Predicting conversions in display advertising based on URL embeddings

Yang Qiu
École Polytechnique & Jellyfish
yanq.qiu@jellyfish.com

Nikolaos Tziortziotis
Jellyfish
ntziorzi@gmail.com

Martial Hue
Jellyfish
martial.hue@jellyfish.com

Michalis Vazirgiannis
École Polytechnique
mvazirg@lix.polytechnique.fr

ABSTRACT

Online display advertising is growing rapidly in recent years thanks to the automation of the ad buying process. Real-time bidding (RTB) allows the automated trading of ad impressions between advertisers and publishers through real-time auctions. In order to increase the effectiveness of their campaigns, advertisers should deliver ads to the users who are highly likely to be converted (i.e., purchase, registration, website visit, etc.) in the near future. In this study, we introduce and examine different models for estimating the probability of a user converting, given their history of visited URLs. Inspired by natural language processing, we introduce three URL embedding models to compute semantically meaningful URL representations. To demonstrate the effectiveness of the different proposed representation and conversion prediction models, we have conducted experiments on real logged events collected from an advertising platform.

ACM Reference Format:
Yang Qiu, Nikolaos Tziortziotis, Martial Hue, and Michalis Vazirgiannis. 2020. Predicting conversions in display advertising based on URL embeddings. In *AdKDD '20, August 23, 2020, San Diego, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In online display advertising [22], advertisers promote their products by embedding ads on the publisher's web page. The majority of all these online display ads are served through Real-Time Bidding (RTB) [6]. RTB allows the publishers to sell their ad placements via the Supply-Side Platform (SSP) and the advertisers to purchase these via the Demand-Side Platform (DSP). More specifically, each time a user visits a website that contains a banner placement, an auction is triggered. The publisher sends user's information to the SSP, which forwards this information to the Ad exchange (AdX), and finally the AdX sends a bid request to the DSPs. Then each DSP decides if it will submit or not a bid response for this impression, based on its information about user, advertisement, urls, etc. Once the DSPs send back to the AdX their bids, a public auction takes place with the impression to be sold to the highest bidder. Figure 1 briefly illustrates the procedure of online display advertising.

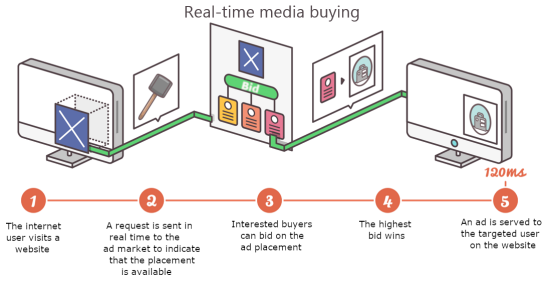

Figure 1: A high-level overview of RTB procedure.

DSPs are agent platforms that help advertisers optimise their advertising strategies. Roughly speaking, DSPs try to estimate the optimal bid price for an ad impression in order to maximise the audience of the campaigns of their advertisers, given some budget constraints. The bid price of an ad impression is highly related to the additive value that this impression could have on the advertising campaign (i.e., the number of ad impressions, clicks or conversions, etc.). In this context, advertisers have at their disposal a number of different pricing models. In the case where the objective of the advertisers is to maximise the exposure of their advertising message to a targeted audience, paying per impression, referred as *cost-per-mille* (CPM), is probably the best option for them. Nevertheless, in most of the cases, performance display advertising is more attractive to advertisers that are interested in accomplishing specific goals reducing their risks. In this case, advertisers are willing to pay for an ad impression if and only if that impression will drive the user to take a predefined action/conversion [14], such as a visit on the advertiser's website, a purchase of a product, etc. Two performance payment models have been introduced for this purpose, referred as *cost-per-click* (CPC) and *cost-per-action* (CPA).

In performance-driven display advertising, DSPs submit a bid for a given ad impression based on the CPC or CPA that the advertiser is willing to pay. To determine the optimal bid price for an ad impression, DSPs estimate the *expected cost per impression*, called eCPI, which is either equal to the click-through-rate (CTR) for this impression multiplied by the value of CPC, or the conversion rate (CVR) multiplied by the value of CPA [5]. As a result, accurate CTR/CVR prediction plays an important role in the success of online advertising. For this purpose, DSPs build CTR/CVR prediction models able to estimate the probability a user converting after their exposure to an advertisement. The accuracy of these models is of high importance for the success of the campaign as if we overestimate click or conversation rates, we will probably submit quite higher bids than we should do, winning possible useless ad impressions. On the other hand, if these rates are underestimated, we will probably miss ad impressions likely to lead to a user action.

AdKDD '20, August 23, 2020, San Diego, CA, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In this work we examine the user conversion problem, where given an advertiser, we want to predict if a user will be converted or not based on their history. In contrast to previous works that use a number of features related to the user profile, ad information and context information, we consider only the user's browsing history. More specifically, each user is represented as a sequence of URLs visited by the user in a single day. Therefore, the problem examined in this paper can be formally described as: given a user's sequence of URLs from a single day, predict the probability this user to take a predefined action on the next day. In our case, a user is considered as *converted* if they visit the advertiser's website. Due to the high cardinality and diversity of URLs, a compact semantically meaningful representation of URLs is of high importance. For this purpose, we build and examine three URL embedding models following the idea of word embeddings [16]. The sequential dependency between user's browsing history has also been considered by using a Recurrent Neural Network (RNN) [8]. In total, ten different prediction conversion models have been introduced. A number of large scale experiments have been executed on a data collected from a real-world advertising platform in order to reveal and compare the prediction abilities of the proposed prediction schemes. Finally, our empirical analysis validates our claims about the effectiveness of our representation models showing that they achieve to group together URLs of the same category. It means that URLs with the same or similar context are also close on the embedding space.

2 RELATED WORK

As the performance of a campaign is directly related on how precisely the CVR/CTR is estimated, it has been the objective of considerable research in the past few years. Typically, the problem of CVR/CTR estimation is formulated as a standard binary classification problem. Logistic regression has been extensively used to accurately identify conversion events [4, 15, 18]. [7] introduced a Bayesian learning model, called Bayesian probit regression, which quantifies the uncertainty over a model's parameters and hence about the CTR of a given ad-impression. A precise user representation (a set of features describing user behaviour) constitutes also the foundation for building a linear model able to estimate CVR with high accuracy. Nevertheless, in most cases it requires a lot of feature engineering effort and the knowledge of the domain. Moreover, linear models are not capable to reveal the relationship among feature. To overcome this problem, a number of non-linear models such as factorisation machines [17, 20] and gradient boosted regression trees [10] have been also proposed to capture higher order information among features. A number of different deep learning methods have been also proposed recently for CTR prediction [3, 13, 23, 24].

Representation learning has been applied with success in several applications and has become a field in itself [1] in the recent years. The URL representation architectures presented in this manuscript have been inspired by those used in natural language processing (NLP) tasks. Learning high-quality representations of phrases or documents is a long-standing problem in a wide range of NLP tasks. *Word2Vec* [16] is one of the most well-known word embeddings algorithms. The main idea behind *Word2Vec* is that words that appear in similar contexts should be close in the learned embedding space. For this purpose, a (shallow) neural network language model is applied that consists of an input, a projection, and an output layer. Its simple architecture makes the training extremely efficient. [9] proposed *search2vec* model that learns user search action representations based on contextual co-occurrence in user search sessions. To the best of our knowledge, URLNet [12] is the only work that learns a URL representation but for the task of malicious URL detection. In contrast to our unsupervised representation scheme that considers the sequential order of URLs, URLNet is an end-to-end (supervised) deep learning framework where its character-level and word-level CNNs are jointly optimized to learn the prediction model.

3 PROPOSED CONVERSION PREDICTION ARCHITECTURE

The goal of this paper is to predict the probability a user to be converted one day after, given their browsing history on a single day. More specifically, we consider each user as an ordered sequence of URLs, sorted chronologically. The notion of conversion corresponds to an action of interest for the advertiser, such as visit on the landing page, purchase of a product, registration, etc. Therefore, we can treat the problem of predicting the user conversion as a binary classification problem [2], where given a sequence of URLs visited by a user $U_n = \{url_1^n, \ldots, url_{T_n}^n\}$, $n = 1, 2, \ldots, N$, we want to predict if U_n will be converted or not, $y_n \in \{0, 1\}$. The length of the URL sequence, T_n, may be different for each user.

As an analogy to text classification, we view a sequence of URLs as a document, or a sequence of sentences. In our case, a URL is itself a sequence of tokens, of length at most three (we ignore the rest tokens as they are quite noisy). Each URL[1] is split with a '/' (slash) character, where the first token corresponds to the domain name. For instance, https://en.wikipedia.org/wiki/Main_Page is mapped to [`en.wikipedia.org`, `wiki`, `Main_Page`].

In order to apply any supervised classification model, such as logistic regression, etc., a semantically meaningful representation of each URL is needed. Therefore, a key intermediate step in our model is the learning of a URL representation. More precisely, the proposed conversion prediction scheme is composed of two consecutive training phases. The first one corresponds to the learning of the URL representations, while the second one corresponds to the training of a classifier. It should be mentioned that the training processes of these two models are independent.

Due to the high cardinality of URLs, we learn the URL representations implicitly by learning and aggregating their tokens representations. In this study, we present and examine four different URL representation models, f_r: $url \rightarrow \boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{R}^d$ and d is the dimensionality of the embedding space. The first one is the simple *one-hot encoding* that treats tokens as atomic units. The main limitation of this representation is that the representation size grows linearly with the corpus and it doesn't consider the similarity between URLs. To overcome these issues, we propose three different URL embedding models (see Section 4).

After having trained a representation model, f_r, and given a training set $\mathcal{D} = \{(U_n, y_n)\}_{n=1}^N$, we produce a new dataset $\mathcal{D}' = \{(X_n, y_n)\}_{n=1}^N$ where $X_n = \{\boldsymbol{x}_1^n, \ldots, \boldsymbol{x}_{T_n}^n\}$ is a sequence of length T_n

[1] The `http(s)://` and `www` parts of each URL are stripped.

with elements $\boldsymbol{x}_i^n = f_r(url_i^n)$. In a nutshell, $\mathcal{D}'$ contains sequences of URL embedding vectors along with their labels. Then, we apply mapping $f_m: X \rightarrow \boldsymbol{z}$ that aggregates the URLs embeddings into an embedding vector $\boldsymbol{z} \in \mathbb{R}^m$, where m can be different from d. It results in a single compact representation $\boldsymbol{z}$ for each sequence of URLs. Next, our goal is to discover a classification model $f_c: \boldsymbol{z} \rightarrow \hat{y}$ from a set $\mathcal{F}$ of possible models with the minimum empirical risk

$$\min_{f_c \in \mathcal{F}} \mathbb{E}_{(X,y)\sim\mathcal{D}'}[\ell(f_c(f_m(X)), y)], \tag{1}$$

where $\ell \in \mathbb{R}$ is a non-negative loss function. In fact, classifier f_c is trained on dataset $\mathcal{D}'' = \{(\boldsymbol{z}_n, y_n)\}_{n=1}^N$. In this work, we use logistic regression where the conversion conditional probability of user (n) given their browsing history X_n, is modeled as:

$$p(y_n = 1|X_n) = \sigma(\boldsymbol{\theta}^\top f_m(X_n) + b), \tag{2}$$

where $\boldsymbol{\theta}$ is a vector with the unknown model parameters, b is the bias term and $\sigma(\cdot)$ is the *logistic sigmoid* function. From a geometrical point of view, $\boldsymbol{\theta}^\top f_m(X_n) + b$ is a hyperplane that separates the two classes. To learn the unknown model parameters, the cross-entropy loss is applied:

$$\mathcal{L} = -\mathbb{E}_{(X,y)\sim\mathcal{D}'}[y \log f_c(f_m(X)) + (1-y)\log(1-f_c(f_m(X)))]. \tag{3}$$

Next, we are describing the three different mapping functions f_m adopted in our work. The first one returns the average of the URLs embedding vectors presented on a sequence: $f_m^{(1)}(X) = \frac{1}{T}\sum_{i=1}^T \boldsymbol{x}_i$. The second one considers the dependencies among the features of the embedding vector returned by the first mapping function. To be more precise, it returns the output of a dense layer with rectified linear units (ReLU), that takes as input the average of the URLs embedding vectors, $f_m^{(2)}(X) = g(\boldsymbol{\theta}^{(1)\top} f_m^{(1)}(X) + b^{(1)})$. The ReLU uses the activation function $g(z) = \max\{0, z\}$. The main limitation of applying one of the two aforementioned mappings is that they do not take into account the chronological order in which the URLs appeared on the sequence. To overcome this limitation we resort to the well-known Long Short Term Memory network (LSTM) [11] that is a special kind of RNNs [19] and is suitable to process variable-length sequences. To be more precise, the third mapping function $f_m^{(3)}$ is an LSTM network able to map an input sequence of URL embeddings X to a fixed-sized vector $\boldsymbol{z}$, that can be considered as the representation of the whole sequence. In all cases, we are feeding the produced vector $\boldsymbol{z}$ to a final dense layer with sigmoid activation function. In the rest of the paper, we denote as `LR`, `DLR` and `RNN` the prediction conversion models which are using the "average", "dense" and "LSTM" mapping functions, respectively. A graphical illustration of the proposed conversion prediction model architecture is presented at Fig.2.

4 URL REPRESENTATION SCHEMES

This section introduces the four URL representation models proposed in our work. These models can be divided in two categories: i) one-hot encoding, and ii) embedding representation. There is no need for learning in the case of one-hot encoding. On the other hand, the embedding representations of URL tokens are learned in advance and then used to form the final URL representation. A representation is also learned for the so-called "rare" and "none" tokens, respectively. A token is considered "rare" if it is present less

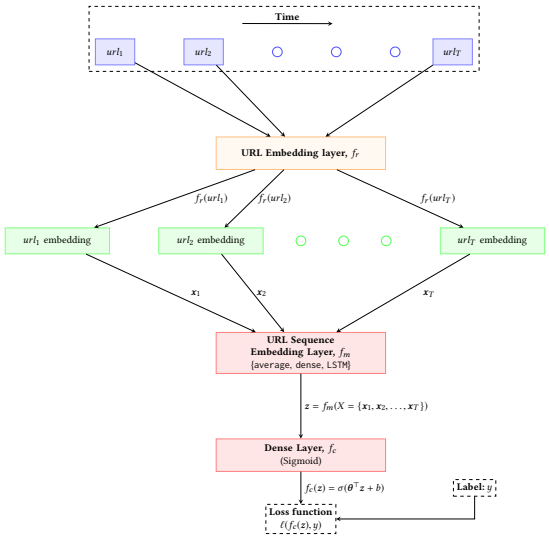

Figure 2: The proposed conversion prediction model architecture. It consists of three parts: i) URL embedding layer (f_r), ii) URL sequence embedding layer (f_m), and iii) Logistic regression classifier (f_c). Only the unknown classifier parameters (Eq. 2) of the dense layer and these of LSTM and "dense" mappings are trainable.

than a predefined number (we set it equal to 20) of times in our data. We denote the token embedding vectors as $\boldsymbol{e}$.

One-hot encoding. First, we introduce a variant of the standard one-hot encoding for representing URLs that is our baseline. As already mentioned, the token representations are used to get the URL representation. Therefore, in our case, the cardinality of the one-hot encoding is equal to the number of all possible tokens appearing in our data. Given a one-hot encoding $\{\boldsymbol{e}_t\}_{t=1}^{n\_tokens \leq 3}$, for each one of the tokens appearing in url, we take their average to encode it: $f_r(url) = \frac{1}{n\_tokens}\sum_{t=1}^{n\_tokens} \boldsymbol{e}_t$.

Embedding learning. Despite its simplicity, the aforementioned one-hot encoding does not take into account the similarity between URLs, while on the same time the representation size grows linearly with the corpus. In fact, one-hot encoding is sparse (curse of dimensionality) and it is not able to capture the distance between individual urls. To tackle this problem, we propose three representation schemes inspired by the idea of *Word2Vec* [16].More specifically, our representation schemes use the *skip-gram* model that given a target word (URL in our case) tries to predict its context words. More formally, the *skip-gram* model tries to find word representations that can be used for predicting the words in its neighborhood. Therefore, given a sequence of words (URLs) $url_1, \ldots, url_T$, our objective is the maximisation of the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(url_{t+j}|url_t), \tag{4}$$

where c specifies the neighborhood of target URL. The conditional probability is defined by using the softmax function, as $p(url_c|url_t) \triangleq$

$\frac{\exp(\boldsymbol{x}_c^\top \boldsymbol{x}_t)}{\sum_{c' \in C} \exp(\boldsymbol{x}_{c'}^\top \boldsymbol{x}_t)}$, where $\boldsymbol{x}_c$ and $\boldsymbol{x}_t$ are the representations for context and target URLs respectively, and C is the set of unique words.

As the direct optimisation of Eq. 4 is computationally expensive, we are adopting the *negative sampling* approach [16]. In negative sampling, we treat the word's representation learning as a binary classification task where we try to distinguish the target-context pairs of words presented on the training data from those that are not. Following the suggestions of [16], for each positive pair we are creating k negative (target-context) pairs.

In contrast to the original *Word2Vec* model, the proposed representation learning architectures try to learn the tokens representations, instead of the URL representations directly. A token representation is also learned for the case of a `Pad` token. For instance, https://en.wikipedia.org/ is mapped to [`en.wikipedia.org`, `Pad`, `Pad`]. Since URLs are padded, the number of tokens of each URL is equal to three. Then, by combining the token representations we form the final URL representation that will be used for the training of the conversion prediction classifier. Actually, the second phase of our model (conversion classifier) can be seen as a way to test the effectiveness of our representation models. The main difference between the three proposed embedding representation models is how the token representations are combined to form the final URL embedding vector:

- `Domain_only` representation uses only the representation of the first token to represent the URL, ignoring the representations of the other two tokens.
- `Token_avg` representation takes the average of the token embedding vectors to represent the URL.
- `Token_concat` representation concatenates the token embedding vectors to represent the URL. In this case, the dimension of the URL embedding vector is three times the dimension of the token embedding vectors.

For instance, let $\{\boldsymbol{e}_t\}_{t=1}^3$ be the token embedding vectors of the three tokens presented on $url = [token_1, token_2, token_3]$. Then, the `Domain_only` representation $\boldsymbol{x}$ is equal to $\boldsymbol{e}_1$, the `Token_avg` representation is equal to $\frac{1}{3}\sum_{t=1}^3 \boldsymbol{e}_t$, and the `Token_concat` representation is given as $[\boldsymbol{e}_1^\top, \boldsymbol{e}_2^\top, \boldsymbol{e}_3^\top]^\top$. A graphical illustration of the proposed embedding learning architecture is provided at appendix.

5 EXPERIMENTS

This section presents the results of our empirical analysis. A real-world RTB dataset was used to train and analyse the performance of the ten proposed prediction models. We built our dataset by using the auction system logs from campaigns launched in France. It should be also mentioned that our dataset is anonymised, and only visited URLs are considered. In this way, each record of the dataset corresponds to a chronologically ordered sequence of visited URLs along with a binary label (specific to the advertiser) that indicates whether or not a conversion has happened on the advertiser's website on the next day. More precisely, the data composed of sequences of URLs along with their labels of three successive dates, $\mathcal{D}_d$, $\mathcal{D}_{d+1}$, and $\mathcal{D}_{d+2}$, where $\mathcal{D}_d$ is used for learning representations, and $\mathcal{D}_{d+1}$ and $\mathcal{D}_{d+2}$ for training and testing the prediction models, respectively. Moreover, the maximum length of a URL sequence is set equal to 500, where only the most recently visited URLs are kept in each sequence, $T_n \leq 500, \forall n \in [1, N]$. In total we examine the performance of the models on five advertisers, belonging to four different categories: banking, e-shop, newspaper, and telecommunications (see Table 1). Details about the statistics of the data are provided in the appendix [url_link] (https://bit.ly/3cSNFXU).

Table 1: Number of converted vs. non-converted records for each one of the 5 advertisers on the training and testing data.

Advertiser Category	**Training** (5, 452, 577)	**Testing** (7, 164, 109)
Banking	(3, 746 − 5, 448, 831)	(8, 539 − 7, 155, 570)
E-shop	(1, 463 − 5, 451, 114)	(1, 821 − 7, 162, 288)
Newspaper_1	(1, 406 − 5, 451, 171)	(2, 923 − 7, 161, 186)
Newspaper_2	(1, 261 − 5, 451, 316)	(1, 291 − 7, 162, 818)
Telecom	(1, 781 − 5, 450, 796)	(2, 201 − 7, 161, 908)

5.1 Settings

All our predictors assume the existence of a URL representation model in order to vectorise the sequence of URLs for each one of the dataset records. Our baseline, `One_hot/LR`, represents the URL sequences using a one-hot encoding vector of size 193, 409, where the first two entries correspond to the "unknown" and "rare" tokens, respectively. The other models rely on an already trained embedding matrix. Each token is embedding into a 100-dimensional vector. The first three rows correspond to the "unknown", "rare", and `Pad` tokens, respectively. The rest rows contain the embedding vector of all non-rare tokens observed in the dataset $\mathcal{D}_d$. The number of non-rare tokens is 22, 098 for the `Domain_only`, and 187, 916 for both the `Token_Average` and `Token_Concatenation`. To train representations, we have considered two different *{pos:neg}* ratios ({1:1} and {1:4}). Due to page limitations, only the results of the {1:4} ratio are presented in this manuscript. See supplementary material [url_link] for a full comparison between these two ratios.

The number of units of the hidden dense layer (dimensionality of its output space) of `DLR` model is set to 30. Furthermore, the number of hidden units of LSTM is set to 10 on the `RNN` model. A dense layer with a sigmoid activation function is applied at the end of each one of the three mapping functions f_m ("average", "dense", "LSTM") in order to form a binary classifier. Through our empirical analysis we have observed that the `DLR` and `RNN` prediction models are prone to overfitting. To enhance the generalization capabilities of these two models, we are using dropout that is set to 0.5. To be more precise, a dropout layer is added right after the f_m layer of the `DLR` model, while both the dropout and the recurrent_dropout parameter of LSTM layer are set equal to 0.5 on the `RNN` model.

For the training of all representation and prediction models, the mini-batch stochastic optimization has been applied by using Adam optimizer with the default Tensorflow 2.0 settings (i.e., lr=0.001, $\text{beta}_1 = 0.9$, $\text{beta}_2 = 0.999$). More precisely, to train the representation models we are doing one full pass over the whole data that is divided to 200 parquet files. The total number of epochs is 200, equal to the number of parquet files. At each epoch we are producing the positive and negative pairs based on the data contained on a single parquet file and are feeding them to the representation model. On the other hand, the size of batches for training prediction models is 64, while the number of epochs and number of steps per epoch is set to 100 in both cases. To tackle the problem of our unbalanced dataset (see Table 1), the ratio of positive and negative records is {1:1} in the batches used for the training of the classifiers.

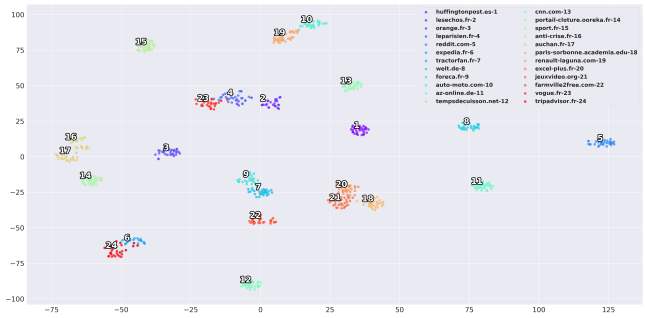

Figure 3: t-SNE visualization of the thirty closest neighbors of 24 different domains.

Table 2: The 10-nearest neighbors of 24 different domains according to our trained `Domain_only` representation model.

Domain	10-nearest neighbors
huffingtonpost.es	cope.es; m.eldiario.es; okdiario.com; verne.elpais.com; blogs.elconfidencial.com; vozpopuli.com; elespanol.com; smoda.elpais.com; libertaddigital.com; cadenaser.com
lesechos.fr	latribune.fr; afrique.latribune.fr; business.lesechos.fr; bfmbusiness.bfmtv.com; financedemarche.fr; challenges.fr; investopedia.com; actufinance.fr; lopinion.fr; contrepoints.org
orange.fr	actu.orange.fr; lemoteur.orange.fr; messagerie.orange.fr; login.orange.fr; finance.orange.fr; sports.orange.fr; meteo.orange.fr; tendances.orange.fr; programme-tv.orange.fr; news.orange.fr
leparisien.fr	cnews.fr; atlasinfo.fr; lefigaro.fr; lejdd.fr; jforum.fr; marianne.net; video.lefigaro.fr; tendanceouest.com; bladi.net; observalgerie.com
reddit.com	imgur.com; old.reddit.com; askreddit.reddit.com; pcgamer.com; anime.reddit.com; france.reddit.com; gamefaqs.gamespot.com; totalwar.reddit.com; nintendoswitch.reddit.com; gaming.reddit.com
expedia.fr	momondo.fr; skyscanner.fr; kayak.fr; fr.lastminute.com; fr.hotels.com; flights-results.liligo.fr; ebookers.fr; esky.fr; opodo.com; secure.lastminute.com
tractorfan.fr	discountfarmer.com; forum.farm-connexion.com; angleterre.meteosun.com; songs-tube.net; materieltp.fr; assovttroc.clicforum.fr; opel-mokka.forumpro.fr; spa-du-dauphine.fr; vanvesactualite.blog4ever.com; calcul-frais-de-notaire.fr
welt.de	zeit.de; sueddeutsche.de; faz.net; tagesspiegel.de; sport1.de; kicker.de; saarbruecker-zeitung.de; tz.de; bild.de; sportbild.bild.de
foreca.fr	my-meteo.com; fr.meteovista.be; fr.tutiempo.net; meteopassion.com; de.sat24.com; nosvolleres.com; meteo-sud-aveyron.over-blog.com; xn--mto-bmab.fr; palombe.com; calculerdistance.fr
auto-moto.com	caradisiac.com; largus.fr; news.autojournal.fr; test-auto.auto-moto.com; auto-mag.info; feline.cc; motorlegend.com; es-sais.autojournal.fr; automobile-magazine.fr; turbo.fr
az-online.de	abountifulkitchen.com; thesurvivalgardener.com; leinetal24.de; brittanyherself.com; symbols.com; ourpaleolife.com; msl24.de; milliondollarjourney.com; arthritis-health.com; thehollywoodunlocked.com
tempsdecuisson.net	cuisine-facile.com; temps-de-cuisson.info; yummix.fr; aux-fourneaux.fr; cuisinenligne.com; audreycuisine.fr; mamina.fr; une-plumedansacuisine.com; cnz.to; ricardocuisine.com
cnn.com	us.cnn.com; stadiumtalk.com; thedailybeast.com; itpro.co.uk; uk.reuters.com; euronews.com; theargus.co.uk; theatlantic.com; thedailymash.co.uk; trendscatchers.co.uk
portail-cloture.ooreka.fr	bricolage-facile.net; mur.ooreka.fr; pierreetsol.com; bricolage.jg-laurent.com; abri-de-jardin.ooreka.fr; aac-mo.com; fr.rec.bricolage.narkive.com; decoration.ooreka.fr; piscineinfoservice.com; bricoleurpro.com
sport.fr	infomercato.fr; parisfans.fr; topmercato.com; vipsg.fr; footradio.com; mercatofootanglais.com; le10sport.com; buzzsport.fr; foot-parisien.com; foot-sur7.fr
anti-crise.fr	clid.fr; forum.anti-crise.fr; gesti-odr.com; echantillonsclub.com; plusdebonsplans.com; cataloguemate.fr; promoalert.com; argent-dubeurre.com; madstef.com; forum.madstef.com
auchan.fr	but.fr; conforama.fr; vente-unique.com; rueducommerce.fr; fr.shopping.com; cdiscount.com; touslesprix.com; promobutler.be; webmarchand.com; mistergoodeal.com
paris-sorbonne.academia.edu	flux-info.fr; elmostrador.cl; makaan.com; univ-montp3.academia.edu; e-lawresources.co.uk; babycenter.com; newocr.com; in-sight.co.kr; grandes-inventions.com; police-scientifique.com
renault-laguna.com	megane3.fr; gps-carminat.com; megane2.superforum.fr; lesamisdudiag.com; car-actu.com; diagnostic-auto.com; r25-safrane.net; lesamisdelaprog.com; forum.autocadre.com; renault-clio-4.forumpro.fr
excel-plus.fr	tech-connect.info; thehackernews.com; lecompagnon.info; panoptinet.com; slice42.com; aliasdmc.fr; astuces.jeanviet.info; nalaweb.com; patatos.over-blog.com; jiho.com
jeuxvideo.org	alsumaria.tv; minecraft-zh.gamepedia.com; infovisual.info; everyonepiano.com; footstream.live; memedroid.com; darkand-light.gamepedia.com; mbti.forumactif.fr; gachagames.net; honga.net
farmville2free.com	goldenlifegroup.com; fv2freegifts.org; juegossocial.com; fv-zprod-tc-0.farmville.com; fb1.farm2.zynga.com; zy2.farm2.zynga.com; gameskip.com; fv-zprod.farmville.com; megazebra-facebook-trails.mega-zebra.com; farmvilledirt.com
vogue.fr	vanityfair.fr; vogue.com; vivreparis.fr; fr.metrotime.be; brain-magazine.fr; o.nouvelobs.com; parismatch.be; pariszigzag.fr; ad-magazine.fr; umlad.co.uk
tripadvisor.fr	fr.hotels.com; cityzeum.com; voyages.michelin.fr; lonelyplanet.fr; monmuage.fr; voyageforum.com; rome2rio.com; toocamp.com; virail.fr; partir.com

5.2 Results

In this section, we formally present the results of our empirical analysis. Firstly, to get an intuition about the ability of the three introduced URL embedding schemes (Sec. 4) to group together URLs that belong to the same category (i.e., sports, news, etc.), we will visualise (Fig. 3) the embedding vectors of 24 selected domains along with those of the thirty closest URLs of each one of them. To be more precise, we are using the embedding matrix learned by the `Domain_only` model. Cosine similarity has been used to measure the similarity between the embedding vectors of two URLs. To project the original high-dimensional embedding vectors on a 2-dimensional space, we apply the Barnes-Hut t-SNE algorithm [21].

To be guaranteed that the URLs belong to the same category with that of their closest domain, Table 2 provides the 10-nearest URLs for each one of the 24 domains (see appendix for the full list of the 30-nearest neighbors). For instance, all URLs that are on the neighborhood of expedia.fr are about *travelling*. Moreover, all the neighbors of sport.fr (16) are URLs about *sports*. The visualization of Fig. 3 illustrates the ability of our model to produce semantically meaningful embeddings. Actually, it becomes apparent that we are getting 24 clearly distinguished clusters. Moreover, we can see that the clusters of 'similar' domains are also close on the embedding space. For instance, the URLs embeddings of clusters (10) [auto-moto.com] and (19) [renault-laguna.com] are close as they are related to the *automobile* category. The same also holds for the URLs embeddings of clusters (2) [lesechos.fr], (4) [leparisien.fr], and (23) [vogue.fr] that belong to the *news* category.

Table 3: Avg (%) and std of the area under ROC curves (5 independent runs) of the 10 prediction models on 5 advertisers.

Method \ Adv	Banking	E-shop	Newspaper_1	Newspaper_2	Telecom
`One_hot/LR`	65.7 ± 0.093	66.4 ± 0.053	75.5 ± 0.379	73.3 ± 0.400	65.4 ± 0.085
`Domain_only/LR`	64.6 ± 0.300	66.6 ± 0.217	75.7 ± 0.507	73.2 ± 0.345	63.2 ± 0.168
`Domain_only/DLR`	69.0 ± 0.214	69.7 ± 0.234	76.8 ± 0.342	75.7 ± 0.658	66.6 ± 0.303
`Domain_only/RNN`	71.4 ± 0.144	72.6 ± 0.422	80.3 ± 0.168	79.4 ± 0.281	71.2 ± 0.250
`Token_avg/LR`	64.5 ± 0.241	67.2 ± 0.390	76.4 ± 0.152	73.1 ± 0.184	62.9 ± 0.468
`Token_avg/DLR`	69.4 ± 0.294	72.1 ± 0.263	79.2 ± 0.242	77.7 ± 0.274	67.9 ± 0.348
`Token_avg/RNN`	**71.9** ± 0.082	**73.1** ± 0.246	**81.2 ± 0.153**	**80.2 ± 0.322**	**71.8 ± 0.153**
`Token_concat/LR`	64.8 ± 0.241	67.2 ± 0.060	76.7 ± 0.179	73.4 ± 0.273	63.6 ± 0.425
`Token_concat/DLR`	69.1 ± 0.222	70.8 ± 0.400	78.2 ± 0.285	76.7 ± 0.255	66.9 ± 0.310
`Token_concat/RNN`	71.5 ± 0.224	72.5 ± 0.460	80.5 ± 0.192	79.1 ± 0.130	70.5 ± 0.278

Next, we formally present the numerical results of the ten prediction models on five different advertisers. The first part of the name of each model specifies the type of URL representation used, while the second one indicates the classifier type. To evaluate and compare the effectiveness of the models we are using the area under ROC curve (AUC) metric. More specifically, we consider the average (%) AUC across five independent runs (see Table 3), where each run corresponds to a specific seed used for the models initialisation. Moreover, Figure 4 illustrates the average ROC curves of the prediction models for each advertiser. The right plot of Fig. 4 illustrates the AUC of the models on the 25 independent runs (5 advertisers × 5 runs for each advertiser)).

Based on the results presented in Table 3, the `Token_avg/RNN` model is more effective in predicting user's conversions compared to the rest models. Precisely, the `Token_avg/RNN` model has the highest average AUC in all advertisers. All models achieve their highest performance on the newspaper advertisers. It is also worth noticing that the performances of `Domain_only/LR`, `Token_avg/LR`, and `Token_concat/LR` are highly competitive, compared to our baseline, `One_hot/LR`. More specifically, `Token_concat/LR` clearly outperforms `One_hot/LR` in 2 out of 5 advertisers (E-shop, Newspaper_1), and has slightly better performance in one of them (Newspaper_2). That remark validates our claims that the proposed representation models produce meaningful embeddings, by distinguishing URLs of the same category and placing them close to the embedding space. Taking a closer look at the standard deviations, we can see that the performance of `One_hot/LR` is quite stable performance in the three advertisers. This was expected as the `One_hot` representation is identical over all runs, and therefore the only variation of the performance of `One_hot/LR` comes only from the training of the `LR` classifier. The performance of `LR` model is almost the same for each representation model. The same also holds for `DLR` and `RNN` model where its performance is more or less the same in the cases of `Domain_only` and `Token_concat`, and slightly better in the case of `Token_avg`. On the other hand, `DLR` performs significantly better when it is combined with `Token_avg` and `Token_concat`, with `Token_avg` to be more preferable (around 1% gain).

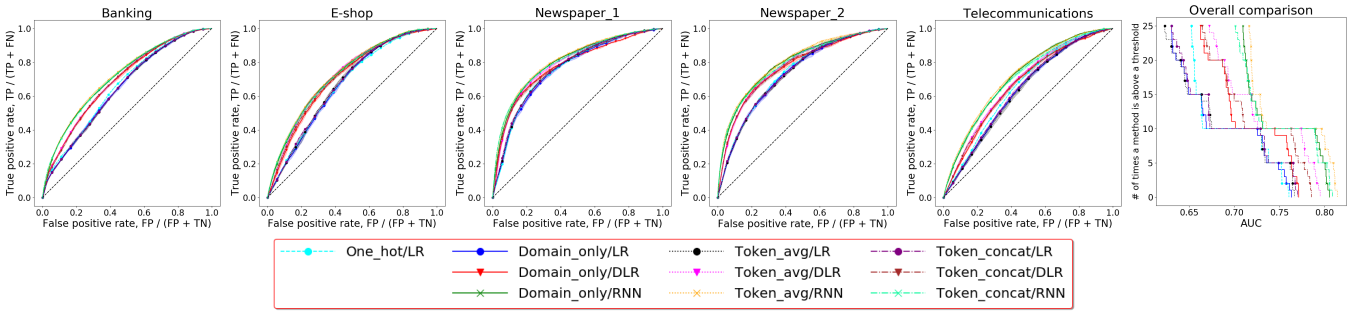

Figure 4: Average ROC curves of the ten conversion prediction models on the five advertisers. Shaded regions represent the std over 5 independent runs. The bottom right plot presents the AUC for each one of the 25 runs (5 advertisers × 5 independent runs for each advertiser) of each model. The •, ▼ and × marks indicate the LR, DLR and RNN classification models, respectively.

Let us now compare the impact of the type of classifier on the performance of the prediction model. The overall comparison presented at Fig. 4 demonstrates that both DLR and RNN performs significantly better compared to LR, with the RNN to be the best one. More precisely, the AUC of RNN is around ∼7% and ∼3% higher compared to those of LR and DLR, respectively. This means that the consideration of the chronological order in which the URLs appeared on the sequence is of high importance. On the other hand, choosing DLR over LR improves around ∼4% the performance of the prediction models independent to the representation model.

To sum up, the main conclusions of our empirical analysis are: i) all three proposed URL embedding models are able to learn high-quality vector representations that precisely capture the URL relationships, ii) the performance of the LR model is relatively invariant to the selection of the representation model, iii) among the three representation models, Token_avg is more adequate to capture the relationship between URLs, with the Token_concat second best, iv) the consideration of the chronological order of the visited URLs (RNN) and the learning of dependencies among the embedding features (DLR) are also of high importance as both improve significantly the performance of the prediction model.

6 CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we considered the problem of user response prediction in display advertising. Ten conversion prediction models were proposed to predict user response based on their browsing history. To represent the sequence of visited URLs, four different compact URL representations were examined. The effectiveness of the proposed models has been experimentally demonstrated offline in a real-world RTB dataset for five different advertisers. The impact of the sequential dependency between users' visited URLs on the performance of the predictors has been also examined. The main conclusions of our empirical analysis were that all three proposed representation models produce a meaningful URL representation, and considering the chronological order of the visited URLs by using RNN significantly improves the model's performance. In the future, we intend to propose an online version of our framework and to extend our empirical analysis to a real-world online scenario.

REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.
[2] C. M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag.
[3] P. P. K. Chan, X. Hu, L. Zhao, D. S. Yeung, D. Liu, and L. Xiao. 2018. Convolutional Neural Networks based Click-Through Rate Prediction with Multiple Feature Sequences. In *IJCAI*.
[4] O. Chapelle, E. Manavoglu, and R. Rosales. 2014. Simple and Scalable Response Prediction for Display Advertising. *ACM Trans. Intell. Syst. Technol.* 5, 4 (2014).
[5] Ye Chen, Pavel Berkhin, Bo Anderson, and Nikhil R. Devanur. 2011. Real-time Bidding Algorithms for Performance-based Display Ad Allocation. In *KDD*.
[6] Google. 2011. The arrival of real-time bidding.
[7] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. 2010. Web-scale Bayesian Click-through Rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine. In *ICML*.
[8] A. Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence, Vol. 385. Springer.
[9] M. Grbovic, N. Djuric, V. Radosavljevic, F. Silvestri, R. Baeza-Yates, A. Feng, E. Ordentlich, L. Yang, and G. Owens. 2016. Scalable Semantic Matching of Queries to Ads in Sponsored Search Advertising. In *SIGIR*.
[10] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *ADKDD*.
[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[12] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi. 2018. URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection. *CoRR* (2018).
[13] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A Convolutional Click Prediction Model. In *CIKM*.
[14] Mohammad Mahdian and Kerem Tomak. 2007. Pay-per-action Model for Online Advertising. In *ADKDD*.
[15] H. Brendan McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica. 2013. Ad Click Prediction: a View from the Trenches. In *KDD*.
[16] T. Mikolov, I. Sutskever, K. Chen, G. S Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*.
[17] Richard J. Oentaryo, Ee-Peng Lim, Jia-Wei Low, David Lo, and Michael Finegold. 2014. Predicting Response in Mobile Advertising with Hierarchical Importance-aware Factorization Machine. In *WSDM*.
[18] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting Clicks: Estimating the Click-through Rate for New Ads. In *WWW*.
[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1988. Neurocomputing: Foundations of Research. MIT Press, Chapter Learning Representations by Back-propagating Errors, 696–699.
[20] A. Ta. 2015. Factorization machines with follow-the-regularized-leader for CTR prediction in display advertising. In *IEEE Big Data*.
[21] Laurens van der Maaten. 2014. Accelerating t-SNE using Tree-Based Algorithms. *Journal of Machine Learning Research* 15, 93 (2014), 3221–3245.
[22] J. Wang, W. Zhang, and S. Yuan. 2017. *Display Advertising with Real-Time Bidding (RTB) and Behavioural Targeting*. Now Publishers Inc.
[23] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. In *AAAI*.
[24] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *KDD*.