Intelligent controller for predictive caching in AR/VR: modeling and analysis of daily living in-home scenarios

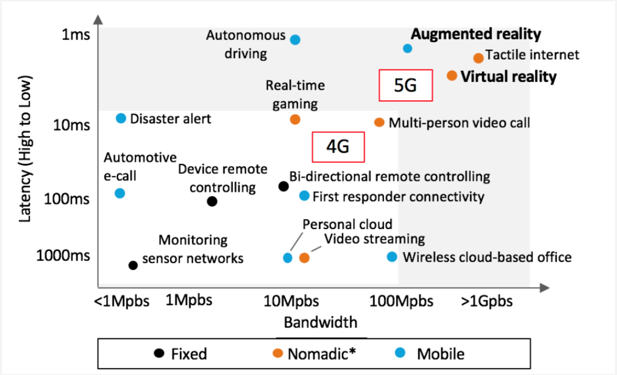
Sharare Zehtabian, Siavash Khodadadeh, Ladislau Bölöni, and Damla Turgut

Department of Computer Science, University of Central Florida
Orlando, Florida, United States of America

Presenter's Email Address: sharare.zehtabian@knights.ucf.edu

Outline

Problem Statement



source: Enabling Mobile Augmented and Virtual Reality with 5G Networks, 2017

- AR/VR experiences require the delivery of large amount of data with low latency.
- Checking the news for major sport events, weather or traffic report, so on. `https://www.youtube.com/watch?v=URnLuSrlNA8`

Problem Statement

- Users' satisfaction relies significantly on the quality level of the delivered information and how quickly it is delivered.
- Deciding what to cache and where to cache are crucial problems.
    - Downloading all the information from the network: bandwidth is costly, especially when the highest quality is requested.
    - Caching copy of the data associated with the highest quality: the device's storage capacity might be limited.

One intelligent solution: predict what experience will the user request and in which time frame of the day this request will happen.

Related Work- User Modeling

- Efficient delivery of experiences requires us to model the user based on his or her behavior.
- Several techniques have been generated by machine learning and reasoning under uncertainty for predictive statistical modelings. For example decision tree, neural networks, classification and rule-induction methods, and Bayesian networks.
- Guha et al. deployed a user modeling system for Google Now personal assistant based on long-term user history with thousands of queries and clicks (2015).

Similarly, we use the user's daily activities to create synthetic request data in an in-home assistant AR/VR scenario.

Related Work- Networking Requirement in AR/VR

- In reporting major sports events or music education, high-quality video streams should be transmitted via a very reliable network.
- It gets even more challenging when many users are requesting the same content at the same time.
- Westphal suggested information-centric networks as a potential architecture to assist the deployment of AR/VR (2017).

We propose a central AR/VR controller responsible for making decisions about the type and quality of the information

Related Work- Caching Strategies for AR/VR

- J. Chakareski designed an optimization framework to maximize the reward that a multi-cellular system can earn when serving AR/VR users by enabling the base stations to select cooperative caching/streaming/edge-computing strategies (2017).
- Du et al. proposed a pre-fetching approach at the user end to preload videos before user requests from a famous Swedish TV service provider by analyzing the request patterns over eleven weeks. (2015).
- Koch et al. used a Convolutional Neural Network (ConvNet) to extract music features to predict the content that is more likely to be accessible within the next time (2018).

Our Work

Motivation:

- We are considering the AR/VR for daily use within a household environment
- The experience quality is limited by the
  (1) Capabilities of the devices through which it is delivered
  (2) Signal limitations such as network delay and bandwidth limitations.

Goals:

- Designing AR/VR controller for making decisions about the forms and quality of the contents, as well as predictive caching and pre-rendering of these experiences.

Outline

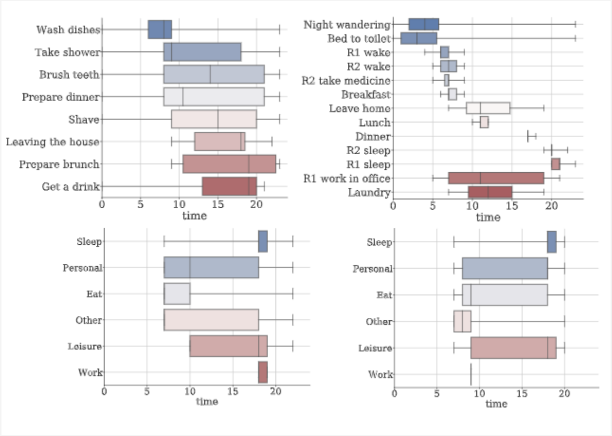
User Modeling

- AR/VR controller learns models of the user behavior that allows it to predict the experiences the user will request and adjusts the caching strategy accordingly.
- Scarcity of training data is a major challenge.

Real-world datasets of user behavior in homes and simulated dataset

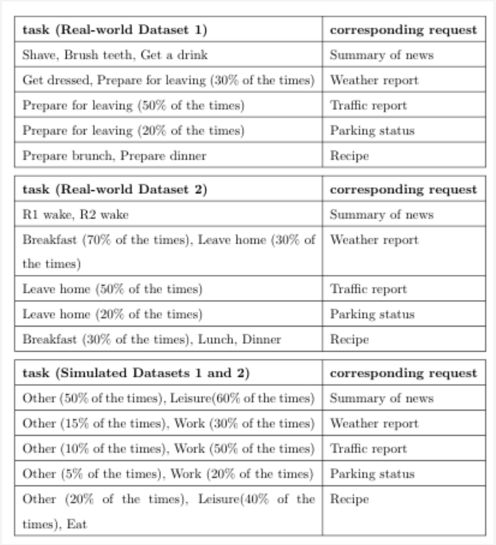
- Real Dataset 1: describes the activities of a 26-year-old man in a smart home with 14 state-change sensors. Sensors were left unattended, collecting data for 28 days in the apartment.
- Real Dataset 2: collected by the CASAS research group, describes the activities of 2 residents in an apartment for 57 days.
- Simulated Datasets: Open Smart-Home simulated (OpenSHS) in which participants performed the Activities of their Daily Livings (ADLs) for different contexts in this simulation environment.

Creating realistic synthetic datasets of AR/VR experience requests



We probabilistically associated certain experiences with activities that are present in the dataset using common-sense associations.

Creating realistic synthetic datasets of AR/VR experience requests



task (Real-world Dataset 1)	corresponding request
Shave, Brush teeth, Get a drink	Summary of news
Get dressed, Prepare for leaving (30% of the times)	Weather report
Prepare for leaving (50% of the times)	Traffic report
Prepare for leaving (20% of the times)	Parking status
Prepare brunch, Prepare dinner	Recipe

task (Real-world Dataset 2)	corresponding request
R1 wake, R2 wake	Summary of news
Breakfast (70% of the times), Leave home (30% of the times)	Weather report
Leave home (50% of the times)	Traffic report
Leave home (20% of the times)	Parking status
Breakfast (30% of the times), Lunch, Dinner	Recipe

task (Simulated Datasets 1 and 2)	corresponding request
Other (50% of the times), Leisure(60% of the times)	Summary of news
Other (15% of the times), Work (30% of the times)	Weather report
Other (10% of the times), Work (50% of the times)	Traffic report
Other (5% of the times), Work (20% of the times)	Parking status
Other (20% of the times), Leisure(40% of the times), Eat	Recipe

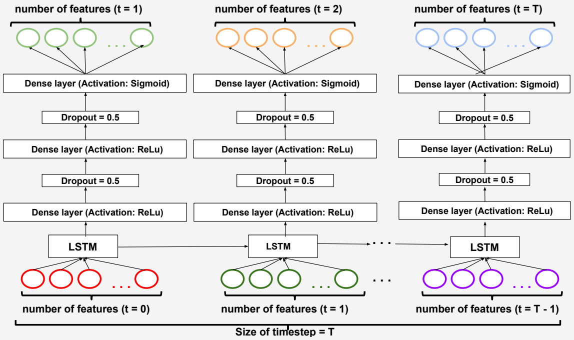
Outline

Probability-based caching

- The proposed algorithm is based on calculating the probability of a specific request in a specific time interval by counting the occurrences of the data in the training set.
- We define 24 intervals with the length of 1-hour for each day in the datasets.

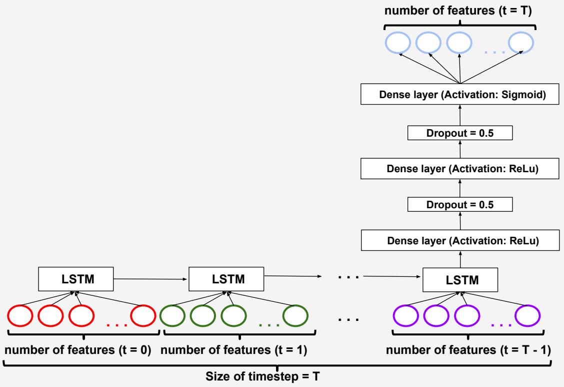
LSTM-based caching-Architecture 1: many-to-many

It can not only process single data points, but also entire sequences of data. So a good choice for predicting the next request.


number of features (t = 1)
number of features (t = 2)
number of features (t = T)
Dense layer (Activation: Sigmoid)
Dropout = 0.5
Dense layer (Activation: ReLu)
Dropout = 0.5
Dense layer (Activation: ReLu)
LSTM
number of features (t = 0)
number of features (t = 1)
number of features (t = T - 1)
Size of timestep = T

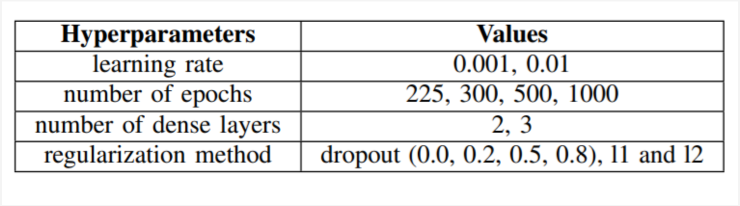
LSTM-based caching-Architecture 2: many-to-one

In this LSTM based prediction, the input is T hours history of request actions, and the output is the request of the next interval


number of features (t = T)
Dense layer (Activation: Sigmoid)
Dropout = 0.5
Dense layer (Activation: ReLu)
Dropout = 0.5
Dense layer (Activation: ReLu)
LSTM
LSTM
LSTM
number of features (t = 0)
number of features (t = 1)
number of features (t = T - 1)
Size of timestep = T

Majority vote-based caching

- Majority voting is an ensemble learning method in which multiple classifiers are used to predict the label based on the majority vote of the classifiers.
- We create 15 different LSTM models by altering hyperparameters.



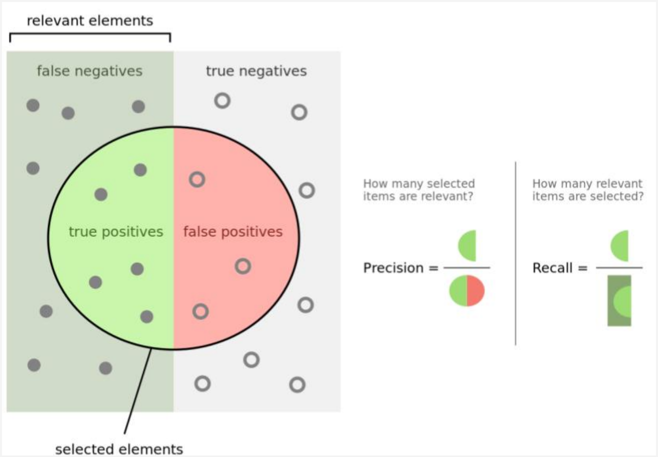
Hyperparameters	**Values**
learning rate	0.001, 0.01
number of epochs	225, 300, 500, 1000
number of dense layers	2, 3
regularization method	dropout (0.0, 0.2, 0.5, 0.8), l1 and l2

Baselines for comparing final scores

- Oracle: We assume that the algorithm can predict the requests with 100% accuracy. The final score that we can achieve with this algorithm is the highest score that we can have.
- Cache everything: Cache every possible experience, ensuring that every experience can be delivered with a delay$= 0$.
- Random Caching: We assume that we do not have any prior knowledge about the request, and thus we cache a randomly chosen request from the pool of possible requests.

Outline

Performance Metric: Prediction accuracy, precision, recall, F1-score



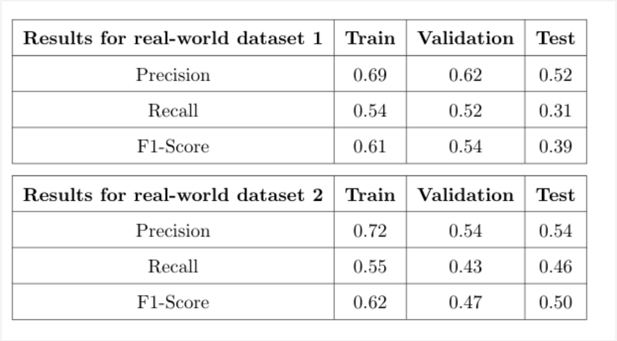
source: Wikipedia https://en.wikipedia.org/wiki/Precision_and_recall

$$F1\_Score = (2 * Precision * Recall)/(Precision + Recall)$$

Performance Metric: Prediction Accuracy with many-to-many architecture

Prediction accuracy is measured by the F1-score for training phase on different datasets. (80% training, 10% validation, 10% testing)

Real Datasets



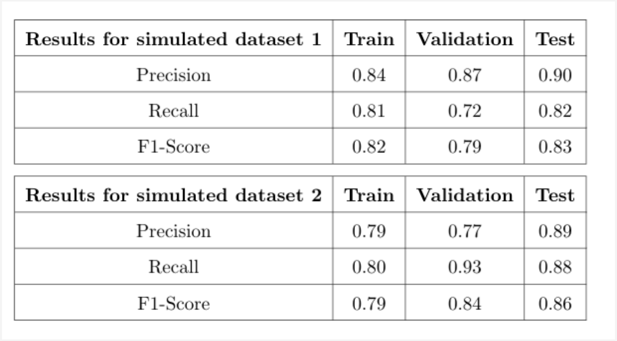
Results for real-world dataset 1	Train	Validation	Test
Precision	0.69	0.62	0.52
Recall	0.54	0.52	0.31
F1-Score	0.61	0.54	0.39

Results for real-world dataset 2	Train	Validation	Test
Precision	0.72	0.54	0.54
Recall	0.55	0.43	0.46
F1-Score	0.62	0.47	0.50

Performance Metric: Prediction Accuracy with many-to-many architecture

Prediction accuracy is measured by the F1-score for training phase on different datasets. (80% training, 10% validation, 10% testing)

Simulated Datasets



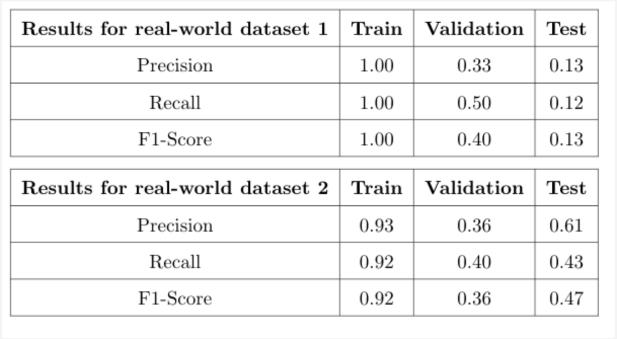
Results for simulated dataset 1	**Train**	**Validation**	**Test**
Precision	0.84	0.87	0.90
Recall	0.81	0.72	0.82
F1-Score	0.82	0.79	0.83

Results for simulated dataset 2	**Train**	**Validation**	**Test**
Precision	0.79	0.77	0.89
Recall	0.80	0.93	0.88
F1-Score	0.79	0.84	0.86

Performance Metric: Prediction Accuracy with many-to-one architecture

Prediction accuracy is measured by the F1-score for training phase on different datasets. (80% training, 10% validation, 10% testing)

Real Datasets



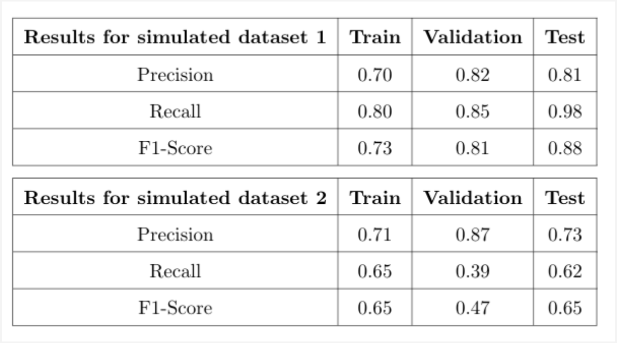
Results for real-world dataset 1	**Train**	**Validation**	**Test**
Precision	1.00	0.33	0.13
Recall	1.00	0.50	0.12
F1-Score	1.00	0.40	0.13

Results for real-world dataset 2	**Train**	**Validation**	**Test**
Precision	0.93	0.36	0.61
Recall	0.92	0.40	0.43
F1-Score	0.92	0.36	0.47

Performance Metric: Prediction Accuracy with many-to-one architecture

Prediction accuracy is measured by the F1-score for training phase on different datasets. (80% training, 10% validation, 10% testing)

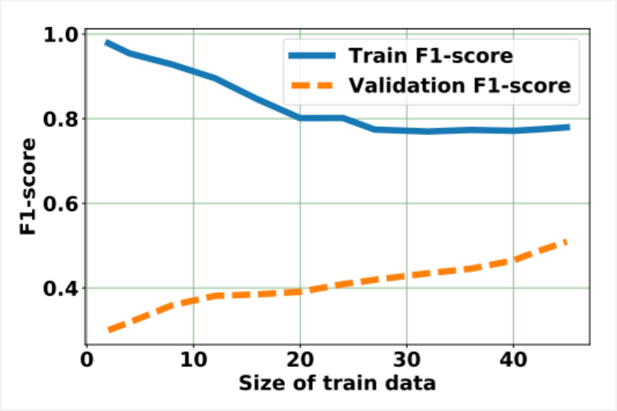
Simulated Datasets



Results for simulated dataset 1	Train	Validation	Test
Precision	0.70	0.82	0.81
Recall	0.80	0.85	0.98
F1-Score	0.73	0.81	0.88

Results for simulated dataset 2	Train	Validation	Test
Precision	0.71	0.87	0.73
Recall	0.65	0.39	0.62
F1-Score	0.65	0.47	0.65

Is size of the data really important?



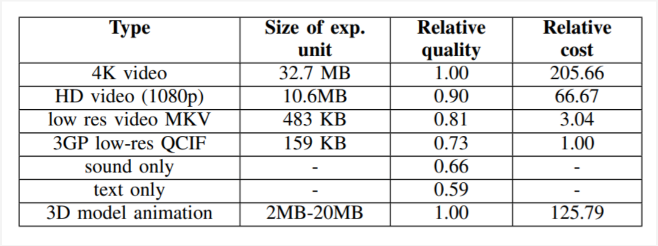
Performance Metric: Final Score

The final score combines the cost and user's satisfaction. The less latency that the agent has, the more satisfied the user would be. Also, the more optimized caching, the less cost the predictive model has.

$$final\_score = \alpha \cdot score - \beta \cdot cost$$

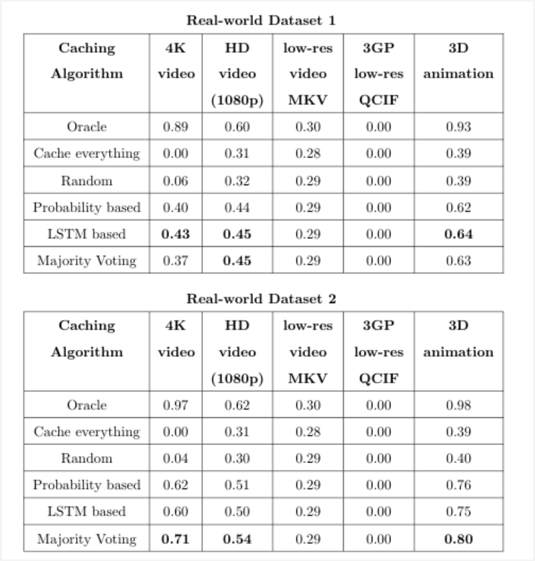
$$score(e_i) = d_d^{d(e_i)} \cdot d_f(f(e_i)) \cdot max\_score$$

$$d(e_i) = \frac{size\_of\_the\_content}{external\_bandwidth}$$



Type	Size of exp. unit	Relative quality	Relative cost
4K video	32.7 MB	1.00	205.66
HD video (1080p)	10.6MB	0.90	66.67
low res video MKV	483 KB	0.81	3.04
3GP low-res QCIF	159 KB	0.73	1.00
sound only	-	0.66	-
text only	-	0.59	-
3D model animation	2MB-20MB	1.00	125.79

Comparison Final Score - Real Datasets



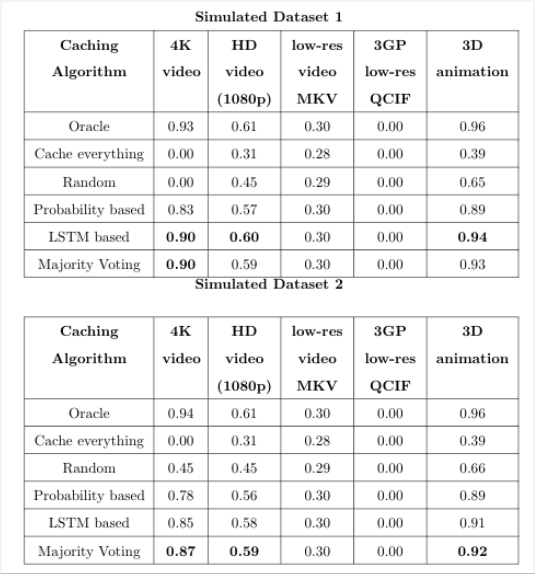
Real-world Dataset 1

Caching Algorithm	**4K video**	**HD video (1080p)**	**low-res video MKV**	**3GP low-res QCIF**	**3D animation**
Oracle	0.89	0.60	0.30	0.00	0.93
Cache everything	0.00	0.31	0.28	0.00	0.39
Random	0.06	0.32	0.29	0.00	0.39
Probability based	0.40	0.44	0.29	0.00	0.62
LSTM based	**0.43**	**0.45**	0.29	0.00	**0.64**
Majority Voting	0.37	**0.45**	0.29	0.00	0.63

Real-world Dataset 2

Caching Algorithm	**4K video**	**HD video (1080p)**	**low-res video MKV**	**3GP low-res QCIF**	**3D animation**
Oracle	0.97	0.62	0.30	0.00	0.98
Cache everything	0.00	0.31	0.28	0.00	0.39
Random	0.04	0.30	0.29	0.00	0.40
Probability based	0.62	0.51	0.29	0.00	0.76
LSTM based	0.60	0.50	0.29	0.00	0.75
Majority Voting	**0.71**	**0.54**	0.29	0.00	**0.80**

Comparison Final Score - Simulated Datasets



Simulated Dataset 1

Caching Algorithm	**4K video**	**HD video (1080p)**	**low-res video MKV**	**3GP low-res QCIF**	**3D animation**
Oracle	0.93	0.61	0.30	0.00	0.96
Cache everything	0.00	0.31	0.28	0.00	0.39
Random	0.00	0.45	0.29	0.00	0.65
Probability based	0.83	0.57	0.30	0.00	0.89
LSTM based	**0.90**	**0.60**	0.30	0.00	**0.94**
Majority Voting	**0.90**	0.59	0.30	0.00	0.93

Simulated Dataset 2

Caching Algorithm	**4K video**	**HD video (1080p)**	**low-res video MKV**	**3GP low-res QCIF**	**3D animation**
Oracle	0.94	0.61	0.30	0.00	0.96
Cache everything	0.00	0.31	0.28	0.00	0.39
Random	0.45	0.45	0.29	0.00	0.66
Probability based	0.78	0.56	0.30	0.00	0.89
LSTM based	0.85	0.58	0.30	0.00	0.91
Majority Voting	**0.87**	**0.59**	0.30	0.00	**0.92**

Resulting Conclusion

We find that LSTM-based and majority vote-based approaches outperform other approaches in order to optimize caching and have the maximum quality of delivery.

Outline

Conclusion and Future Work

- We proposed an approach to perform a local caching of AR/VR experiences for a household scenario.
- We investigated 3 different approaches:
  (1) Probability-based
  (2) LSTM-based approach
  (3) Majority voting over multiple LSTM networks.
- Majority voting over LSTMs approach yielded the performance that best balances the cost of caching with the perceived experience value.
- The performance could be significantly improved if more real-world data is available.

Acknowledgement

The support for this work was provided by the National Science Foundation under Award No. 1800961. Any opinions, findings, and conclusions and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.