Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis Business Analytics and Quantitative Marketing

Introducing Probabilistic Neural Network based models for advertisement conversion prediction and attribution modelling

Student: (Katherine) K.Y.M. de Kruiff, 444677

Supervisor: dr. (Kathrin) K. Gruber

Second assessor: (Eoghan) E.P. O'Neill

Date final version: November 3, 2021

Abstract

Advertisers are constantly optimizing their online marketing strategy and ad buying decisions. At the same time, ad exchanges aim to optimize their ad pricing scheme. To do so, advertisers and ad exchanges require meaningful insights which have to be generated based on the browsing behavior of the Internet user. The current literature employs attribution models to generate data-driven insights and to assign credits to advertisements based on their contribution to a conversion. This research introduces a new model to the current literature, the Attention Probabilistic Neural Network (APNN), beneficial for both advertisers and ad exchanges. The APNN is simple to implement and avoids long training times compared to the existing models and enables attribution modelling by making use of the relation between individual advertisements. Based on the newly introduced APNN, this research also introduces the similar point APNN which optimizes the use of relevant information to further improve model accuracy. To evaluate the performance of the different models, a dataset of the online advertising research company Criteo is used. It follows that the newly introduced (similar point) APNNs are easy to implement and interpret while having compatible model performance with existing neural networks which employ attention mechanisms and sequential modelling for attribution modelling and conversion prediction.

The content of this thesis is the sole responsibility of the author and does not reflect the view of the supervisor, second assessor, Erasmus School of Economics or Erasmus University.

Contents

1 Introduction

With the increase of online shopping, the online customer journey is becoming more advanced and more important for advertisers and ad exchanges. Every day customers are targeted with personal advertisements through different channels, e.g., e-mail, social media, search engines or mobile platforms. The browsing behavior of the customers and the data generated by the response to the advertisements allow advertisers and ad exchanges to generate meaningful insights in their marketing strategy and ad pricing scheme respectively. However, it remains a challenge to identify which advertisements contributed most to the final purchase, also referred to as conversion.

The customer journey and browsing behavior of an Internet user typically consist of multiple advertisement touchpoints, also referred to as impressions. Measuring the influence of each touchpoint on the conversion is referred to as the attribution problem (Arava, Dong, Yan, Pani, et al., 2018). For advertisers it is very important to allocate the right conversion credit to each touchpoint along the customer journey of an Internet user for two main reasons; *i*) knowing the contribution of each touchpoint along the customer journey to bringing in sales enables advertisers to make informed decisions on the buying behavior of customers (Lee, Orten, Dasdan, & Li, 2012), and *ii*) a reliable attribution model uncovers the contributions from all relevant advertisements giving insights on which advertisements advertisers should focus to increase sales while minimizing advertising costs (Geyik, Saxena, & Dasdan, 2014). Also ad exchanges benefit from a reliable attribution model as it can be used to optimize the pricing scheme of advertisements between advertisers and publishers and thus maximize profit.

At first the attribution problem was addressed by applying different rule-based heuristics, like the last- or linear-touch rules, where credits are given to the last touchpoint or equally distributed over all touchpoints of the customer journey respectively (J. Wang, Zhang, & Yuan, 2016). These rule-based heuristics are easy to employ, but don't rely on data support. Therefore, several other, data-based approaches are proposed in the literature of attribution modelling to statistically learn the attributions of the different touchpoints ranging from simple probabilistic to advanced methods (Shao & Li, 2011; Anderl, Becker, Von Wangenheim, & Schumann, 2016; Dalessandro, Perlich, Stitelman, & Provost, 2012; Xu et al., 2016; Li & Kannan, 2014). Due to the increase in data availability ad exchanges often address the attribution problem using machine learning algorithms (Arava et al., 2018; Ren et al., 2018; Du, Zhong, Nair, Cui, & Shou, 2019). However, these methods often consist of models with multiple layers making interpretation of the model difficult and apply sequential modelling which leads to long training times. For both advertisers and ad exchanges interpretability of the model is important. If a model is well-understood, this will enhance its usage, which will lead to more effective advertising. Further, although real-time use of the model is often not necessary, it may be beneficial for advertisers to avoid long training times of the model in order to quickly generate insights in newly developed advertisements.

In order to increase interpretability and improve the training time of attribution and advertising conversion models for both advertisers and ad exchanges, this research examines the performance of a simple Probabilistic Neural Network (PNN) model in combination with an attention mechanism (APNN) to model the attribution of the different touchpoints. In addition, the similar point APNN is introduced based on the principle of k-nearest neighbors to increase the accuracy of conversion prediction for advertisers. In the literature a PNN has widely been used in classification and pattern recognition problems, but has not yet been examined in the field of advertising conversion in combination with an attention layer. The attribution and conversion prediction of the different touchpoints is based on the cosine similarity between the touchpoints. Further, the (A)PNNs are able to deal with the imbalanced nature of the data, which is normally a challenge in modelling advertising conversions as the data often contains very few conversions, which makes conversions appear like an 'outlier' and hard to recognize. To deal with the imbalanced nature of the data for sequential modelling, sampling techniques are used which often result in ignoring part of the dataset or making assumptions which might be invalid. In contrast, due to the structure of a PNN based model, an imbalanced nature of the data is no issue and all available data can be used.

Thus, in order to contribute in the field of advertising conversion, this research aims to answer the following research questions: *i) How does one employ a PNN with an attention* *layer (APNN)? ii) How does an APNN perform compared to the benchmark models, like for example the Recurrent Neural Network (RNN) with attention mechanism?* and *iii) How can one interpret the attribution allocation of the APNN?* To answer these research questions, part of the dataset of the online advertising company Criteo is used. The complete dataset consists of 16.5 million touchpoints, resulting in 6.8 million user sequences from which approximately 550,000 led to a conversion.

Empirical results show that the similar point APNN using the most similar touchpoint for predicting the convergence of a sequence, is the best performing model of the standard PNN and APNNs based on accuracy and F-score. As the similar point APNN is based on cosine similarity, its good performance is mainly due to the usage of the most relevant information for conversion prediction and ignoring touchpoints which are not informative. However, the similar point APNN is not able to model the attribution of different touchpoints and is therefore not of great interest for ad exchanges. To include attribution modelling, the standard APNN is evaluated which is able to generate (quick) attribution results based on the relation between individual touchpoints. Hence, it uses a different approach than the advanced deep neural networks with an attention mechanism using sequence-to-sequence modelling. The main advantage of the standard PNN and APNNs is the ease of implementation and interpretation and the avoidance of long training times of the models. Hence, for an accurate, simple and easy interpretable model for conversion prediction, advertisers should be referred to the similar point APNN. For attribution modelling, both advertisers and ad exchanges have to decide which attribution approach is preferred, the APNN or a neural network, and have to weigh this against the importance of interpretability and training time of the model.

The remainder of this research is organized as follows. First, Section 2 gives an overview of the relevant literature. Second, Section 3 describes the data used to answer the research questions. Third, Section 4 describes the benchmark models and lays out the details of the standard PNN and APNNs. Fourth, Section 5 gives an overview of the results of the applied methods. Finally, Section 6 summarizes the conclusions, answers the research questions and suggests topics for further research.

2 Literature Review

The literature review consists of three main parts. First, Section 2.1 gives an introduction to attribution modelling and reviews several methods used to model touchpoint attribution. Second, Section 2.2 describes neural network based models using an attention mechanism for attribution modelling. Lastly, Section 2.3 gives an overview of the metrics used to evaluate models for imbalanced data.

2.1 Attribution Modelling

Advertisers invest in marketing campaigns and advertisements to draw in customers to purchase their product through their website, mobile app, and/or store, and thus increase sales. Hence, customers are typically exposed to all types of advertisements in their path to a purchase. Therefore, attribution modelling is used to determine the appropriate credit for each touchpoint on a user's path to a purchase, which is possibly distributed over different online, mobile, and offline channels. Furthermore, insights on the overall attribution of different channels can be generated by aggregating the individual attributions of the touchpoints per channel. Thus, by making use of attribution modelling several questions can be answered, for example *i*) Which advertisements drive conversions most? *ii*) Which advertising channels are most effective? *iii*) Does the impact of the advertisement differ between devices? and many more (Kannan, Reinartz, & Verhoef, 2016).

Thus, on the one hand attribution modelling is used by advertisers to efficiently (re)allocate their advertising budget and make informed ad buying decisions (Geyik et al., 2014). On the other hand, attribution modelling is used by ad exchanges to optimize the pricing scheme between advertisers and publishers. To do so several attribution methods are proposed in the literature of attribution modelling which can be divided in three main categories; *i*) Rule-based heuristics, *ii*) Data-driven algorithmic models, and *iii*) Machine Learning models.

2.1.1 Rule-based Attribution Models

First, rule-based attribution models were developed which rely on a set of rules to attribute credits to specific touchpoints. These models are simple, straightforward and widely adopted in the literature of attribution modelling. Amongst others, examples of such models are the last-touch, first-touch, Time Decay and linear-touch rule model. The first- and last-touch rule model assign all the credits to the first and last touchpoint of a user sequence respectively preceding a conversion thereby ignoring the effects of other touchpoints. The linear-touch rule model distributes the credits evenly over all touchpoints and hence assumes equal contribution of all touchpoints to the conversion. Further, the Time-Decay attribution model assumes that the credit of a touchpoint decays based on the decay parameter which is based on intuition, but not supported by the data itself. An extensive literature review of other examples of rule-based attribution models is given by Kee (2012).

The rules of the rule-based attribution models are not based on the data meaning that the attributions are determined without taking into consideration whether an advertisement in the customer journey led to a conversion. Therefore, these models are far from optimal and easily fooled by advertisers in order to gain credits and not considered in this research.

2.1.2 Data-driven Algorithmic Models

Second, in order to overcome the drawbacks of rule-based heuristics and attribute credits to the different touchpoints based on the data, data-driven models were developed taking into account the outcome of the customer journey of an Internet user, conversion yes or no. This is also referred to as Multi-Touch Attribution (MTA) as these models aim to track the behavior of a user after being exposed to advertisements of different channels. In the literature of attribution modelling often the simple Logistic Regression (LR) is used which allows for a direct interpretation of the relationship between a touchpoint and the probability of a conversion which will be explained in more detail in Section 4.1.1. Further, Shao and Li (2011) proposed a bagged LR method where they use weights to distribute the credits for attribution modelling and compare it to a probabilistic model.

Many other models and techniques based on probabilistics are employed in the literature of attribution modelling. For example, Zhang, Wei, and Ren (2014) introduce the Additive Hazard (AH) model which uses survival analysis and an additive hazard function to predict conversion while taking into consideration the time of a touchpoint. Similarly, Ji, Wang, and Zhang (2016) propose the Additional Multi-touch Attribution (AMTA) model which is also based on survival analysis to model conversion and makes use of the hazard rate of conversion at a specific time to model the conversion attribution of the different touchpoints. However, in general probabilistic models have two main limitations; *i*) they rely on distributional assumptions which may be invalid, and *ii*) as the data availability increases tremendously in size and complexity, probabilistic models need to rely on approximations to make the attribution problem computationally feasible, e.g., browsing data has to be trimmed back. Amongst others, Ren et al. (2018) evaluated the performance of the bagged LR, AH and AMTA model and their results will be used as a benchmark for the newly developed model introduced in this research.

Another example of a data-driven model which aims to fairly distribute credits across advertisements is Shapley Value, a concept based on cooperative game theory (Shapley, 1953). Shapley Value evaluates the marginal contribution of each player in a game, where in case of attribution modelling for advertisements the different channels are seen as players and the marketing strategies as the cooperative game. Hence, the channels aim to convert users by cooperatively attract and influence users. Then, the attribution is computed based on the contribution of the channel to the conversion (Zhao, Mahboobi, & Bagheri, 2018). The computed attribution of a channel gives an indication of the attribution of the advertisements belonging to this channel. Hence, a disadvantage of the Shapley Value algorithm is that it requires the definition of a limited number of advertisement/touchpoint labels (e.g., channels) in order to be computationally tractable and that touchpoints have to be grouped according to broad definitions which might cause a loss of valuable information. Hence, Shapley Value is not considered in this research.

2.1.3 Machine Learning Attribution - Attention Mechanisms

To deal with the increase in data availability, ad exchanges more often rely on machine learning techniques for attribution modelling. Where machine learning initially was mainly employed to translate a sequence of words from one language to another or image and speech recognition, new approaches have been proposed to derive conversion attributions (Arava et al., 2018; Ren et al., 2018; Du et al., 2019). These approaches often rely on neural networks which make use of sequential modelling and are trained on one sequence at the time. In order to attribute credits to the different touchpoints based on their contribution to the likelihood of a conversion, attention mechanisms are used.

Attention mechanisms were introduced to help neural networks focus on the important parts of the input data. An attention function can be seen as mapping a query and a set of key-value pairs to an output, where the query, keys, values and output are all vectors. Then, the output is computed as a weighted sum of the values, where the weights are computed by a compatibility function of the query with the corresponding key. In terms of attribution modelling for advertising, the input consists of a user sequence consisting of different touchpoints which are then summarized into the output vector. Figure 1 shows an example of a Location-Base attention mechanism, where $\mathbf{x}_j$ reflects the input, (query, keys and values), in this case the sequence of touchpoints of a user, h_j the hidden states, f the feed-forward network, after which the compatibility function, here the softmax, is applied to compute the final weight scores, a_i, which reflect the attention given to each hidden state. Using the attention weights and the outcomes of the hidden states the output $\mathbf{c}$ is computed. The output vector $\mathbf{c}$ reflects the context vector of all the sequences of a user capturing, for example, the different patterns of advertisements being clicked. Using the context vector, the final conversion rate of a user sequence is predicted.

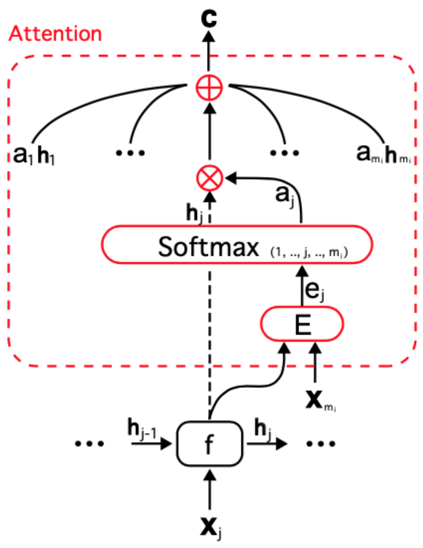

Figure 1: Location-Base Attention Mechanism. Source: Ren et al. (2018)

The two most common attention mechanisms are additive and (scaled) dot-product attention (Bahdanau, Cho, & Bengio, 2014; Vaswani et al., 2017). In the additive attention model a vector is created as a weighted sum of the hidden states, where the weights reflect the attribution given to each state in a sequence. To do so the additive attention uses a feed-forward network with a single hidden layer to compute the compatibility function. Compared to the additive attention, the (scaled) dot-product attention is faster and more space-efficient as it makes use of a compatibility function which is equal to the dot-product between the hidden and forward state representation allowing for the use of highly optimized matrix multiplication code. By scaling the dot-product for the dimension of the keys, Vaswani et al. (2017) suggest that the scaled dot-product attention results in at least the same or better performance than the additive attention mechanism. However, due to the multiplicative nature of the scaled dot-product attention issues of numerical under- and/or overflow can occur.

Besides additive and (scaled) dot-product attention, common attention mechanisms are the Content-Base and Location-Base attention mechanisms which depend on a *cosine* and $softmax$ compatibility function respectively for computation of the attention scores. The Content-Base attention mechanism computes the weight of the target hidden state based on a comparison with each of the previous hidden states for which Graves, Wayne, and Danihelka (2014) use the cosine similarity. Then the weights are determined by normalizing these similarities and used for the computation of the context vector. The Location-Base attention mechanism determines the weights merely based on the target hidden state. Hence, the context vector of the Location-Base attention mechanism is computed as a weighted average over all hidden states, as also described above and shown in Figure 1 (Luong, Pham, & Manning, 2015).

2.2 Attention based Attribution Models

Besides the bagged LR and probabilistic based models for attribution modelling mentioned in Section 2.1.2, there exist models which have implemented the above mentioned attention mechanisms in a neural network over a wide variety of industries. An example of a basic model in the field of advertising conversion is the Attention Recurrent Neural Network (ARNN) which employs a single Location-Base attention mechanism to model impression-level patterns when determining the conversion attribution. As an extension to the ARNN, Ren et al. (2018) developed the Dual-Attention Recurrent Neural Network (DARNN) which not merely models impressions-level patterns, but also models the click-level through sequence-to-sequence modelling using a Location-Base attention mechanism. Hence, it aims to capture the sequential user behavior patterns and learns the optimal attentions using sequential modelling, while taking into consideration different behavior types. Further, Arava et al. (2018) proposed the Deep Neural Net With Attention multi-touch attribution model (DNAMTA) which also makes use of sequential modelling, but at the same time includes time-decay functions and static user information in combination with dynamic touchpoint observations.

However, all the above attention based attribution models are multiple layered models, often built around neural networks and therefore relatively difficult to interpret for advertisers and ad exchanges making them inaccessible for usage. Besides, many of these models make use of sequential modelling which is computationally expensive. Also, a wide variety of neural networks like RNN's require a long training time to train the model. To avoid these limitations, Vaswani et al. (2017) for example introduce the Transformer which is a simple network architecture solely based on attention mechanisms. For the same reasons, this research proposes a classification method based on established statistical principles, the PNN, and aims to combine the idea of attention with the PNN into a newly developed model, referred to as APNN. For comparison, amongst others, the ARNN and DARNN are used as benchmark models for which results are provided by Ren et al. (2018).

2.3 Performance Measures for Imbalanced Data

Besides the difficulties of finding and improving approaches or models to model advertising conversion using imbalanced data, finding the appropriate evaluation measure for model assessment is important when dealing with imbalanced data (Bekkar, Djemaa, & Alitouche, 2013). The literature of evaluation measures consists of several types; *i*) measures based on the confusion matrix, *ii*) combined measures of the confusion matrix, *iii*) graphical performance evaluation measures, and *iv*) probabilistic measures.

In machine learning classifiers are mostly evaluated using the confusion matrix in which the number of True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN) are reported. From the confusion matrix, several evaluation measures can be constructed, like accuracy, error rate, sensitivity (recall), specificity, and precision. As accuracy gives equal weight to TP and TN it is often not considered as performance metric for imbalanced data. However, as both the correct prediction of converting and non-converting advertisements is important, this research considers accuracy as a performance metric to evaluate the models. Besides accuracy, additional evaluation measures, like precision and recall, were developed as these are independent of the data distribution and used in case the prediction of one of the classes is more important (Bekkar et al., 2013).

In the literature of model evaluation many combinations of the above mentioned evaluation measures have been proposed, like G-means, likelihood ratios, discriminant power, F-(beta)measure, balanced accuracy, Youden index and Matthews correlation coefficient. The usage of these combination measures often depends on the application and goal of the study. For example, the likelihood ratios are mainly used in medical diagnosis predictions (Delacour, François, Servonnet, Gentile, & Roche, 2009). In this research the F-measure will be considered in combination with the accuracy as it is often employed in the field of fraud detection, which is comparable to the rare event of an advertising conversion (Di Martino, Decia, Molinelli, & Fernández, 2012; Bekkar et al., 2013). Also, the F-measure puts more emphasis on the minority class making the metric useful for imbalanced data.

Further, one can use graphical performance evaluation like the Receiver Operating Characteristics (ROC) curve. For this curve it holds that the more the curve is toward the upper left corner, the better the ability of the classifier to distinct between the minority and majority class. However, there are some disadvantages when evaluating models with imbalanced data using the ROC curve (Drummond & Holte, 2000), e.g., the difficulty to compare models when two ROC curves intersect. Hence, the area under the ROC curve (AUC) measure is considered which is a summary indicator of the pairwise ranking performance of the ROC curve and therefore has no relationship to the direct output value of the model. Thus, the AUC is able to order models by overall performance and is therefore preferred in model evaluation (Batista, Prati, & Monard, 2004). Ren et al. (2018) have also used the AUC to compare their models which are used as a benchmark in this research.

Lastly, if besides the incorrect vs. correct class prediction one is also interested in the uncertainty that a model has in predicting the classes, probabilistic metrics should be considered, like log-loss also known as Cross-Entropy or the Brier score. Often in case of binary classification problems and as seen in research with similar settings the log-loss is used as a performance metric (Ren et al., 2018). Hence, the log-loss will be considered as evaluation metric for some of the models evaluated in this research which will be further explained in Section 4.

3 Data Description

To answer the research questions stated in the introduction, a dataset of Criteo is used. Criteo is a company in online advertising research which merely sells advertisements via display and which published a dataset for attribution modelling in real-time auction-based advertising (Diemert, Meynet, Galland, & Lefortier, 2017). With this dataset Criteo is able to analyze online user browsing behavior and to target different segment groups of users with their advertisements. The dataset contains live traffic data of Criteo over 30 days consisting of user information after displaying advertisements/impressions. For every advertisement shown to a user, the dataset contains the relative timestamp, if the advertisement was clicked and led to a conversion, the unique user and campaign ID, the timestamp of the conversion if applicable, the number of clicks, the time since last click for the given advertisement, and a category ID associated with contextual features of the advertisement. The 59,098 unique category IDs reflect amongst others the different communication platforms and formats of the advertisement (e.g., video or audio). Further, the dataset contains information on the price paid by Criteo for displaying the advertisement and the cost per order in case of an attributed conversion to Criteo. Table 1 shows the relevant data used in this research which consists of whether an advertisement is clicked in combination with its contextual features.

The dataset consists of 16.5 million impressions over a range of 675 campaigns resulting in 6.1 million sequences from which 550,000 resulted in a conversion. In order to use the data some pre-processing and cleaning has to be done. Given the original data and the activity of the users, sequences per user and conversion can be constructed based on the process of Ren et al. (2018). If a user has multiple conversions, the sequence will be split based on the conversion time to construct separate sequences for all (non-)conversions. As shown in Figure 2, the dataset contains many sequences consisting of one impression. However, as this data is used to segment online users based on their browsing behavior, the sequences consisting of one touchpoint can be omitted as one cannot identify browsing behavior based on one touchpoint. After omitting sequences with one touchpoint, it follows that the dataset has a conversion ratio of 4.34%. The training and test set contain 80% and 20% of the sequences respectively while stratifying the conversion ratio in the training and test set. For the benchmark models, Ren et al. (2018) applied an undersampling technique in order to deal with the imbalanced nature of the data and to retrieve a 50-50 split of converting and non-converting sequences. For the PNN based model undersampling is not required and all data can be used, which is an advantage as no information is lost.

Figure 2 shows the frequency with which sequence lengths occur in the dataset for converted and non-converted sequences. For illustration purposes sequences with more than 30 impressions are excluded in this figure, corresponding to 0.1% of the total number of sequences. Further, Table 2 gives descriptive statistics about the data before and after removing the sequences containing one impression. It shows that the number of impressions in case of a converted sequence is lower than for non-converted sequences.

For evaluation and comparison of the benchmark models with the newly introduced standard PNN and different APNNs, only five days of the Criteo dataset, resulting in 2.75 million touchpoints, are used for computation and memory purposes. In this way enough data is used to get an indication of the performance of the standard PNN and different APNNs while saving computation time and avoiding memory issues. The conversion ratio of this partial dataset is 4.1%.

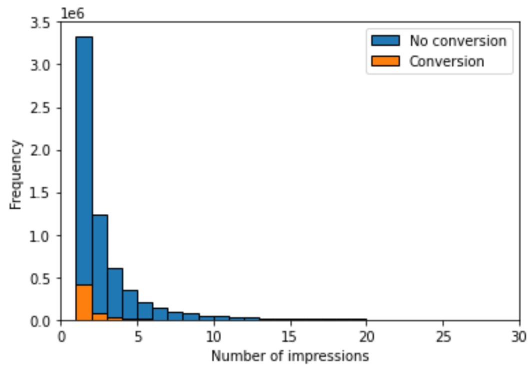

Figure 2: Number of impressions per sequence. Sequences with more than 30 impressions (0.1%) are excluded for illustration purposes.

Table 1 *Relevant information per touchpoint. In total there are 59,098 unique category IDs.*

Input	Type	# of options
Clicked	Binary	0 or 1
Campaign ID	Categorical	675
Category 1	Categorical	9
Category 2	Categorical	70
Category 3	Categorical	1,829
Category 4	Categorical	21
Category 5	Categorical	51
Category 6	Categorical	30
Category 7	Categorical	57,196
Categroy 8	Categorical	11
Category 9	Categorical	30

Table 2 *Descriptive statistics of converting and non-converting sequences. Sequences consisting of one impression are not considered as they are not informative in segmenting user behavior.*

	Min.	25% Quantile	Median	Mean	75% Quantile	Max.
Before cleaning						
Length converting seq.	1	1	1	1.467	1	81
Length non-converting seq.	1	1	1	2.486	3	880
After cleaning						
Length converting seq.	2	2	2	2.907	3	81
Length non-converting seq.	2	2	3	4.157	5	880

4 Methodology

The methodology can be split into three main parts. First, Section 4.1 gives a description of the benchmark models evaluated by Ren et al. (2018). Second, 4.2 describes the standard PNN and the newly introduced PNN with an attention layer, referred to as APNN. Lastly, Section 4.3 describes the evaluation metrics which are used to evaluate the different models. Ren et al. (2018) implemented all models in Python and trained all the deep models separately over one NVIDIA GeForce GTX 1080 Ti wit Intel Core i7 processor for five hours. The standard PNN and APNNs are run using the Lisa system, which is a cluster computer consisting of several hundreds of multi-core nodes running the Linux operating system.

4.1 Benchmark Models

The models evaluated by Ren et al. (2018) are used as a benchmark for the standard PNN and APNNs. Ren et al. (2018) use four baseline models to compare to the ARNN and their newly introduced DARNN. In order to understand these models Section 4.1.1 gives a description of the bagged LR and SP model, Section 4.1.2 describes the AH and AMTA model, and lastly, Section 4.1.3 describes the ARNN and DARNN.

4.1.1 Bagged Logistic Regression and Simple Probabilistic Model

First, the bagged LR model is used as a benchmark model which was introduced by Shao and Li (2011). Shao and Li (2011) combine the LR model with the idea of bagging which enables the model to reduce the estimation variability which is caused by highly correlated covariates between channels. The bagged LR model results in an easy interpretable, simple logistic model and generates stable and reproducible estimation results. The model consists of two main steps

Step 1: Sample a proportion of all sample observations and covariates for the given dataset, fit a LR model on the sampled covariates and sampled data, and save the estimated coefficients.

Step 2: Repeat Step 1 M number of times, after which the final coefficient estimates are computed by averaging over the M estimated coefficients of each iteration.

Shao and Li (2011) have shown that the bagged LR model achieves a similar missclassification rate as the standard LR model. Further, the attribution modelling is based on the coefficients of the resulting bagged LR model which indicate the contribution of each of the channels to the conversion. The variability of the coefficients of the bagged LR model is much smaller than the coefficients of the standard LR model which is preferred for attribution modelling in order to fairly distribute credits amongst advertisers. Hence, using the stable results of the bagged LR model advertisers can develop a clear strategy to optimize their resource allocations and advertisers can optimize their ad pricing scheme accordingly.

Second, an SP model is considered which is based on the idea of Dalessandro et al. (2012). The SP achieves low estimation variability, is simpler than the (bagged) LR model making the model easy to interpret, but loses accuracy. In the SP model, Ren et al. (2018) compute the conversion rate of each user sequence the same way as done in Zhang et al. (2014), where

$$p(y=1|\{c_j\}_{j=1}^{m_i}) = 1 - \prod_{j}^{m_i}(1-\Pr(y=1|c_j=k)), \tag{1}$$

where $\Pr(y = 1|c_j = k)$ is the conversion probability from the observed data for the k-th channel and m_i the total browsing activities of a user. Then, using the structure as described in Shao and Li (2011) who use a combination of first and second-order conditional probabilities, the contribution of each channel and/or touchpoint towards the conversion can be computed in two steps

Step 1: First, the conversion probability and pairwise conditional probability are computed as

$$P(y|x_k) = \frac{N_{y=1}(x_k)}{N_{y=1}(x_k) + N_{y=0}(x_k)}, \tag{2}$$

and

$$P(y|x_k, x_l) = \frac{N_{y=1}(x_k, x_l)}{N_{y=1}(x_k, x_l) + N_{y=0}(x_k, x_l)}, \tag{3}$$

respectively, for $k \neq l$, where y is the binary outcome, conversion yes or no, and $x_k, k = 1, ..., p$, the p different advertising channels. Then, $N_{y=1}(x_k)$ and $N_{y=0}(x_k)$ reflect the number of users with and without conversion respectively exposed to channel k, and $N_{y=1}(x_k, x_l)$ and $N_{y=0}(x_k, x_l)$ reflect the number of users with and without conversion respectively exposed to both channel k and l.

Step 2: Then, the contribution of channel k is computed for each converted user as

$$C(x_k) = p(y|x_k) + \frac{1}{2N_{k \neq l}} \sum_{i \neq j} \Big\{ p(y|x_i, x_j) - p(y|x_i) - p(y|x_j) \Big\}, \tag{4}$$

where $N_{k \neq l}$ reflects the number of channels (l's) not equal to k (N-1).

Hence, attribution modelling using the SP model is done by computing the different conversion probabilities for the different channels which has an even more direct interpretation than the results of the bagged LR model. Due to the overlapping influences of different touchpoints through similarly designed advertisements/channels and a user's exposure to multiple media channels the second-order probability is included. Hence, the second-order probability estimation can be seen as the joint effect of all possible combinations of two channels. Higher order probabilities can also be included, but the number of observations which contain overlapping higher order interactions often drops significantly, even for very large datasets (Shao & Li, 2011).

4.1.2 Additive Hazard and Additional Multi-touch Attribution Model

Third, Ren et al. (2018) evaluate the performance of the AH model from Zhang et al. (2014). Besides the strength of the impact of different advertisements and channels, the AH model is able to incorporate the differences in time-decaying speed of advertisements. This is done by using a hazard function with a set of additive exponential kernel functions, which are assumed to each reflect the dynamics of the influences of an advertisement channel on the conversion. The hazard function is considered to be additive on the clicking of the different advertisement channels and is specified as follows for user u

$$\lambda_u(t) = \begin{cases} \sum_{t_i^u \leq t} g_{a_i^u}\left(t - t_i^u\right), & t \leq t_{l_u} \\ 0, & \text{otherwise} \end{cases}, \tag{5}$$

where $g_{a_i^u}(\cdot)$ is the kernel function which is used to model the effect of channel a_i^u on the conversion of user u, l_u the length of the sequence of touchpoints of user u, and t_i^u the specific time an advertisement was shown to user u (Zhang et al., 2014). Using this hazard function, the contribution of each touchpoint to the likelihood of a conversion can be measured as well as its time-decaying property. Hence, the AH model models the conversions as a user's 'death' after a sequence of advertisements using survival theory with the above specified hazard function. The final model is fitted by iteratively maximizing the log-likelihood function for all users.

After fitting the AH model, the attribution to a specific channel of the different touchpoints is derived from the conversion probability that a user clicks on an advertisement from that specific channel which is defined as

$$P(y = 1|\beta_{a_i^u}, \omega_{a_i^u}, T) = 1 - \exp(-\beta_{a_i^u}(1 - \exp(-\omega_{a_i^u}T))), \tag{6}$$

where T is the predefined observational window over which the hazard function is computed, $\beta_{a_i^u}$ the strength of the impact of the advertisement on the conversion, and $\omega_{a_i^u}$ its time-decaying property. Note that the derivation of Equation (6) is based on the usage of exponential kernels for $g_{a_i^u}(\cdot)$. More mathematical details about the AH model can be found in the paper of Zhang et al. (2014).

Fourth, the AMTA model is considered which is similar to the AH model. Besides survival analysis for conversion modelling, the AMTA model uses the hazard rate of conversion at the specific time of the touchpoint to model the conversion attribution (Ji & Wang, 2017). Therefore, the AMTA model is based on two main assumptions; *i*) the effect of an advertisement is fading with time, and *ii*) the effects of advertisements on the browsing path of a user are additive. The hazard rate for conversion at time t for user u is specified as follows

$$h(t|b^u) = \sum_{t_i^{\leq} t} \alpha_{a_i^u}(x_{e,i}^u) f_{a_i^u}\Big(t - t_i^u, x_{d,i}^u\Big), \tag{7}$$

where b^u is the browsing path of user u, x_i^u a set of features, $x_{e,i}^u$ the measure for the effect of the advertisement exposure on a specific channel at a specific time, $\{a_i^u, t_i^u\}$, $x_{d,i}^u$ the decay speed of the advertisement, and $f_{a_i^u}$ the time-decaying kernel . Given the browsing path b^u and advertisement $\{a_i^u, t_i^u, x_i^u\}$, and using the hazard rate specified above, the contribution of an advertisement to the conversion at timestamp t and the contribution of channel k for the conversion of user u at timestamp t can be computed as

$$att_i^u = \frac{\alpha_{a_i^u}(x_{e,i}^u) f_{a_i^u}(t - t_i^u, x_{d,i}^u)}{h(t \mid b^u)} \tag{8}$$

and

$$att_k^u = \frac{\sum_{t_i^u < t, a_i^u = k} \alpha_{a_i^u}(x_{e,i}^u) f_{a_i^u}(t - t_i^u, x_{d,i}^u)}{h(t \mid b^u)} \tag{9}$$

respectively. As for the AH model, the AMTA model is fitted by maximizing the log-likelihood function for all users. Ji and Wang (2017) have shown that the AMTA model is superior to state-of-the-art techniques in conversion rate prediction. More details and mathematical specifications of the AMTA model can be found in the paper (Ji & Wang, 2017).

4.1.3 ARNN and DARNN

Apart from the four previously mentioned baseline models, Ren et al. (2018) implemented neural networks using an attention mechanism to model the attribution of different advertisements and channels. Instead of optimizing a log-likelihood function or estimating coefficients per channel, neural networks are trained by feeding the system one sequence at the time, referred to as sequential modelling. The advantage of sequential modelling is that the model is able to generate more precise attribution results based on individual user behavior as the models take an entire user sequence as input and learn the attribution for one sequence at the time. However, a downside is that the training process of the neural networks is time consuming and that neural network based models often consist of multiple layers making the models relatively hard to interpret and therefore potentially inaccessible for advertisers to use.

Ren et al. (2018) have implemented a standard RNN with a single attention mechanism to illustrate the advantage of their newly introduced dual-attention mechanism. The RNN with the single attention mechanism which models the impression-level behavior, is the encoder part of the model illustrated in Figure 3. The RNN is fed one user sequence at the time consisting of several touchpoints. As each touchpoint is mostly categorical, an embedding layer is included in the RNN which takes the sparse input feature reflecting a user sequence consisting of multiple touchpoints as input and transforms this into a dense representation vector, which has been widely used in related literature (Qu et al., 2016; X. Wang et al., 2017). Then, the embeddings are fed through the encoder RNN function as

$$h_j = f_e(x_j, h_{j-1}), \tag{10}$$

where h is the hidden vector at each time step j, and f_e a standard long short-term memory (LSTM) model as described in Hochreiter and Schmidhuber (1997). Each hidden state gives a representation of the input data, a user sequence consisting of multiple touchpoints. Using the attention of the impression-level behavior the conversion rate of the sequence is predicted.

In addition to the above encoder modelling the impression-level behavior, the DARNN models the click probability and applies dual-attention to include both the impression-level behavior and click probability in predicting the conversion. Hence, the DARNN can be split into three parts; *i*) the encoder for the impression-level behavior modelling as described above, *ii*) the decoder and sequential prediction for the click probability, *iii*) dual-attention for jointly modelling impression and click behavior in order to predict the conversion rate. Using this sequence-to-sequence construction, the DARNN can alleviate the data sparsity problem of the rare event of a click and an even rarer event of a conversion by using the signal of the click behavior to boost estimation capacity for the sparse conversion behaviors. Ren et al. (2018) have shown that modelling the click-level attribution is statistically more important for attributing credits than impression-level behaviors as the clicking of advertisements generates revenue for the publishers.

The goal of the sequential prediction of the click-level is to model the click action at each time an ad is shown to the user. This is done by defining a probability over the click outcomes, $\mathbf{z}$, as

$$p(\mathbf{z}) = \prod_{j=1}^{m_i} p(z_j = 1 | \{z_1, ..., z_{j-1}\}, \mathbf{x}), \tag{11}$$

where $\mathbf{z} = (z_1, ..., z_{m_i})$ and $\mathbf{x} = (x_1, ..., x_{m_i})$ with m_i the total browsing activity of user i. Then, with the decoder shown on the right hand side of Figure 3 the conditional probability of a click is modeled as

$$\hat{z}_j = p\left(z_j = 1 \mid \{z_1, \ldots, z_{j-1}\}, \mathbf{x}\right) = g\left(z_{j-1}, s_j\right), \tag{12}$$

where g is the output function which is a multi-layer fully connected perceptron of Feed Forward NN with sigmoid activation function which transforms the output into the click probability, $p(z_j = 1)$. Further, s_j is the hidden vector at click-level of the j^{th} touchpoint containing information about the probability of the touchpoint being clicked, which is computed as

$$\boldsymbol{s}_j = f_d\left(\boldsymbol{s}_{j-1}, z_{j-1}, \boldsymbol{h}_{m_i}\right), \tag{13}$$

where f_d has the same structure of the LSTM model as the encoder f_e of Equation (10) and where the last hidden state h_{m_i} from the encoder is used. A more detailed mathematical description of the DARNN can be found in Ren et al. (2018).

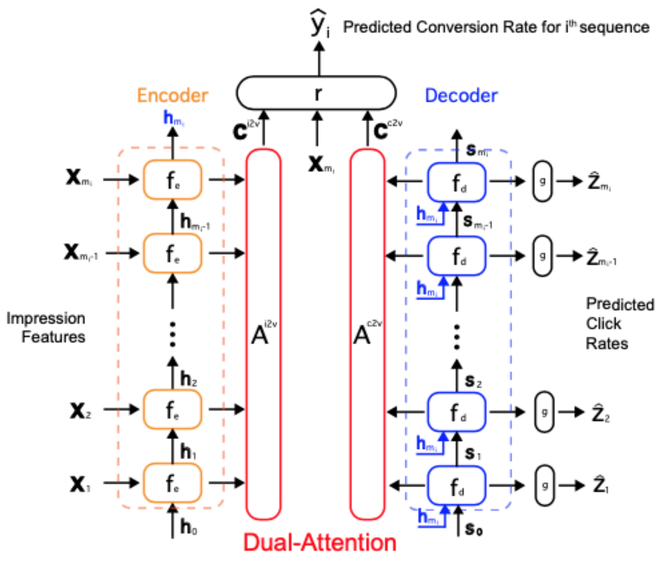

Figure 3: Sequential modelling with dual-attention (DARNN). Source: Ren et al. (2018)

4.2 PNN and Attention PNN

Due to the possible disadvantages of interpretability and long training times of neural networks mentioned earlier, this research proposes a PNN which incorporates the idea of attention to model the attribution of the touchpoints. The PNN was first introduced by Specht

(1990) to resolve the issue of long training times. It uses a Bayesian approach to generate solutions and thereby avoids local minima of heuristic approaches using back propagation, as done in the ARNN and DARNN. PNN's are used in many applications ranging from classification and mapping tasks to directly estimating posteriori probabilities. Especially because of the training simplicity and sound statistical foundation in Bayesian estimation theory, the PNN has widely been used in classification problems (Mao, Tan, & Ser, 2000). The main advantages of the PNN are; *i*) real-time use of the model, *ii*) few to no hyper-parameters which require tuning, and *iii*) relatively easy interpretability of the model. As shown in Figure 4, a standard PNN for a binary classification consists of four layers; *i*) the input layer, *ii*) the pattern layer, *iii*) the summation layer, and *iv*) the decision layer. This research proposes a method to incorporate the idea of an attention mechanism into a PNN for which the structure is shown in Figure 5.

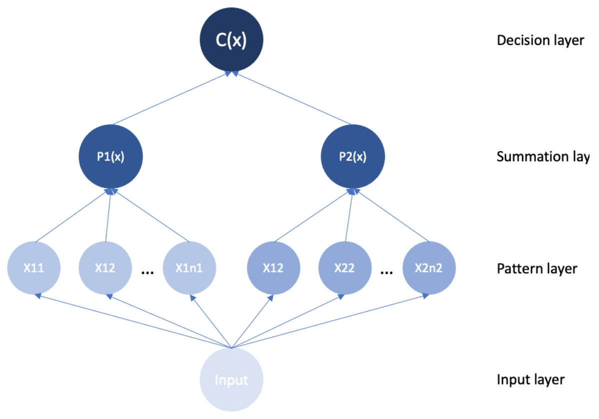

Figure 4: PNN model model structure for a binary classification problem.

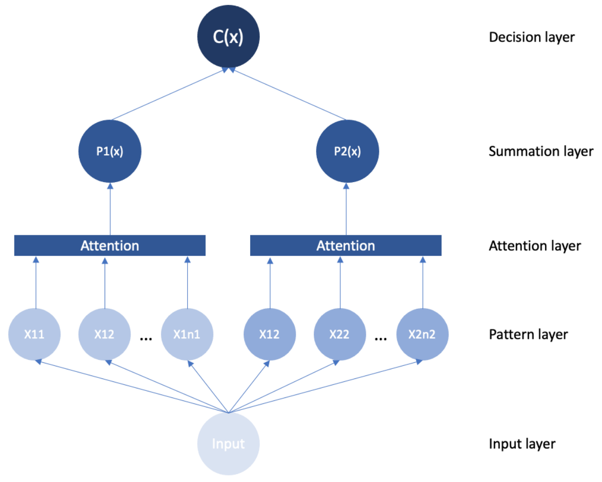

Figure 5: APNN model structure for a binary classification problem.

4.2.1 Classical PNN

The classical and standard PNN without attention layer consist of four layers and its structure for a binary classification problem is shown in Figure 4. First, the input layer distributes the input to the neurons in the pattern layer, where the pattern layer is split for the different classes, which in this research is either conversion or no conversion. Hence, in the pattern layer each neuron represents a case of the training set. At each neuron $\mathbf{x}_{ij}$ of the pattern layer the distance to the input $\mathbf{x}$ is computed as follows

$$\phi_{ij}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\Big[- \frac{(\mathbf{x} - \mathbf{x}_{ij})^T(\mathbf{x} - \mathbf{x}_{ij})}{2\sigma^2} \Big], \tag{14}$$

where d is the dimension of the input vector $\mathbf{x}$, and σ the smoothing parameter which can be tuned (Mao et al., 2000).

Thereafter, the output of the pattern layer is passed on to the summation layer neurons. The summation layer computes the maximum likelihood of pattern $\mathbf{x}$ being classified to class 0 or 1 by summarizing and averaging the output of all neurons belonging to the same class as

$$p_i(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{N_i} \sum_{j=1}^{N_i} \exp\Big[- \frac{(\mathbf{x} - \mathbf{x}_{ij})^T(\mathbf{x} - \mathbf{x}_{ij})}{2\sigma^2} \Big], \tag{15}$$

where N_i is the total number of samples in class i. The output $p_i(\mathbf{x})$ reflects the probability of input $\mathbf{x}$ being classified to class i, where $i \in \{0, 1\}$ in case of a binary classification.

Lastly, the decision layer assigns a class to the input $\mathbf{x}$ based on the output of the summation layer as follows

$$\hat{C}(\mathbf{x}) = \arg\max p_i(\mathbf{x}), \;\; i \in \{0, 1\}, \tag{16}$$

where $\hat{C}(\mathbf{x})$ is the estimated class 0 or 1. The above decision rule corresponds to the Bayes' decision rule if the a priori probabilities for each class are the same, and is based on the assumption that losses for making an incorrect decision are the same for each class (Mao et al., 2000).

The structure of the PNN as just described enables the usage of the model in real-time, which is an advantage of the PNN as mentioned before. This is due to the fact that once one pattern representation for a class has been observed, new patterns can be generated by the network allowing the PNN to operate in parallel without the need for feedback from the neurons (Specht, 1990). Further, by choosing the suitable value of the smoothing parameter σ the shape of the decision surfaces can be made as complex as necessary and can approach the Bayes optimal.

4.2.2 Standard PNN

The standard PNN computes the distance between the neuron and each new input. In the application of advertisement conversion, the input is a single touchpoint from the test set and the classes correspond to touchpoints of converted and non-converted sequences. Hence, in contrast to the benchmark models the input for the standard PNN and different APNNs is a single touchpoint of a sequence instead of the entire sequence.

A touchpoint consists of the click level, campaign IDs and category IDs. As these are all categorical variables, Equation (14) cannot be applied in the pattern layer. To be able to use Equation (14) as in the standard PNN one could learn embeddings per touchpoint. However, doing this the advantage of no required model training is lost as training would have to be done to generate the embeddings for each touchpoint. To avoid training, each touchpoint is transformed into a sparse vector containing zeros and ones, where the first element reflects the click level, the next 675 elements the campaign IDs and the last 59,098 elements the category IDS, resulting in

$$\mathbf{x} = \begin{bmatrix} \underbrace{1}_{\text{click}} & \underbrace{00...00}_{\text{campaign IDs}} & \underbrace{01...10}_{\text{category IDs}} \end{bmatrix}, \tag{17}$$

where the presence of a one at the first element in the matrix indicates whether the advertisement is clicked, a one in the next 675 elements the campaign of the advertisement, and ones in the last 59,098 elements the categories corresponding to the advertisement. Hence, $\mathbf{x}$ is the sparse vector representation of each touchpoint.

After the transformation of the touchpoints into the sparse vectors, the 'distance' between the touchpoints can be computed using the cosine similarity. In the pattern layer the cosine similarity of a new touchpoint and each of the training points is computed at each of the neurons of the two classes. Given the sparse input vector $\mathbf{x}$ and the neuron $\mathbf{x}_{ij}$ the cosine similarity is computed as

$$\psi_{ij} := \frac{\mathbf{x} \cdot \mathbf{x}_{ij}}{\|\mathbf{x}\| \|\mathbf{x}_{ij}\|}. \tag{18}$$

As our data merely consists of display advertisements, all touchpoints can easily be compared with each other by computing the cosine similarity. Further, Equation (18) does not contain a parameter which has to be tuned as the smoothing parameter σ in Equation (14), which benefits the real-time use of the model.

After the pattern layer, the summation layer computes the average cosine similarity for the converted and non-converted class as follows

$$\Psi_i := \frac{1}{N_i} \sum_{j=1}^{N_i} \psi_{ij}, \tag{19}$$

where ψ_{ij} is the cosine similarity between $\mathbf{x}$ and neuron $\mathbf{x}_{ij}$ and N_i the number of touchpoints in class i. This is similar to Equation (15) of the standard PNN and assumes equal contribution of each touchpoint in predicting the conversion. After the summation layer of both classes, the touchpoint is assigned to the class with the highest average cosine similarity as follows

$$\hat{C}(\mathbf{x}) = \arg\max \Psi_i, i \in \{0, 1\}. \tag{20}$$

Hence, the standard PNN as just described takes each touchpoint into consideration independently of its sequence and focuses on the prediction of the class of individual touchpoints. Thus, as Zhang et al. (2014) and Shao and Li (2011), the standard PNN assumes that conversion of a user sequence can be predicted based on an individual advertising touchpoint and examines whether similar touchpoints have led to a conversion in the past. As there are so many different individuals leading to many different shopping behaviors, it is very time consuming and computationally intensive to model each of these behaviors individually. Further, given that non-converted sequences contain relatively many touchpoints compared to converted sequences, one can assume that there are many advertisements which are not triggering and persuasive in generating conversions. Hence, it seems plausible to assume that if a new advertisement is developed which is similar to many of the advertisements included in sequences which did not lead to an actual conversion, that this advertisement is not likely to be successful in generating conversions. The same holds the other way around, if a new advertisement is developed which is similar to advertisements often contained in converting sequences, this advertisement is likely to be successful in generating conversions. Hence, the standard PNN examines the relation between touchpoints rather than modelling sequential behavior and uses these relations to predict conversion. The relation used in this research is the cosine similarity in which the click level is included which is a good indicator for a triggering advertisement and which is important for attribution modelling. In the standard PNN all touchpoints are assumed to be equally important in the prediction of a conversion. Hence, the standard PNN does not include attribution modelling.

4.2.3 Standard APNN

To include attribution modelling in the model and to be able to give more credits to touchpoints/advertisements which contributed more to the likelihood of a conversion, the standard APNN is introduced. The standard APNN contains one additional layer, the attention layer, after the pattern layer as shown in Figure 5. The attention mechanism of the standard APNN is based on the similarities of touchpoints and assumes that sequences containing touchpoints which are very similar will lead to the same outcome. Thus, these touchpoints should be given more weight. In the attention layer the weights of the converted and non-converted touchpoints/neurons are computed by normalizing the output, the computed cosine similarities, of the pattern layer, which is similar to the Content-Base attention mechanism introduced by Graves et al. (2014). Then, the summation layer computes the weighted average cosine similarity of the training samples per class as

$$\Psi_i = \frac{1}{N_i} \sum_{j=1}^{N_i} w_{ij} \psi_{ij}, \tag{21}$$

where w_{ij} is the weight for each neuron $\mathbf{x}_{ij}$, ψ_{ij} the similarity between $\mathbf{x}$ and neuron $\mathbf{x}_{ij}$ and N_i the number of touchpoints in class i, with $i \in \{0, 1\}$. As the weights are determined based on the similarities of both converted and non-converted touchpoints, one expects for example that the weights for the similarities of a converted touchpoint are higher for the neurons in the converted class and vice versa. After the summation layer, the output is passed on to the decision layer, which is the same as for the standard PNN defined in Equation (20). As the standard APNN includes an attention mechanism and aims to give attention to more relevant touchpoints, one expects that this model is able to predict conversions and non-conversions more accurately than the standard PNN.

4.2.4 Similar Point APNN

The standard APNN described above takes into account all touchpoints through the weighted average cosine similarity. However, although touchpoints within a class have led to the same outcome and are therefore similar to some extend, touchpoints within a class can still differ resulting in a lower cosine similarity between these touchpoints. As these touchpoints are more difficult to compare, they are not as relevant in predicting the conversion outcome of the input as the touchpoints with the highest cosine similarity. In order to minimize the use of relatively uninformative information, the similar point APNN is introduced for which the summation and decision layer are based on the idea of k-nearest neighbors. Instead of computing the weighted average cosine similarity over all points within a class in the summation layer, the similar point APNN examines the conversion outcome of the k most similar touchpoints. Then, in the decision layer the touchpoint is assigned to the most common outcome amongst the k most similar touchpoints. Hence, the prediction is only based on the k most similar/important neurons instead of all training points. Thus, the similar point APNN focus on improving the accuracy of conversion prediction, but thereby loses the ability of attribution modelling and is not able to provide the uncertainty of a prediction.

The similar point APNN minimizes the use of uninformative information even more than the standard APNN. Hence, it is expected that the similar point APNN will predict the conversion outcome more accurately than both the standard PNN and standard APNN. The hyperparameter of this model is the number of similar points to include, k. In order to gain insights in the number of 'neighbors' or similar points to include, the model will be evaluated for $k \in \{1, 3, 5, 9, 15\}$.

4.2.5 Attribution Modelling with Standard APNN

In comparison with the benchmark models, the standard APNN does not employ sequential modelling or optimizes the log-likelihood for all users. The standard APNN described in Section 4.2.3 includes attribution modelling from a different angle as it bases the attribution on the relation (cosine similarity) between individual touchpoints. Hence, the standard APNN assumes that the similarity is a reliable indicator for attributing credits to the different advertisements. This assumption is plausible as for example advertisements which contributed relatively much to the likelihood of a conversion are probably clicked often, leading to a higher cosine similarity. Using the computed cosine similarities for the converted touchpoints, a researcher can attribute credits to the different touchpoints by normalizing the cosine similarities of the touchpoints. Thus, this approach assumes that advertisements which have much in common with many of the converted touchpoints/advertisements are more likely to contribute to the conversion and will therefore achieve a higher cosine similarity and receive more credits. However, this approach is highly dependent on the input variable used to compute the cosine similarities and therefore the resulting similarity weights indicating the attribution are not stable and reproducible.

To improve stability for attribution modelling, the bagging idea of the bagged LR baseline model can be used. For all touchpoints in the test set, the cosine similarities with the converted touchpoints of the training set are computed as credits are merely assigned in case of a conversion. To stabilize the attribution estimates, all computed cosine similarities for the converted touchpoints with each point of the test set are saved and averaged after wards. The computed average cosine similarity also contains the similarity of the converted training touchpoints with touchpoints of the test set which eventually did not lead to a conversion. However, as it is expected that the cosine similarity between these non-converted test points and the converted training points will be relatively low for all converted touchpoints, this is not likely to influence the attribution results significantly. Finally, the contribution of each touchpoint to the likelihood of a conversion of a user sequence is computed by normalizing the average cosine similarities of the converted class. Hence, by averaging and normalizing the cosine similarities, an indication of the attribution credits per advertisement in case of a conversion is given.

The dataset of Criteo used in this research merely consists of display advertisements. However, in case a dataset consists of advertisements of different channels, display, video, music, etc., the weights of the touchpoints per channel can be aggregated and used to gain insights in the effectiveness of the different channels and attribute credits to the different channels accordingly. The higher the attribution weight, the bigger the contribution of the channel to the likelihood of a conversion and the higher the rewarded credits to the channel should be.

4.3 Performance Evaluation

To evaluate the performance of the benchmark models, Ren et al. (2018) used the AUC and the log-loss/Cross-Entropy evaluation metrics. As the standard PNN and APNNs do not predict a conversion probability like the benchmark models, the performance of the standard PNN and APNNs are mainly evaluated using accuracy in combination with the F-score. However, in order to compare the standard PNN and APNNs with the benchmark models, an approximation of the conversion probability based on the (weighted) average similarity is used to compute the log-loss and AUC for the standard PNN and APNN.

First, the AUC is computed as the area under the ROC curve which plots the True Positive Rate (TPR), also known as recall, against the False Positive Rate (FPR) at different classification thresholds. The classification threshold indicates the number of positive items and by lowering the threshold both the False Positives and True Positives increase. Therefore, the AUC has no relationship to the direct output value of the model, but considers the pairwise ranking performance. Hence, the similarities of the PNN and APNN can be used as an approximation to compute the conversion probability on which the different thresholds of the AUC will be based. To approximate the conversion probability the softmax activation function is applied to convert the (weighted) average cosine similarities into a probability,

$$p(y=1) = \frac{e^{\psi_{y=1}}}{e^{\psi_{y=1}} + e^{\psi_{y=0}}}, \tag{22}$$

where $\psi_{y=1}$ and $\psi_{y=0}$ are both between zero and one and reflect the (weighted) average similarity of the input with converted touchpoints and non-converted touchpoints respectively. Hence, if $p(y=1)$ is larger than 0.5 the (weighted) average similarity of the converted touchpoints is higher than the (weighted) average similarity of the non-converted touchpoints.

Thus, the input touchpoint is more likely to lead to a conversion.

Second, Ren et al. (2018) use the log-loss as an evaluation metric which is computed as

$$\text{log-loss} = -\frac{1}{N}\Big[\sum_{i=1}^{N}(y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i))\Big], \tag{23}$$

where N is the number of data points, y_i the true value taking a 0 or 1 for data point j and $\hat{p}_i$ is the probability corresponding to classifying data point i as 1. Using the conversion probability approximation described above, the log-loss is computed for the standard PNN and APNN. However, as the prediction of the similar point APNN is based on the mode of the corresponding classes of the k number of similar points, the approximation process of the conversion probability cannot be applied. Hence, for the similar point APNN the AUC and log-loss are not computed.

Third, the accuracy is straightforward and computed as the number of correctly classified touchpoints divided by the total number of touchpoints which had to be classified. Lastly, the F-measure is computed as

$$F = \frac{(1 + \beta^2) \cdot recall \cdot precision}{(\beta^2 \cdot precision) + recall}, \tag{24}$$

where β determines if either precision or recall is weighted more heavily. In this research $\beta = 1$ for which precision and recall are given equal weight in which case F reflects the harmonic mean of the two. The higher the F-score, the better. In general values for the AUC scores are higher and differ less than the values of the F-scores as the AUC has no direct relationship to the output and the threshold used for computing the F-scores is potentially far from optimal. However, optimizing the threshold used to compute the F-score is beyond the scope of this research.

Although there is no one-to-one correlation between the AUC, log-loss, F-score and accuracy, all measures tell something about the information level of the model. Hence, it is plausible to assume that models obtaining higher accuracy rates and F-scores have a better model performance and thus also obtain higher AUC scores and lower values for the log-loss.

Using this, an indication of the comparative performance of the benchmark models and the standard PNN and APNNs can be given based on the AUC and log-loss reported by Ren et al. (2018) for the benchmark models and the computed AUC and log-loss for the standard PNN and APNNs evaluated in this research.

5 Results

The results can be split into four parts. First, Section 5.1 discusses the results for the benchmark models generated by Ren et al. (2018). Second, Section 5.2 evaluates the performance of the standarad PNN and APNNs. Third, Section 5.3 compares the benchmark models with the standard PNN and APNNs. Lastly, Section 5.4 elaborates upon the attribution results generated based on the similarity between the converted touchpoints.

5.1 Results Benchmark Models

The results for the six benchmark models are shown in Table 3. These results show that the attention based models, ARNN and DARNN, result in much better performance for the conversion prediction than the other models based on AUC and log-loss. This is due to the pattern recognition capability of the deep neural network applied when using sequential modelling (Ren et al., 2018). As the AH and AMTA do not recognize much of the sequential pattern behavior and model the additive hazard ratio of conversion for each touchpoint, these models perform worse with respect to the deep neural network models based on both AUC and log-loss. However, as the AMTA model includes a hazard rate for the time of the conversion its model performance improves significantly compared to the AH model. Further, the bagged LR model is a solid, flexible baseline model resulting in a relatively good model performance, whereas the SP model performs worse due to a more inflexible structure which allows for less covariates. However, the bagged LR model is slightly more difficult to interpret compared to the intuitive SP model as described in Section 4.1.1 (Shao & Li, 2011). For a more detailed description of the performance of these benchmark models, the reader is referred to the paper from Ren et al. (2018).

Table 3 *Benchmark model evaluation. Results generated by Ren et al. (2018); AUC: the higher, the better; log-loss: the lower, the better.*

Model	AUC	Log-Loss
Bagged Logistic Regressions (LR)	0.9286	0.3981
Simple Probabilistic (SP)	0.6718	0.5535
Additive Hazard (AH)	0.6791	0.5067
Adiitional Multi-Touch Attribution (AMTA)	0.8465	0.3897
Attention Recurrent Neural Network (ARNN)	0.9793	0.1850
Dual-Attention Recurrent Neural Network (DARNN)	0.9799	0.1591

5.2 Results PNN and APNNs

Table 5.2 shows the accuracy and F-scores of the standard PNN and different APNNs. As expected, the standard APNN outperforms the standard PNN based on accuracy and log-loss and achieves similar scores for the AUC and F-score. As mentioned in Section 4.3, the values for the AUC are higher than the F-scores indicating that the threshold used in this research to compute the F-scores is probably not optimal. However, as mentioned optimizing this threshold is beyond the scope of this research. The improved model performance of the standard APNN is explained by the usage of weights in computing the average cosine similarity allowing to put more emphasis on the outcome of similar points in the prediction of the conversion.

Further, all similar point APNNs outperform the standard PNN and APNN which is in line with the expectation as the similar point APNN minimizes the use of uninformative information. Besides, the similar point APNNs outperform the case if only the majority class (no conversion) would have been predicted, which would have resulted in an accuracy of ca. 96% as described in Section 3. The standard PNN and APNN and similar point APNN with $k = 15$ achieve the lowest F-score of the evaluated models, indicating that these models perform worst in accurately classifying a conversion, the minority class (Bekkar et al., 2013). Hence, this indicates that including more than ca. ten touchpoints for predicting a conversion reduces the ability of the model to correctly classify the minority class. As mentioned earlier, the AUC and log-loss cannot be computed for the similar point APNNs as these models do not predict a conversion probability, but predict the class based on the most common outcome of the k-nearest neighbors, in this application the k most similar touchpoints. Hence, this is a disadvantage of the similar point APNNs as it is not possible to evaluate the uncertainty of the predictions. On the other hand, the increase in accuracy and F-score of the similar point APNNs is due to a more efficient use of information by merely looking at the outcome of the most similar points and ignoring irrelevant touchpoints for predicting a conversion. Based on the accuracy and F-score the optimal value of k is $k = 1$ and including more points does not significantly increase model performance. Although the accuracy is slightly higher for the similar point APNN with $k = 3, 5, 9$ and 15, the F-score for the similar point APNN with $k = 1$ achieves by far the highest value of the evaluated models. Hence, the similar point APNN with $k = 1$ is the best PNN based model to accurately predict the minority class and therefore the similar point APNN is preferred (Bekkar et al., 2013).

Table 4 *Performance evaluation standard PNN and APNNs. Results generated for ca. 2 million touchpoints and rounded for last printed digit; Accuracy, F-measure, AUC: the higher, the better; log-loss: the lower, the better. AUC and log-loss cannot be computed for similar point APNNs as prediction is the actual class which cannot be transformed into a probability.*

Model	Accuracy (%)	F-measure	AUC	Log-loss
PNN	70.6	0.272	0.908	0.679
APNN	71.6	0.272	0.903	0.693
Similar Point APNN - $k = 1$	98.3	0.540	*na*	*na*
Similar Point APNN - $k = 3$	98.4	0.505	*na*	*na*
Similar Point APNN - $k = 5$	98.5	0.497	*na*	*na*
Similar Point APNN - $k = 9$	98.5	0.422	*na*	*na*
Similar Point APNN - $k = 15$	98.4	0.273	*na*	*na*

5.3 Comparison Benchmark and (A)PNNs

As advertisers and ad exchanges have different marketing strategies and objectives, this section gives an overview of the different models and compares their performance, strengths and weaknesses (Shao & Li, 2011). However, it is difficult to compare the results given above for the different models for three reasons; *i*) this research and Ren et al. (2018) have not used the exact same data to evaluate the models as this research used part of the Criteo dataset and Ren et al. (2018) used undersampling, *ii*) Ren et al. (2018) do not provide the accuracy and F-score of their models as done in this research, and *iii*) the similar point APNNs predict the class instead of the conversion probability which disables the computation of the AUC and log-loss. Nevertheless, as described, the accuracy, log-loss, F-score and AUC say something about the performance and information level of a model. Hence, it is expected that *if* the AUC and log-loss could have been computed for the similar point APNNs, the AUCs and log-losses would increase and decrease respectively compared to the AUC and log-loss of the standard PNN and APNN as the accuracy and F-scores increase. Hence, the AUC and log-loss for the similar point APNN with $k = 1$ are expected to approach the AUC and log-loss achieved by the ARNN and DARNN making the performance of the similar point APNN with $k = 1$ comparable with the performance of the ARNN and DARNN.

The main difference between the nature of the standard PNN and different APNNs and the nature of the ARNN and DARNN is the usage of individual touchpoints and sequential modelling respectively for predicting a conversion and modelling attribution. Further, the benchmark models, the bagged LR, SP, AH, and AMTA model, are not only able to predict *if* a sequence is going to lead to a conversion, but also *when* this conversion will take place (Ji & Wang, 2017). The standard PNN and different APNNs merely predict *if* a sequence containing a certain touchpoint is going to lead to a conversion. Hence, this is a limitation for advertisers using the standard PNN or an APNN as they lack this type of information when optimizing their advertising strategies. On the other hand, advantages of the standard PNN and different APNNs are that training of the models is not required, which benefits quick generation of results, and that no sampling technique has to be used allowing all data to be used. Also, no likelihood function has to be optimized as for the bagged LR and SP model, which could be computationally intensive. Given all results above, Table 5 provides an overview of the capabilities of the different models.

Based on Table 5 one can see that an advertiser or ad exchange has to make a trade-off between on the one hand real-time usage and interpretability of the model and on the other hand long training times and the ability and approach of attribution modelling. Although simple implementation and interpretation of a model are always stimulating for advertisers or ad exchanges to enhance a model, this trade-off has to be made based on the objective of the advertiser or ad exchange. For example, if an ad exchange has to set a new pricing scheme for advertisements which has to be based on sequential user behavior, the DARNN is most suited in case long training time and relatively hard interpretation of a model are no issue. On the other hand, if an advertiser wants to gain insights in the performance of newly developed advertisements and avoid long training times of a model, the similar point APNN with $k = 1$ is most suited as it is simple to implement.

Table 5 *Model comparison. A model is able (disable) to foresee the feature in the column if a tick (blank) is shown. The ringlet indicates that the model is able to foresee the feature, but not as precise or accurate as models with a tick.*

Model	Real-time usage	Interpretability	Precise attribution	Accurate conversion prediction	Prediction of time of conversion
Bagged LR	∼	✓		✓	✓
SP	∼	✓			✓
AH					✓
AMTA				∼	✓
ARNN			✓	✓	
DARNN			✓	✓	
PNN	✓	✓			
APNN	✓	✓	∼		
Similar point APNN	✓	✓		✓	

5.4 Attribution Results of Standard APNN

Compared to the bagged LR, SP, AH, AMTA, ARNN and DARNN model, the standard APNN does not learn the attribution taking into account the outcome of all the sequences, as it merely examines the advertisements included in sequences which actually converted. However, as mentioned in the literature review, the rule based heuristics used for attribution modelling also do not consider the outcome which is a disadvantage as the system can then easily be fooled. In contrast to rule based heuristic mechanisms, the attributions computed by the standard APNN are based on the cosine similarity which is based on the input of the data of the converted touchpoints. Therefore, the standard APNN does not entirely ignore the final outcome of the user sequences and is thus not easily fooled by advertisers. Hence, the standard APNN is able to attribute credits to different advertisements and channels.

Although direct comparison of the models is not possible as described above, it does seem that the model performance of the standard APNN is relatively bad compared to the ARNN and DARNN based on the AUC and log-loss when it comes to conversion prediction. Also, the standard APNN models attribution from a different angle than the ARNN and DARNN as it bases the attribution on the relation of individual touchpoints rather than using the entire user sequence. Further, the standard APNN does not require the long training times of the model which could be the case for the ARNN and DARNN for large amounts of data. Hence, based on the objective, an advertiser or ad exchange should take the above aspects into consideration when choosing a model for attribution modelling. Lastly, it should be noted that the similar point APNN with $k = 1$ is expected to have comparable model performance with the ARNN and DARNN, but merely examines the relation with the most similar touchpoint and is therefore not able to model the attribution per touchpoint.

Table 6 gives the attribution results generated by the standard APNN. It shows a list of the 20 most popular features of the touchpoints with the highest (top $x\%$) rewarded attribution. The nine features listed in bold in every column are always amongst the top 20 most popular features regardless of the top $x\%$. For example, one can see that touchpoints with the highest attribution results are always clicked. This is in line with the expectation as it is expected that advertisements which contributed most to the conversion are triggering and therefore trigger the Internet user to click the advertisement. The other nineteen features are the campaign or category IDs linked to the touchpoints with the highest rewarded attribution. Unfortunately, Criteo has not disclosed the contextual information linked to these category IDs as described in Section 3. Hence, this research is not able to provide further insights in the success of these categories.

Table 6 *Attribution results of standard APNN. List of 20 most popular features of touchpoints which are rewarded the highest (top x%) attribution. The nine features in bold are always in the top 20 most popular features regardless of the top x%.*

Top 10%	Top 20%	Top 30%	Top 40%	Top 50%
Clicked	**Clicked**	**Clicked**	**Clicked**	**Clicked**
16184514	16184514	16184514	**1619851**	27385784
1619851	25259856	**1619851**	1692980	**1619851**
1696469	**1619851**	1692980	1696469	1692980
2038799	1692980	1696469	**2038799**	1696469
2864985	**2038799**	**2038799**	2139670	1722636
5620638	2864985	2864078	2864078	**2038799**
6190107	8141218	2864985	2864985	2139670
14366054	**14366172**	**14366172**	**14366172**	2864078
14366172	**16312763**	**16312763**	**16312763**	14363299
16312763	21681023	21681023	21681023	**14366172**
21681023	21752252	21752252	21752252	**16312763**
22839829	**22839829**	**22839829**	**22839829**	21752252
22920417	**22920417**	**22920417**	**22920417**	**22839829**
24298181	23007218	23007218	23007218	**22920417**
27040390	23086591	23086591	23086591	23007218
28650087	**24298181**	**24298181**	**24298181**	23086591
29640100	29640100	29640100	29640100	**24298181**
29889521	**29889521**	**29889521**	29642271	**29889521**
30104181	29889556	29889556	**29889521**	29889556

6 Conclusion and Discussion

This research contributes to the current literature of attribution modelling by introducing new models based on the PNN which benefit both advertisers and ad exchanges. In attribution modelling it is essential to seek for a model that optimizes the fairness of credit attribution based on the contribution of the advertisement to the likelihood of a conversion, while preserving a data-driven nature of the model and not losing the ease of implementation and interpretation of the model. This research proposes the standard APNN, a PNN with an attention layer, which allows advertisers and ad exchanges to generate data-driven insights and to attribute credits to advertisements based on the relation (cosine similarity) between individual touchpoints. Further, the similar point APNN was introduced which implements the idea of k-nearest neighbors into the model and aims to use information of relevant touchpoints efficiently in order to make accurate predictions.

To evaluate the performance of the standard PNN and different APNNs, a dataset from the online advertising company Criteo and six benchmark models evaluated by Ren et al.

(2018) are used, the bagged LR, SP, AH, AMTA, ARNN and DARNN. This research concludes that from the standard PNN and different APNNs, the similar point APNN with $k = 1$ results in the best model performance for conversion prediction based on the combination of model accuracy and the corresponding F-score. Hence, using the idea of k-nearest neighbors in combination with the APNN improves the accuracy performance of the model significantly. Although direct comparison with the results of the benchmark models is difficult, the similar point APNN with $k = 1$ achieves comparable model performance with the best performing benchmark models, the ARNN and DARNN. The main advantages of the PNN and APNNs over the benchmark models are the avoidance of long training times and the relatively easy interpretation of the model making the models accessible for advertisers and ad exchanges. However, some of the PNN based models lose precision compared to the ARNN and DARNN which have the great pattern recognition capability used with sequential modelling. Further, the similar point APNNs are not able to take into account the uncertainty of the prediction as they predict a class rather than the probability of a conversion. Hence, the similar point APNN with $k = 1$ is mostly suited for advertisers and ad exchanges who want to avoid long training times and value simplicity of the model and for advertisers who quickly want to gain insights in the performance of their newly developed advertisements.

For attribution modelling advertisers and ad exchanges should use the standard APNN as the similar point APNN is not capable of modelling attribution as it focuses on accurate conversion prediction. The standard APNN does not use sequence-to-sequence modelling as done in the ARNN and DARNN to model the attribution of the touchpoints, but approaches attribution modelling from a different angle by computing the attribution based on the relation, cosine similarity, with the converted touchpoints. Thus, the standard APNN assumes that advertisements which have much in common with many of the converted advertisements are more likely to contribute to the conversion and should therefore receive higher credits. The advantage of this approach is that the standard APNN does not require long training times to generate attribution results. However, a disadvantage is that the standard APNN is less accurate in predicting a conversion than most of the benchmark models and especially compared to the ARNN and DARNN.

The results generated in this research provide interesting opportunities for future research. First, one of the opportunities for future research is to evaluate the standard PNN and different APNNs for a dataset containing advertisements from different channels as the dataset used in this research merely contains different types of display advertisements. Given the approach in the methodology it would be interesting to examine the performance of the APNN for a dataset containing these different types of advertisements and to compare the attribution results to the attribution results of the benchmark models. Second, it would be interesting to examine if the standard PNN and different APNNs could include the position of the touchpoints in the sequence and to see if this improves model performance. One way this could be done is, by using a binary indicator for the position in addition to the sparse input vector currently used for computing cosine similarity. Third, costs of advertisements are an important aspect of the advertising strategy of advertisers. Therefore, it would be of added value to include the cost factor of advertisements into the model such that advertisers can use this when optimizing their advertising accuracy.

References

Anderl, E., Becker, I., Von Wangenheim, F., & Schumann, J. H. (2016). Mapping the customer journey: Lessons learned from graph-based online attribution modeling. *International Journal of Research in Marketing*, *33*(3), 457–474.

Arava, S. K., Dong, C., Yan, Z., Pani, A., et al. (2018). Deep neural net with attention for multi-channel multi-touch attribution. *arXiv preprint arXiv:1809.02230*.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, *6*(1), 20–29.

Bekkar, M., Djemaa, H. K., & Alitouche, T. A. (2013). Evaluation measures for models assessment over imbalanced data sets. *J Inf Eng Appl*, *3*(10).

Dalessandro, B., Perlich, C., Stitelman, O., & Provost, F. (2012). Causally motivated attribution for online advertising. In *Proceedings of the sixth international workshop on data mining for online advertising and internet economy* (pp. 1–9).

Delacour, H., François, N., Servonnet, A., Gentile, A., & Roche, B. (2009). Les rapports de vraisemblance: un outil de choix pour l'interprétation des tests biologiques. *Immuno-analyse & Biologie Spécialisée*, *24*(2), 92–99.

Diemert, E., Meynet, J., Galland, P., & Lefortier, D. (2017). Attribution modeling increases efficiency of bidding in display advertising. In *Proceedings of the adkdd'17* (pp. 1–6).

Di Martino, M., Decia, F., Molinelli, J., & Fernández, A. (2012). Improving electric fraud detection using class imbalance strategies. In *Icpram (2)* (pp. 135–141).

Drummond, C., & Holte, R. C. (2000). Explicitly representing expected cost: An alternative to roc representation. In *Proceedings of the sixth acm sigkdd international conference on knowledge discovery and data mining* (pp. 198–207).

Du, R., Zhong, Y., Nair, H., Cui, B., & Shou, R. (2019). Causally driven incremental multi touch attribution using a recurrent neural network. *arXiv preprint arXiv:1902.00215*.

Geyik, S. C., Saxena, A., & Dasdan, A. (2014). Multi-touch attribution based budget allocation in online advertising. In *Proceedings of the eighth international workshop on data mining for online advertising* (pp. 1–9).

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Ji, W., & Wang, X. (2017). Additional multi-touch attribution for online advertising. In *Thirty-first aaai conference on artificial intelligence.*

Ji, W., Wang, X., & Zhang, D. (2016). A probabilistic multi-touch attribution model for online advertising. In *Proceedings of the 25th acm international on conference on information and knowledge management* (pp. 1373–1382).

Kannan, P., Reinartz, W., & Verhoef, P. C. (2016). *The path to purchase and attribution modeling: Introduction to special section.* Elsevier.

Kee, B. (2012). *Attribution playbook–google analytics.*

Lee, K.-c., Orten, B., Dasdan, A., & Li, W. (2012). Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th acm sigkdd international conference on knowledge discovery and data mining* (pp. 768–776).

Li, H., & Kannan, P. (2014). Attributing conversions in a multichannel online marketing environment: An empirical model and a field experiment. *Journal of Marketing Research*, *51*(1), 40–56.

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mao, K. Z., Tan, K.-C., & Ser, W. (2000). Probabilistic neural-network structure determination for pattern classification. *IEEE Transactions on neural networks*, *11*(4), 1009–1016.

Qu, Y., Cai, H., Ren, K., Zhang, W., Yu, Y., Wen, Y., & Wang, J. (2016). Product-based neural networks for user response prediction. In *2016 ieee 16th international conference on data mining (icdm)* (pp. 1149–1154).

Ren, K., Fang, Y., Zhang, W., Liu, S., Li, J., Zhang, Y., . . . Wang, J. (2018). Learning multi-touch conversion attribution with dual-attention mechanisms for online advertising. In

Proceedings of the 27th acm international conference on information and knowledge management (pp. 1433–1442).

Shao, X., & Li, L. (2011). Data-driven multi-touch attribution models. In *Proceedings of the 17th acm sigkdd international conference on knowledge discovery and data mining* (pp. 258–264).

Shapley, L. S. (1953). A value for n-person games. *Contributions to the Theory of Games*, *2*(28), 307–317.

Specht, D. F. (1990). Probabilistic neural networks. *Neural networks*, *3*(1), 109–118.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, J., Zhang, W., & Yuan, S. (2016). Display advertising with real-time bidding (rtb) and behavioural targeting. *arXiv preprint arXiv:1610.03013*.

Wang, X., Yu, L., Ren, K., Tao, G., Zhang, W., Yu, Y., & Wang, J. (2017). Dynamic attention deep model for article recommendation by learning human editors' demonstration. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining* (pp. 2051–2059).

Xu, J., Shao, X., Ma, J., Lee, K.-c., Qi, H., & Lu, Q. (2016). Lift-based bidding in ad selection. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 30).

Zhang, Y., Wei, Y., & Ren, J. (2014). Multi-touch attribution in online advertising with survival theory. In *2014 ieee international conference on data mining* (pp. 687–696).

Zhao, K., Mahboobi, S. H., & Bagheri, S. R. (2018). Shapley value methods for attribution modeling in online advertising. *arXiv preprint arXiv:1804.05327*.

A Code Data Preprocess

```
# ----- DATA ------
# Set paths filezilla to retrieve original data and save processed data
path_unprocessed_data = r'/home/kdkruiff/Thesis/Data/criteo_attribution_dataset.tsv'
path_processed_data = r'/home/kdkruiff/Thesis/Data/Part1/criteo_dataset_cleaned.tsv'
path_processed_data_duplicates =
    r'/home/kdkruiff/Thesis/Data/Part1/criteo_dataset_cleaned_duplicates.tsv'

def DataPreprocessing(path_unprocessed_data,path_processed_data):

    pd.set_option('display.max_columns', None)
    data = pd.read_csv(path_unprocessed_data, sep='\t').iloc[0:8250000,:]
    data_cleaned = data.drop(['attribution', 'click_nb', 'click_pos', 'cost',
        'cpo', 'time_since_last_click'], axis=1)
    data_cleaned['set_cat'] = data_cleaned.apply(lambda row: set(
        [row['cat1'], row['cat2'], row['cat3'], row['cat4'], row['cat5'],
        row['cat6'], row['cat7'], row['cat8'], row['cat9']]), axis=1)
    data_cleaned['set_cat'] = data_cleaned['set_cat'].apply(tuple)
    data_cleaned = data_cleaned.drop(['cat1', 'cat2', 'cat3', 'cat4', 'cat5',
        'cat6', 'cat7', 'cat8', 'cat9'],axis=1)

    # CLEAR WORKSPACE
    del data

    data_sorted_partial = data_cleaned.sort_values(by='uid')
    col_index_sequence_id = len(data_cleaned.columns.values)
    col_index_position_id = col_index_sequence_id + 1
    data_sorted_partial.insert(col_index_sequence_id, 'sequence_id', np.nan)
    data_sorted_partial.insert(col_index_position_id, 'position_id', np.nan)

    sequence_id = 0
    position_id = 1
    data_sorted_partial.iloc[0, col_index_sequence_id] = sequence_id
    data_sorted_partial.iloc[0, col_index_position_id] = position_id
    for i in range(len(data_sorted_partial)-1):
        if data_sorted_partial.iloc[i, 1] != data_sorted_partial.iloc[i + 1, 1]:
            sequence_id += 1
            position_id = 1
        elif (data_sorted_partial.iloc[i, 1] == data_sorted_partial.iloc[i + 1, 1]
            and data_sorted_partial.iloc[i, 5] != data_sorted_partial.iloc[i + 1, 5]):
            sequence_id += 1
            position_id = 1
        data_sorted_partial.iloc[i + 1, col_index_sequence_id] = sequence_id
        data_sorted_partial.iloc[i + 1, col_index_position_id] = position_id
        position_id += 1

    # Remove all sequences with one touchpoint
    data_cleaned_new = data_sorted_partial.drop(['uid','conversion_timestamp',
        'conversion_id'], axis=1)
    data_duplicates = data_sorted_partial[['uid', 'conversion',
        'conversion_timestamp', 'conversion_id']]
    data_singles = data_duplicates[~data_duplicates.duplicated(keep=False)]
    data_duplicates_new = data_duplicates[data_duplicates.duplicated(keep=False)]
    indices_duplicates = data_duplicates_new.groupby(list(data_duplicates_new))
        .apply(lambda x: x.index).values.tolist()
    indices_singles = data_singles.groupby(list(data_singles))
        .apply(lambda x: x.index).values.tolist()

    # Add column with sequence encoding
    col_index_encoding = len(data_cleaned_new.columns.values)
    data_cleaned_new.insert(col_index_encoding, 'encoding', np.nan)
    encoding = 1
    encoding_singles = 1
    for i in range(len(indices_singles)):
        data_cleaned_new.loc[indices_singles[i], col_index_encoding] = encoding
        encoding += 1
    for i in range(len(indices_duplicates)):
        duplicates = indices_duplicates[i]
        for j in range(0, len(indices_duplicates[i])):
            data_cleaned_new.loc[duplicates[j], col_index_encoding] = encoding
        encoding += 1
    print("Finished␣preprocessing")
    return data_cleaned_new

data_cleaned_new = DataPreprocessing(path_unprocessed_data,path_processed_data)

# Create dataset with only duplicates (remove single touchpoints)
data_cleaned_duplicates = data_cleaned_new[
    data_cleaned_new.duplicated(subset=['sequence_id'],keep=False)]

# Write a pandas dataframe to zipped CSV file
data_cleaned_duplicates.to_csv(path_processed_data_duplicates,sep='\t')
data_cleaned_new.to_csv(path_processed_data,sep="\t")
```

B Code PNN and APNN

```
# Define location processed data and new location for PNN data input and output
path_processed_data_duplicates =
    r'/home/kdkruiff/Thesis/Data/Part1/criteo_dataset_cleaned_duplicates.tsv'
path_processed_data_input = r'/home/kdkruiff/Thesis/Data/Part2/PNN_data_input.tsv'
path_processed_data_output = r'/home/kdkruiff/Thesis/Data/Part2/PNN_data_output.tsv'

# Load processed data
data_cleaned_duplicates = pd.read_csv(path_processed_data_duplicates,sep='\t')

def tuple_transform(x):
    return tuple(map(int, x[1:-1].split(',␣')))

# ---------- CREATE SPARSE SEQUENCES (PNN) ----------
data_cleaned_duplicates = data_cleaned_duplicates.drop(
    columns=data_cleaned_duplicates.columns[0], axis =1)
PNN_data_output = pd.DataFrame(data_cleaned_duplicates.conversion)
PNN_data_output.to_csv(path_processed_data_output,sep='\t')

# Set number of categories and campaigns beside click level
categories_series = data_cleaned_duplicates.iloc[:,4].apply(tuple_transform).sum()
```

```
categories = sorted(set(categories_series))
campaigns = sorted(set(data_cleaned_duplicates.campaign))
n_columns = 1+len(campaigns)+len(categories)+1
data_input = np.zeros((data_cleaned_duplicates.shape[0], n_columns))
len_before_cat = 1 + len(campaigns)
len_after_cat = len_before_cat + len(categories)

for i in range(data_cleaned_duplicates.shape[0]):
    print(i)
    # Set click to 0 or 1
    data_input[i,0] = data_cleaned_duplicates.iloc[i,3]

    # Set campaign index to 1
    campaign_index = campaigns.index(data_cleaned_duplicates.iloc[i,1])
    data_input[i,campaign_index] = 1

    # Set categories involved with touchpoint to 1
    set_cat_tp = tuple(map(int,data_cleaned_duplicates.iloc[i,4][1:-1].split(', ')))
    for j in range(len(set_cat_tp)):
        cat_index = categories.index(set_cat_tp[j])

        data_input[i,len_before_cat + cat_index - 1] = 1

    # Add sequence and position id
    data_input[i,len_after_cat -1] = data_cleaned_duplicates.iloc[i,6]
    data_input[i,len_after_cat] = data_cleaned_duplicates.iloc[i,5]

# Save input data set
df_data_input = pd.DataFrame(data_input)
df_data_input.to_csv(path_processed_data_input,sep='\t')
```

C Code Model Evaluation

```
# Retrieve input and output data
path_data_input = r'/home/kdkruiff/Thesis/Data/Part2/PNN_data_input.tsv'
path_data_output = r'/home/kdkruiff/Thesis/Data/Part2/PNN_data_output.tsv'
PNN_data_input = pd.read_csv(path_data_input,sep='\t')
PNN_data_output = pd.read_csv(path_data_output,sep='\t')
PNN_data_input = PNN_data_input.drop(columns=PNN_data_input.columns[0], axis =1)
PNN_data_output = PNN_data_output.drop(columns=PNN_data_output.columns[0], axis =1)

# ———— TEST AND TRAINING SET ————
X_train_PNN, X_test_PNN, y_train_PNN, y_test_PNN =
        train_test_split(PNN_data_input, PNN_data_output, test_size = 0.2,
                         random_state = 1111, stratify = PNN_data_output)

# Construct conversion and non-conversion matrix and set everything to numpy
X_test = X_test_PNN.values
y_test = y_test_PNN.values
y_train = y_train_PNN.values

# —— MAIN ——
# Compact PNN, APNN and 'last touch' PNN model
```

```
def PNN_and_APNN(x, X_train, y_train):
    # ——— Pattern layer ———
    cos_matrix = cosine_similarity(x, X_train)
    cos_matrix_conv = cos_matrix[np.reshape(
        np.concatenate(y_train==1),(1,X_train.shape[0]))]
    cos_matrix_no_conv = cos_matrix[np.reshape(
        np.concatenate(y_train==0),(1,X_train.shape[0]))]
    cos_matrix_conv_shape = np.concatenate(cos_matrix_conv.reshape(
        cos_matrix_conv.shape[0],1))

    # ——— Summation layer ———
    norm_cos_matrix = cos_matrix / sum(cos_matrix[0])

    # ——> PNN
    p_conversion_PNN = sum(cos_matrix_conv) / cos_matrix_conv.shape[0]
    p_no_conv_PNN = sum(cos_matrix_no_conv) / cos_matrix_no_conv.shape[0]
    p_conv_PNN = softmax([p_conversion_PNN,p_no_conv_PNN])[0]

    # ——> APNN
    p_conversion_APNN = sum(cos_matrix_conv*norm_cos_matrix[
        np.reshape(np.concatenate(y_train==1),(1,X_train.shape[0]))]) /
        cos_matrix_conv.shape[0]
    p_no_conv_APNN = sum(cos_matrix_no_conv*norm_cos_matrix[
        np.reshape(np.concatenate(y_train==0),(1,X_train.shape[0]))]) /
        cos_matrix_no_conv.shape[0]
    p_conv_APNN = softmax([p_conversion_APNN, p_no_conv_APNN]) [0]

    # ——— Decision layer ———
    prediction_PNN = 0
    prediction_APNN = 0
    if p_conversion_PNN > p_no_conv_PNN:
        prediction_PNN = 1
    if p_conversion_APNN > p_no_conv_APNN:
        prediction_APNN = 1

    # ——— Max Similar ———
    prediction_max = y_train[np.argmax(cos_matrix[0], axis=0)]
    prediction_max_3 = np.bincount(np.concatenate(
            y_train[np.argpartition(cos_matrix[0], -3)[-3:]])).argmax()
    prediction_max_5 = np.bincount(np.concatenate(
            y_train[np.argpartition(cos_matrix[0], -5)[-5:]])).argmax()
    prediction_max_9 = np.bincount(np.concatenate(
            y_train[np.argpartition(cos_matrix[0], -9)[-9:]])).argmax()
    prediction_max_15 = np.bincount(np.concatenate(
            y_train[np.argpartition(cos_matrix[0], -15)[-15:]])).argmax()
    return cos_matrix_conv_shape, prediction_PNN, prediction_APNN, prediction_max,
            prediction_max_3, prediction_max_5, prediction_max_9, prediction_max_15,
            p_conv_PNN, p_conv_APNN

def softmax(x):
    """Compute softmax values for each sets of scores in x."""
    return np.exp(x) / np.sum(np.exp(x), axis=0)

# ———— RESULTS ————
# Evaluate prediction results
predictions = np.zeros((y_test.shape[0],10))
cos_matrix_conv_tot = np.zeros((sum(y_train==1)[0],))
for i in range(X_test.shape[0]):
    cos_matrix_conv, pred_PNN, pred_APNN, pred_max, pred_max3,
    pred_max5, pred_max9, pred_max15, p_conv_PNN, p_conv_APNN =
        PNN_and_APNN(X_test[i,].reshape(1,X_test.shape[1]),X_train_PNN,y_train)
    predictions[i,0] = pred_PNN
    predictions[i,1] = pred_APNN
    predictions[i,2] = pred_max
    predictions[i,3] = pred_max3
    predictions[i,4] = pred_max5
    predictions[i,5] = pred_max9
    predictions[i,6] = pred_max15
    predictions[i,7] = y_test[i]
    predictions[i,8] = p_conv_PNN
    predictions[i,9] = p_conv_APNN
    cos_matrix_conv_tot += cos_matrix_conv

# Histogram which has to be saved
cos_matrix_conv_tot_norm = cos_matrix_conv_tot /
    sum(cos_matrix_conv_tot)
hist_attributions = plt.hist(cos_matrix_conv_tot /
    sum(cos_matrix_conv_tot),bins=100,facecolor='grey')
plt.xlabel("Credit Attribution Weight")
plt.ylabel("Number of touchpoints")
plt.title("Credit Attribution")
plt.savefig(r'/home/kdkruiff/Thesis/CreditAttribution.png')

correct_PNN = 0
correct_APNN = 0
correct_APNN_max = 0
correct_APNN_max3 = 0
correct_APNN_max5 = 0
correct_APNN_max9 = 0
correct_APNN_max15 = 0
for i in range(predictions.shape[0]):
    if predictions[i,0] == predictions[i,7]:
        correct_PNN += 1
    if predictions[i,1] == predictions[i,7]:
        correct_APNN += 1
    if predictions[i,2] == predictions[i,7]:
        correct_APNN_max += 1
    if predictions[i,3] == predictions[i,7]:
        correct_APNN_max3 += 1
    if predictions[i,4] == predictions[i,7]:
        correct_APNN_max5 += 1
    if predictions[i,5] == predictions[i,7]:
        correct_APNN_max9 += 1
    if predictions[i,6] == predictions[i,7]:
        correct_APNN_max15 += 1

# ----- GENERATE TABLE WITH RESULTS -----
table_results = np.zeros((7,4))
```

```
# Accuracy
table_results[0,0] = correct_PNN / predictions.shape[0]
table_results[1,0] = correct_APNN / predictions.shape[0]
table_results[2,0] = correct_APNN_max / predictions.shape[0]
table_results[3,0] = correct_APNN_max3 / predictions.shape[0]
table_results[4,0] = correct_APNN_max5 / predictions.shape[0]
table_results[5,0] = correct_APNN_max9 / predictions.shape[0]
table_results[6,0] = correct_APNN_max15 / predictions.shape[0]
# F-measure
table_results[0,1] = f1_score(predictions[:,7], predictions[:,0], average='binary')
table_results[1,1] = f1_score(predictions[:,7], predictions[:,1], average='binary')
table_results[2,1] = f1_score(predictions[:,7], predictions[:,2], average='binary')
table_results[3,1] = f1_score(predictions[:,7], predictions[:,3], average='binary')
table_results[4,1] = f1_score(predictions[:,7], predictions[:,4], average='binary')
table_results[5,1] = f1_score(predictions[:,7], predictions[:,5], average='binary')
table_results[6,1] = f1_score(predictions[:,7], predictions[:,6], average='binary')
# AUC
# Compute Area Under Curve (ROC)
def AUC(y_true,y_prob):
    ROC_AUC_eval = roc_auc_score(y_true, y_prob)
    return ROC_AUC_eval
table_results[0,2] = AUC(predictions[:,7],predictions[:,8])
table_results[1,2] = AUC(predictions[:,7],predictions[:,9])
table_results[0,3] = log_loss(predictions[:,7],predictions[:,8])
table_results[1,3] = log_loss(predictions[:,7],predictions[:,9])

# Convert results to dataframe
df_predictions = pd.DataFrame(predictions)
df_table_results = pd.DataFrame(table_results)
df_cos_matrix_conv_arr = pd.DataFrame(cos_matrix_conv_tot_norm)

# ---- SAVE RESULTS ----
path_prediction_results = r'/home/kdkruiff/Thesis/Data/Part3/all_predictions.csv'
path_results_table = r'/home/kdkruiff/Thesis/Data/Part3/table_results.csv'
path_results_atrr = r'/home/kdkruiff/Thesis/Data/Part3/attribution_results.csv'
df_predictions.to_csv(path_prediction_results)
df_table_results.to_csv(path_results_table)
df_cos_matrix_conv_arr.to_csv(path_results_atrr)
```