# A LSTM Based Personalized News Recommendation System

**Final Report**

**Ketki Savle, Yangqi Su, Divya Kulkarni, Rakesh Harish**

## Introduction:

Reading news online has become very popular as the web provides access to news articles from millions of sources around the world. However, a key challenge of news websites is helping users find the articles that fits reader interests. This not only includes articles that may be of interest to the general public, but also news recommendations that are tailored to each individual reader. This challenge becomes even more pressing since an individual's preferences may change with time. Here we are aiming to recommend news articles to the profile based users as per their interest and previously read articles.

### Problem Statement:

Through this project we have tried to address the personalized news recommendation implementation which is different than the traditional collaborative and content based filtering approach. Unlike product recommender, the content of the news recommendation system is highly dynamic, our aim is not only to recommend the news articles as per user's interests but also to improvise on the cold start area which remains the challenge in the field of news recommender system.

### Motivation:

The motivation to do this project comes from our common interest to implement a recommender system. We wanted to use the real world dynamic news and user data to be able to build a system. Earlier many researchers have worked on news recommendation system using content based and collaborative filtering. Most of them have used smaller and static user interaction dataset for limited number of articles. Our enthusiasm was to build a system using real time and large data also, to use the Neural network architecture to train the model unlike other researchers who have used other Machine Learning algorithm such as Naive Bayes, Logistic Regression, KNN etc.

The motivation behind this project was to use the 16GB of the Adressa dataset unlike other research where most of the time static and relatively smaller dataset was used. Also use Neural network architecture to train the model to recommend the news articles to the users.

Finally, the recommender systems are pervasive. They play critical role in every area of consumer industry and thus we wanted to apply our Machine Learning knowledge to be able to build a recommender system.

**Review Of Other Researches:**

Recommendation systems in general can be divided into collaborative recommendation and content based recommendation. In a narrower sense, in collaborative filtering based recommendations, an item is recommended to a user if similar users liked that item. Collaborative filtering can be further divided into user collaborative filtering, item collaborative filtering or a hybrid of both user and item collaborative filtering. Some of the technique include Bayesian matrix factorization, matrix completion, Restricted Boltzmann Machine, nearest neighbour modelling, etc. Liu et al. [2] proposed a news recommender system which predicts user interest via Bayesian Framework, and recommends news articles through collaborative filtering method.

Early pioneer work which used neural network was done, where a two-layer Restricted Boltzmann Machine (RBM) is used to model users' explicit ratings on items. The work has been later extended to model the ordinal nature of ratings. Recently autoencoders have become a popular choice for building recommendation systems

A general framework named NCF, short for Neural Collaborative Filtering that replaces the inner product (which calculates the similarity between a user and an item) with a neural architecture that can learn an arbitrary function from the given data. It uses a multi-layer perceptron to learn the user-item interaction function. NCF is able to express and generalize matrix factorization.

For Location based news recommender system, GeoFeed and GeoRank recommend to users some news happening at the users' current locations or within a given range, where GeoRank uses only static point locations of both users and news, while GeoFeed allows news with spatial extent. The Euclidean distance between user and news locations to measure the importance of news. The locations are described using topic vectors, and the relevance of a news article to a user. Other researchers used Location-Aware Personalized News Recommendation With DSA that is measured by the similarity between the topic vectors of the news and the current location of the user. Therefore, the topic representations of locations are crucial for topic based location-aware news recommendation, and a range of topic models (such as Latent Dirichlet Allocation (LDA),Explicit Semantic Analysis (ESA), Probabilistic Latent Semantic Analysis

(PLSA), and their improved models, have been used for geographical topic-based location-aware news recommendation.

**Open Questions in the Domain:**

**Privacy:** Exposure of user profile data may lead to privacy concerns in Recommender system. For more generic news recommendation, the system collects user click history and page access pattern implicitly. The need and association of user profiles give rise to privacy concerns in the news domain, whereas privacy of user identity, user behavior in terms of page access patterns contributes to the overall privacy risks in the news domain.

**Serendipity:** Recommender system could also isolate the users with same choices, never allowing users to read completely different news. This seems to be contradictory as system could be limiting user's variety of different articles while showing similar articles from different sources. Users would be trapped in an information bubble where the recommender system would control the user.

**Scalability:** Will the model work on larger datasets like Google News. News articles tend to be in flood within a short period of time, requiring much more computation for recommendation. Can the methods used in this recommender system be used in other specific domains like Amazon, Netflix, Pandora Radio etc.

**Changing Users Interest:** Typically, news reading services retrieve news articles relevant to reading preferences of individual users, and adapt their services based on the change of the user's reading interests by employing different recommendation approaches. The interest of news articles with respect to a user is regressive. i.e. after he/she clicks the first piece of news he/she interested in, the interest of the user on a news item may change if he chooses other news items also. The news reading preferences of people can be affected by ongoing circumstances in the world as well as their age, cultural level and even their mood.

**Recency:** News articles typically have short shelf lives. For example, some sports fans would be concerned with the past breaking scores of four days ago. In contrast, the shelf lives of products and movies could extend for several months or even years. Most of the users want to read fresh news instead of old dated articles. So the importance of news items decreases in time. On the other hand, some news articles may be connected with each other that the user may want to read the previous news items related to the one she already reads or he/she may want to keep informed about that subject

**Cold Start:** A problem may arise in collaborative filtering technique where the recommender system cannot recommend new news article if it does not have user's click history/data. Moreover it would become more difficult to recommend if the new user's data is unavailable.

**Synonymy:** Some news items could be named differently but would carry the same meaning. Hence would the system be able to identify that it is referring to the same item. For example even if the "children's movie" and "children's film" have the same meaning, they can be treated as different items by the recommender system.

**Proposed Approach:**

Here we will aim to recommend articles and news to the user by implementing a LSTM neural network model that can learn the user's preferences using the user's news interaction history. In short, either a fix sized sequence of interactions or a fixed time frame of interactions will be fed to the model and the output will be used to make recommendations to users by ranking its cosine similarity with news the user has not seen and selecting the top ranked news as recommendations.

## Background:

Most recently, Park et al[2], proposed a session based and history based Recurrent Neural Network that was made for news recommendation. In their work, they used word embedding to represent articles from the NAVER dataset and used user click sessions to predict the recommendation articles. Furthermore, they applied a Convolutional Neural Network for the modeling of a user's long term preferences and ranked the recommended articles according to the user's long term preferences. For their RNN, they used a custom pair ranking loss function. Their model used the two RNNs for the recommendation of articles that were close to the user's immediate and short terms interests, and then re-ranked the articles according to the similarity between the articles categories and the user's long term category preferences.

The work we present here is partially inspired by their work and the work of others that attempted to use recurrent neural networks in session or sequential based recommendation systems.

## Methods:

### Dataset:

To achieve this goal, we have used The light version of the Adressa dataset. This dataset consists of news articles from Norway region that users have read in the first 3 month of 2017. Along with the information on the articles, the data captures user's interaction with the article when he or she is reading it, time spent reading a particular article, location from where user has read the article, etc.

### Pre-processing:

The Adressa dataset according to its source contains 48486 articles and over 27 million user-article interactions of 3083438 users[4]. Of all the users, there are 380527 subscribing users for which we have recorded interaction data. Upon preprocessing the light version of the dataset (containing only key terms of each article) however, we found that there were in fact 132597 different items (articles, videos, photos etc.) that were being viewed by users during those 3 month.

Of all these items, we filtered out those that did not contain any information (items which had information on concepts, title, keywords, people, entities, and news location; if all 6 features were missing, then the item was dropped) regarding the item in the dataset, leaving 96922 items in total. Of the interactions, we only kept interactions that happened between subscribing users and the filtered items, which accounted for over 99% of all interactions, thus the filtering out of items did not significantly impact the dataset. Further subsetting of the data included the following steps:

1. Only include interactions that had an active time of larger than 0 seconds.
2. Only include users that had over 500 interactions (active time > 0s).

Post subsetting of the data, we embedded each item to a FastText vector using a pre trained word embedding matrix given here [5]. Each item had some information regarding at least one of the previously described 6 features. The information was mapped to the FastText matrix, multiplied by a given weight from the source data if applicable and summed for each item. The end result was each item had a vector of 300 features.

Some items came with a classification feature while others did not. There were a total of 23 classes that were given by the original dataset. To remedy the missing classifications, we performed a crude process of generating classifications using the following method:

1. URLs of articles that had the classification feature were retrieved and keywords in the URLs were extracted.
2. There were in total around 60 frequent keywords, each of which was matched to a classification vector of length 23, which were the probabilities that that keyword was associated with the classification.
3. URLs of articles without classification had their keywords extracted and matched to one of the 60 keywords to obtain a probability vector that indicated its classification.

The final representation of an item was the concatenated length 23 vector and the 300 feature vector.
For each user item interaction, the vectorized representation included a final feature that is the normalized active time of the interaction (all active time of interactions of a user was divided by the max active interaction time of the user), making each interaction a length 324 (23+300+1) vector.

For the goal of the project, we then took the following 2 approaches to build the training data:

1. Use a fixed sized sequence (timesteps) of user item interactions to predict the next item the user will like to read (i.e. use previous 30 user item interactions to predict the 31st item, the target will be a length 300 vector while the features will be 30x324)
2. Use a fixed time frame of user item interactions to predict the next item the user will likely interact with (i.e. use all interactions with a 3 hour time frame (size (timesteps) variable) to predict the item interacted right after the 3 hour time frame)

For model testing and validation during training, interaction data from the last 5 days was left out. Of the last 5 days of interactions, 95 % (**96324** out of **102317**) of all items interacted with during those 5 days were unique to those 5 days, meaning they did not appear in the training data. We then filtered out interactions that were with items that appeared in the training data to generate the Cold Start test data set for validation and testing.

**Models:**
Implementations of the all models were done using **Keras (v.2.1.6)** with **Tensorflow** as the backend.
Prior to building the model, hyper-parameter optimization was performed using **hyperas**, which is a wrapper of the optimization package hyperopt dedicated for Keras models. Here, we use the Tree-structured Parzen Estimator (TPE) Approach implemented in

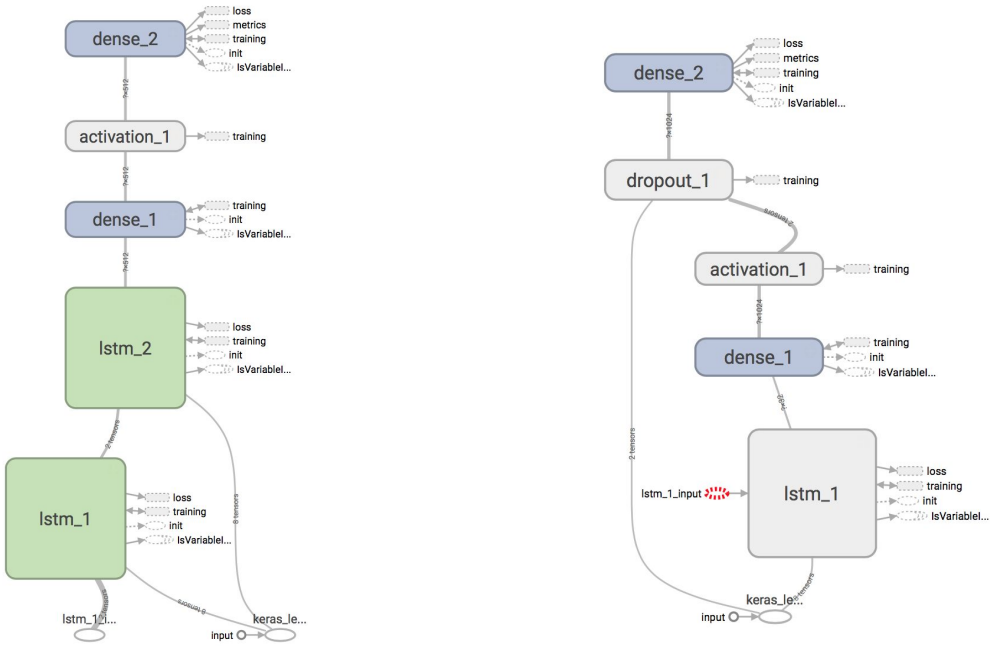
**hyperopt** to find the best estimators. The TPE algorithm is essentially an optimization algorithm that tries model the likelihood of the hyperparameters to produce a good model. The parameters we search for included: **# of units in each layer** (LSTM and Dense), **# of LSTM layers**, **dropout probability**, **L2 regularization coefficients** and **optimizers**. Due to time concerns, the parameters were evaluated using only 5 epochs, and total amount of evaluation of all parameters was set to 50. With the outputs of the hyper-parameter optimization, we then built our LSTM models.

**Evaluation:**

Metrics commonly used for the evaluation of recommendation systems include HR@K (hit rate at rank k), R@K (recall at rank k) and MRR@K (mean reciprocal rank at rank k). Currently, this portion is still under development, thus in this report, the only evaluations are of the cosine similarity scores of the cold start test data.

## Experiments:

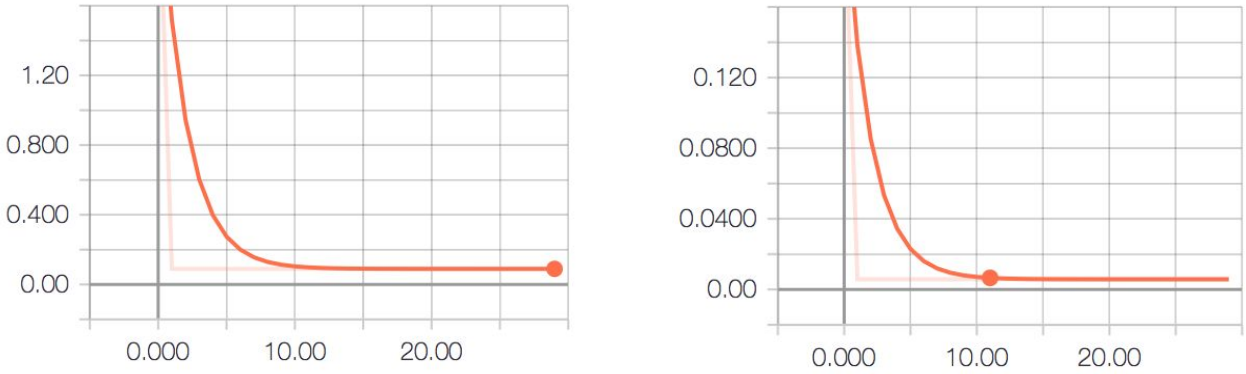
After hyper-parameter tuning using **hyperas,** we were able to obtain the best possible parameters for each of the 2 networks. The best network structures are shown below:




**A** **B**

**The LSTM models used for A. Fixed Sized Sequence and B. Fixed Timeframe Sequence Prediction.**

The fixed sequence size based LSTM shown above (A) has 256 units in the first LSTM, followed by a stacked LSTM on top which has 512 units. The final LSTM outputs into a dense layer of 512 hidden units before Relu activation and final Dense layer that outputs a length 300 vector used to find the possible article to recommend.
The time frame based LSTM shown above (B) has 32 hidden units, followed by one dense layer of 1024 units, after which the outputs go through a Relu activation and a dropout (probability 0.5) layer before the final dense layer which outputs a vector of length 300.
All kernel weights in the LSTM were regularized using L2 regularization, and dropouts were also added to the LSTM layers.
The loss function used in both the Networks is mean squared error. However, validation of the models ( and hyperparameter tuning) was done using the cosine similarity metric as a guide. As shown below, both model training loss converged.

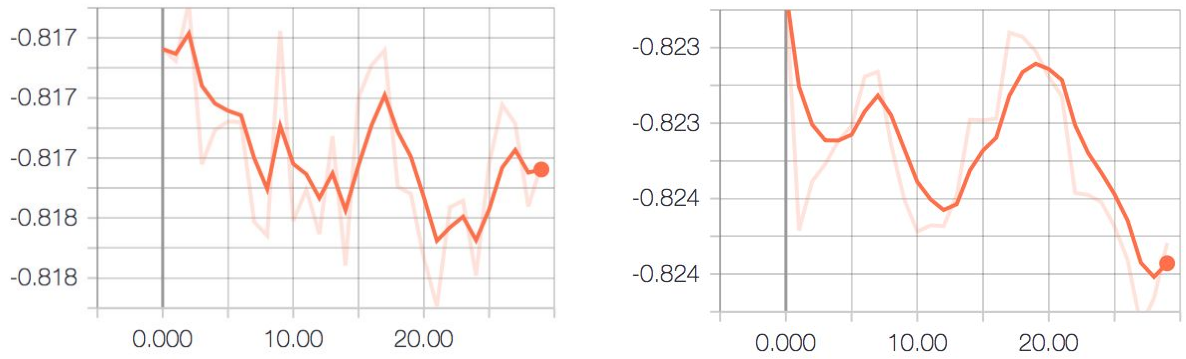



A. B.

**Mean square error loss of A. Fixed Sized Sequence and B. Fixed Timeframe Sequence Prediction**

## Comparison of Models:

Looking at the results of the 2 models (shown below), it becomes clear that the time-frame based model slightly performs better than the fixed sequence/timestep size based model. This could be because using time frame is more relevant for news recommendation, since a fixed length of sequence might cross the boundaries where a user's interests change, while it is not very possible that a user's preference change dramatically in a fixed amount of time. Furthermore, The best time-frame model contains significantly less parameters than the best fix sequence size model, which could mean that it can generalize better and that the fix sequence size model might be slightly overfitting. However, the differences are not as pronounced, thus further testing and evaluation using other metrics are required.




**A.** **B.**

**Comparison of Cosine Similarity Metric between the 2 models on Cold Start Data**

**A. Fixed Sized Sequence and B. Fixed Timeframe Sequence Prediction**

## Conclusions:

Personalized recommender systems are becoming widely used solution for reducing information overload of diverse domains. Nowadays, it is evident to see a recommender system working in the background of any website that you visit. Even though there are many uses of this system still many challenges persists for a true user based personalized recommender system. In the project, one thing became very evident in the beginning, which was the importance of preprocessing the data. Data preprocessing became the bulk of our work, and was the most time consuming, especially due to the fact the the data was in Norwegian and thus many aspects were not as intuitive. Furthermore, model building, though simple at first sight, actually required a decent amount of coding for the implementations of hyper-parameter optimization. Finally, there is still much to learn in the evaluation of these recommendation model, as many of the metrics used in recommendation systems are from the field of information retrieval which took much time to understand.

## Future Work:

Further work would involve further evaluation of the models using the aforementioned metrics for a better view of how good the model is actually doing in recommending news to users. Other improvements would involve pulling the actual article contents from the websites using the item URLs to generate a better representation of the item, implementing a classifier to better classify items without classification instead of using prior vectors, include user location data, which is already available, as a feature in the prediction process and create an end-to-end agent that

automatically recommends articles to the user using a mix of Reinforcement learning instead of calculating cosine similarity.

## Response to the feedback:

As we met our instructor on multiple occasions during the progress of the report, as per the most important suggestion, professor expected us to come up with the project idea that is our own research work in recommendation domain. Our dataset is unique and has not been fully utilized by other researchers to run the experiment on such size of the data. Preprocessing of this data has solely been done on our own.

The implementation of the model also is unique in its nature as the input LSTM layer was our own way of computing. The other researchers though have used the LSTM technique, they have computed several different types of input for the neural nets model.

**References:**


[1] Vaibhav Kumar, Dhruv Khattar, Shashank Gupta, Manish Gupta1,Vasudeva Varma, "User Profiling based Deep Neural Network for Temporal News Recommendation" International Institute of Information Technology Hyderabad, 2017.

[2] Park, Lee, Choi, "Deep Neural Networks for News Recommendations", CIKM 2017, November 6-10, 2017, Singapore.

[3] David Zhan Liu, Gurbir, "A Recurrent Neural Network Based Recommendation System", Department of Computer Science 3 Stanford University 4 Stanford, CA 94305.

[4] Adressa Dataset: https://dl.acm.org/citation.cfm?id=3109436

[5] Facebook pretrained FastText Matrix:
**https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md:**