel executable to your PATH environment variable.

4) Download TensorFlow source code:

    a) git clone https://github.com/tensorflow/tensorflow.git
    b) cd tensorflow
    c) git checkout branch_name # r1.9, r1.10, etc.
    d) bazel test -c opt -- //tensorflow/... -//tensorflow/compiler/... -//tensorflow/contrib/lite/...

5) Configure the build:
    Configure your system build by running the following at the root of your

TensorFlow source tree:

This script prompts you for the location of TensorFlow dependencies and asks for additional build configuration options (compiler flags, for example). The following shows a sample run of ./configure (your session may differ):

a) ./configure

6) Build the pip package:

GPU Support:

To make the TensorFlow package builder with GPU support:

a) bazel build --config=opt --config=cuda
//tensorflow/tools/pip_package:build_pip_package

Bazel build options:

Building TensorFlow from source can use a lot of RAM. If your system is memory-constrained, limit Bazel's RAM usage with: --local_resources 2048,.5,1.0.

The official TensorFlow packages are built with GCC 4 and use the older ABI. For GCC 5 and later, make your build compatible with the older ABI using: --cxxopt="-D_GLIBCXX_USE_CXX11_ABI=0". ABI compatibility ensures that custom ops built against the official TensorFlow package continue to work with the GCC 5 built package.

7) Build the package:

The bazel build command creates an executable named build_pip_package—this is the program that builds the pip package. For example, the following builds a .whl package in the /tmp/tensorflow_pkg directory:

a) ./bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorflow_pkg

Although it is possible to build both CUDA and non-CUDA configurations under the same source tree, it's recommended to run bazel clean when switching between these two configurations in the same source tree.

8) Install the package:

   The filename of the generated .whl file depends on the TensorFlow version and your platform. Use pip install to install the package, for example:

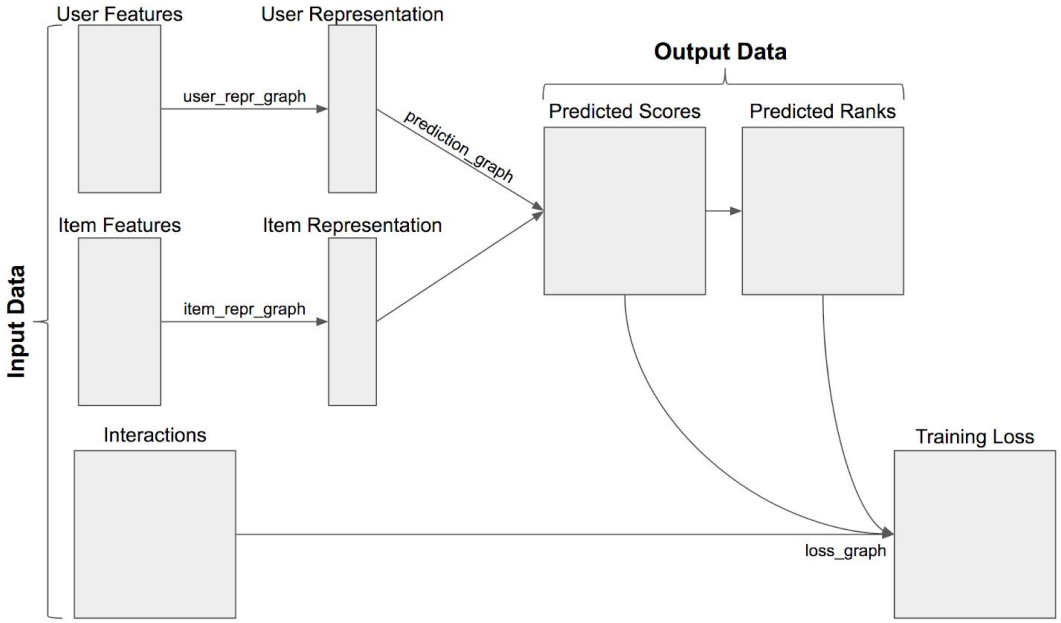
   a) pip install /tmp/tensorflow_pkg/tensorflow-version-cp27-cp27mu-linux_x86_64.whl

MODULE 2: TensorRec

TensorRec is a Python recommendation system that allows you to quickly develop recommendation algorithms and customize them using TensorFlow.

TensorRec lets you to customize your recommendation system's representation/embedding functions and loss functions while TensorRec handles the data manipulation, scoring, and ranking to generate recommendations.

A TensorRec system consumes three pieces of data: user_features, item_features, and interactions. It uses this data to learn to make and rank recommendations.



Examples:

Basic usage:

```
import numpy as np
import tensorrec


# Build the model with default parameters
model = tensorrec.TensorRec()


# Generate some dummy data
interactions, user_features, item_features = tensorrec.util.generate_dummy_data(
   num_users=100,
   num_items=150,
   interaction_density=.05
)


# Fit the model for 5 epochs
model.fit(interactions, user_features, item_features, epochs=5, verbose=True)


# Predict scores and ranks for all users and all items
predictions = model.predict(user_features=user_features,
```

```
                        item_features=item_features)
predicted_ranks = model.predict_rank(user_features=user_features,
                                item_features=item_features)


# Calculate and print the recall at 10
r_at_k = tensorrec.eval.recall_at_k(predicted_ranks, interactions, k=10)
print(np.mean(r_at_k))
```

Attention Example:

```
from tensorrec import TensorRec
from tensorrec.eval import fit_and_eval
from tensorrec.representation_graphs import (
  LinearRepresentationGraph, NormalizedLinearRepresentationGraph
)
from tensorrec.loss_graphs import BalancedWMRBLossGraph

from test.datasets import get_movielens_100k

import logging
logging.getLogger().setLevel(logging.INFO)

# Load the movielens dataset
train_interactions, test_interactions, user_features, item_features, _ =
get_movielens_100k(negative_value=0)

# Construct parameters for fitting
epochs = 500
alpha = 0.00001
n_components = 10
verbose = True
learning_rate = .01
n_sampled_items = int(item_features.shape[0] * .1)
fit_kwargs = {'epochs': epochs, 'alpha': alpha, 'verbose': verbose, 'learning_rate': learning_rate,
          'n_sampled_items': n_sampled_items}

# Build two models -- one without an attention graph, one with a linear attention graph
model_without_attention = TensorRec(
  n_components=10,
  n_tastes=3,
  user_repr_graph=NormalizedLinearRepresentationGraph(),
  attention_graph=None,
  loss_graph=BalancedWMRBLossGraph(),
)
```

```
model_with_attention = TensorRec(
  n_components=10,
  n_tastes=3,
  user_repr_graph=NormalizedLinearRepresentationGraph(),
  attention_graph=LinearRepresentationGraph(),
  loss_graph=BalancedWMRBLossGraph(),
)
results_without_attention = fit_and_eval(model=model_without_attention,
                                         user_features=user_features,
                                         item_features=item_features,
                                         train_interactions=train_interactions,
                                         test_interactions=test_interactions,
                                         fit_kwargs=fit_kwargs)
results_with_attention = fit_and_eval(model=model_with_attention,
                                      user_features=user_features,
                                      item_features=item_features,
                                      train_interactions=train_interactions,
                                      test_interactions=test_interactions,
                                      fit_kwargs=fit_kwargs)
logging.info("Results without attention: {}".format(results_without_attention))
logging.info("Results with attention:    {}".format(results_with_attention))
```

MODULE 3: Term Frequency–Inverse Document Frequency:

In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. Tf–idf is one of the most popular term-weighting schemes today; 83% of text-based recommender systems in digital libraries use tf–idf.

Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf–idf can be successfully used for stop-words filtering in various subject fields, including text summarization and classification.

One of the simplest ranking functions is computed by summing the tf–idf for each query term; many more sophisticated ranking functions are variants of this simple model.

- Term frequency:
  Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, "the brown cow". A simple way to start out is by eliminating documents that do not contain all three words "the", "brown", and "cow", but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its term frequency. However, in the case where the length of documents varies greatly, adjustments are often made (see definition below). The first form of term weighting is due to Hans Peter Luhn (1957) which may be summarized as:

The weight of a term that occurs in a document is simply proportional to the term frequency.

- Inverse document frequency:
  Because the term "the" is so common, term frequency will tend to incorrectly emphasize documents which happen to use the word "the" more frequently, without giving enough weight to the more meaningful terms "brown" and "cow". The term "the" is not a good keyword to distinguish relevant and non-relevant documents and terms, unlike the less-common words "brown" and "cow". Hence an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

Karen Spärck Jones (1972) conceived a statistical interpretation of term specificity called Inverse Document Frequency (idf), which became a cornerstone of term weighting:

The specificity of a term can be quantified as an inverse function of the number of documents in which it occurs.

Code Example:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel


ds = pd.read_csv("sample-data copy.csv")


tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3), min_df=0, stop_words='english')
tfidf_matrix = tf.fit_transform(ds['description'])


cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)


results = {}
```

```
for idx, row in ds.iterrows():
  similar_indices = cosine_similarities[idx].argsort()[:-100:-1]
  similar_items = [(cosine_similarities[idx][i], ds['id'][i]) for i in similar_indices]


  # First item is the item itself, so remove it.
  # Each dictionary entry is like: [(1,2), (3,4)], with each tuple being (score, item_id)
  results[row['id']] = similar_items[1:]


print('done!')



def item(id):
  return ds.loc[ds['id'] == id]['description'].tolist()[0].split(' - ')[0]



def recommend(item_id, num):
  print("Recommending " + str(num) + " best mentors suited to your profile " + """ +
item(item_id) + """ + ":")
  print("-------")
  recs = results[item_id][:num]
  rank = 1
  for rec in recs:
    print("Mentor #" + str(rank) + " " + item(rec[1]) + " (score:" + str(rec[0]) + ")")
    rank = rank + 1



recommend(item_id=498, num=10)
```

LEARNING AFTER TRAINING

I learnt the usage and importance of various tools apart from the ones used in my project. I am confident enough to implement a Machine Learning algorithm for any occasion and to do freelancing projects.

There are enormous advantages of Machine Learning like it pays the bills like everyone else says. Machine learning (ML) is a category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. New technologies come out non-stop giving us new toys to play with and new things to learn**.** If we're a geek and like to tinker with things and like to see how things work, this a great field for it. Probably my favourite part of being a machine learning engineer - while with most jobs there's not much tangible you can actually show people out side of your company. As a web developer probably the majority of your work will all be public facing, and if you take pride in what you do each of those algorithms will be like works of art.

I learnt quite a lot about python, TensorFlow, pandas, NumPy and PyTorch which will be very helpful to me in machine learning.

SUMMARY

During my 30 days training I worked for the project work entitled "Recommender System using TensorFlow" and I implemented an algorithm for my project. It was a wonderful learning experience for me while working on this project. This project took me through the various phases of project development and gave me the real insight into the world of Computer Science Engineering. Familiarity with the industry level implementation of a real machine learning algorithm is now known to me.