ETH zürich

Digitec Galaxus’ Shopping Assistant

Data Science Lab - Fall 2018
Mélanie Bernhardt, Mélanie Gaillochet, Laura Manduchi

Goal

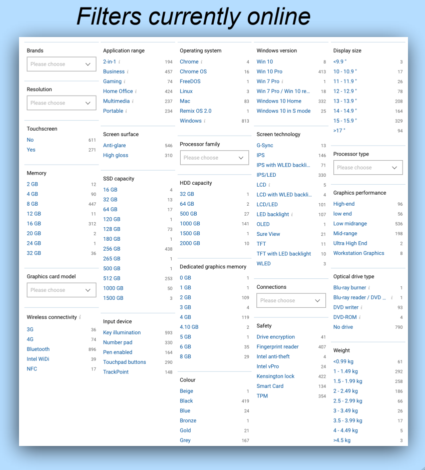
Currently, customers browsing on Galaxus Digitec’s website can filter out products by using up to 27 different filters. This is not very convenient for the user.

The goal of this project was to provide the most comfortable and efficient way for customers to find the product they need, by asking the **optimal sequence of questions** such that they are given a reduced subset of products as fast as possible.

From Digitec’s database, we had access to:

- **Product catalog**, with information on the products
- **Purchase table**, with history of past purchased products
- **Traffic table**, with all filters used by the user in a session

Filters currently online



Stage 1: Greedy Algorithm

<u>Purpose:</u> Obtain the most informative data (i.e. small subset of meaningful products) with the minimum amount of actions (i.e. questions asked).

Our greedy algorithm is inspired by [1], [2] and finds the optimal next question to ask by maximizing **Shannon's mutual information** between the question $\boldsymbol{Q_i}$ and the products $\boldsymbol{Y}$, given the history h_t of questions and answers provided until then.

$$
\begin{aligned}
Q_{opt[t]} &= \arg \max_{\boldsymbol{Q_i} \in setQ(t)} I(\boldsymbol{Q_i}; \boldsymbol{Y} \mid \boldsymbol{H_t} = h_t) \\
&= \arg \max_{\boldsymbol{Q_i} \in setQ(t)} \left[H(\boldsymbol{Y} \mid h_t) - H(\boldsymbol{Y} \mid \boldsymbol{Q_i}, h_t) \right] \\
&= \arg \max_{\boldsymbol{Q_i} \in setQ(t)} \left[-H(\boldsymbol{Y} \mid \boldsymbol{Q_i}, h_t) \right] \quad \text{since } H(\boldsymbol{Y} \mid h_t) \text{ is the same for all questions, for a given } \boldsymbol{Y} \\
&= \arg \max_{\boldsymbol{Q_i} \in setQ(t)} \left[-\sum_{a \in \boldsymbol{Q_i}} p\left(\boldsymbol{Q_i} = a \mid \boldsymbol{H_t} = h_t\right) \sum_{y \in Y} p(y \mid \boldsymbol{Q_i} = a, \boldsymbol{H_t} = h_t) \cdot \log\left(p\left(y \mid \boldsymbol{Q_i} = a, \boldsymbol{H_t} = h_t\right)\right) \right]
\end{aligned}
$$

Some questions available in the database are very specific $\longrightarrow$ High probability of asking questions that a real user would not be able to answer (e.g. "model of computer’s battery").

Hence, we introduced a **prior on filters**, and we gave more weight to questions that had previously been used by users (from the real historical data).
This made our algorithm more user-friendly.

Notations:

$\boldsymbol{Y}$	*final product*
$\boldsymbol{Q_i}$	*question n°i (possible answers are its realizations)*
$setQ(t)$	*set of available questions at timestep t*
$\boldsymbol{H_t}$	*history of questions and answers asked before timestep t.*
$H(\boldsymbol{Y} \mid h_t)$	*entropy of $\boldsymbol{Y}$ given the history.*
$H(\boldsymbol{Y} \mid \boldsymbol{Q_i}, h_t)$	*conditional entropy between $\boldsymbol{Q_i}$ and $\boldsymbol{Y}$ given the history.*

Interestingly, this method also gave **optimal results** in **various noisy settings**.

Stage 2: Imitation Learning

<u>Purpose</u>: **Increase the computational speed** of our system, by training a deep neural network (DNN) to imitate the policy of our greedy algorithm (“teacher”) for finding the next question.

We implemented the **DAgger algorithm** [3], that first pre-trains a DNN with an initial dataset, generated from our teacher. It then runs its newly learned policy to explore new states. For each one, the pair [state, q], where q is the next question predicted by the teacher, is added to the initial dataset to retrain the model iteratively.

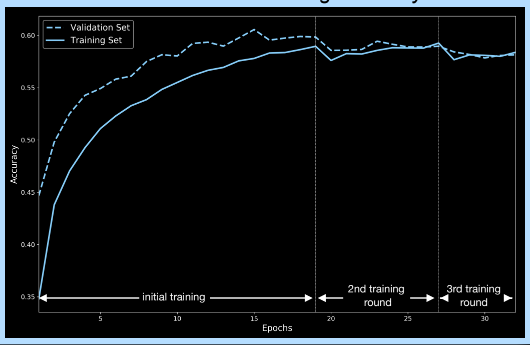
Since we were able to generate enough initial training data (> 70,000 states and labels), DAgger could not significantly improve **classical imitation learning** (i.e. there is no exploration of new states after initial training).

Therefore, to avoid overfitting, we decided to only use the model trained during initial training.

Algorithm:

1) Collect the initial training data, D_0, in the form of [state, next question] by running the Greedy Algorithm on 7000 products. States are defined as the collection of all previously asked questions and their corresponding answers.
2) Train the DNN on D_0 to get the first policy.

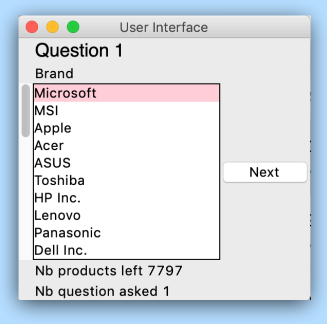
Validation and training accuracy



Stage 3: User interface

The interactive interface we created uses our model to compute the next question that is asked to the user. It keeps asking questions until the number of products left is less than a predetermined threshold (50 products). Finally it returns the reduced list of products to choose from.

Screenshot from our interface



User Interface
Question 1
Brand
Microsoft
MSI
Apple
Acer
ASUS
Toshiba
HP Inc.
Lenovo
Panasonic
Dell Inc.
Next
Nb products left 7797
Nb question asked 1

Results

The evaluation of our algorithm’s performance is based on the **total number of questions asked** (Q) to output a subset of less that 50 products. We compare our algorithm’s performance with a **random baseline** (where the next question is selected uniformly at random among the questions not already asked).

To add **robustness** to our model, we used several noise settings that simulated different kinds of users.

Evaluation procedure:

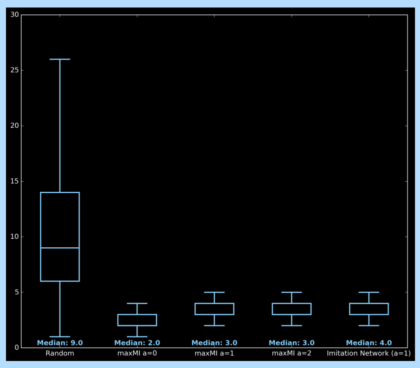
For a predetermined number of iterations:
- Sample one product (representing the target product of our fake user)
- Sample the answers to all possible questions according to the chosen noise setting
- Run the Greedy or Imitation algorithm to get the predicted list of questions to ask to the simulated user.

For all the noise settings both our algorithms **outperformed** the random baseline and the median of Q is mostly 3 times lower than the one of our baseline.

Evaluation of the number of question asked per user:

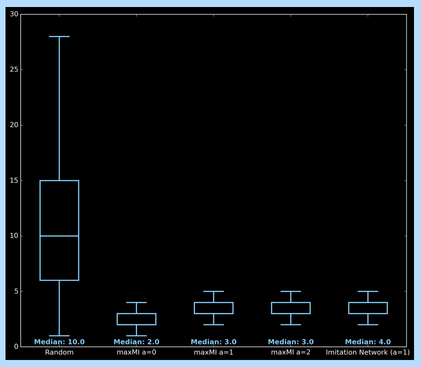
Test set size: 1000 simulated users. Outliers not shown.

<u>No noise</u>
Additional P('I don't know')= 0
P(2 answers)=0
P(3 answers)=0

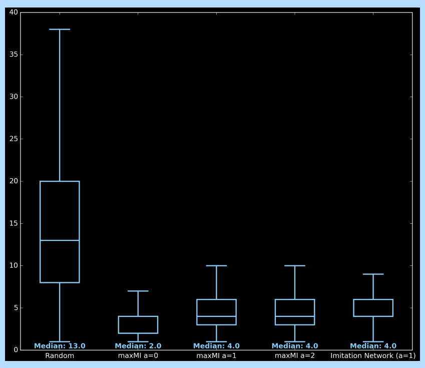


<u>Setting 1 (medium noise)</u>
Additional P('I don't know')= 0
P(2 answers)=0.2
P(3 answers)=0.1

We can observe that the results are nearly identical to those obtained in the non-noisy setting.



<u>Setting 2 (very noisy)</u>
Additional P('I don't know')= 0.1
P(2 answers)=0.4
P(3 answers)=0.3



References

[1] Chen, Y. & Hassani H. S. & Karbasi, A. & Krause, A. (2015) Sequential Information Maximization: When is Greedy Near-optimal?. JMLR: Workshop and Conference Proceedings,40:1–26.
[2] Chen, Y, & Hassani H. S. & Karbasi, A. & Krause, A. (2017) Near-optimal Bayesian active learning with correlated and noisy tests. Electronic Journal of Statistics,11:4969–5017.
[3] Ross, S. & Gordon, G. J. & Bagnell J. A. (2011) No-regret reductions for imitation learning and structured prediction. CoRR, bs/1011.0686, 2010.