tcc2vec: RFM-Informed Representation Learning on Call Graphs for Churn Prediction

Sandra Mitrović[a,*], Bart Baesens[a,b], Wilfried Lemahieu[a], Jochen De Weerdt[a]

[a]*Department of Decision Sciences and Information Management, KU Leuven, Leuven, Belgium*
[b]*School of Management, University of Southampton, Southampton, United Kingdom*

Abstract

Applying social network analytics for telco churn prediction has become indispensable for almost a decade. However, in the current literature, the uptake does not reflect in a significantly increased leverage of the available information that these networks convey. First, network featurization in general is a very cumbersome process due to the complex nature of networks and the lack of a respective methodology. This results in ad hoc approaches and hand-crafted features. Second, deriving certain structural features in very large graphs is computationally expensive and, as a consequence, often neglected. Third, call networks are mostly treated as static in spite of their inherently dynamic nature. In this study, we propose *tcc2vec*, a panoptic approach aiming at devising representation learning (to address the first problem) on enriched call networks that integrate interaction and structural information (to overcome the second problem), which are being sliced in different time periods in order to account for different temporal granularities (hence addressing the third problem). In an extensive experimental analysis, insights are provided regarding an optimal choice of interaction and temporal granularities, as well as representation learning parameters.

*Corresponding author
Email address: `sandra.mitrovic@kuleuven.be` (Sandra Mitrović)

Keywords: Representation Learning, Dynamic Networks, Enriched (Social) Networks, Churn Prediction in Telco

1. Introduction

Churn prediction (CP) using social network analytics, both in telco as well as in other industries, relies on expert-driven featurization of underlying (call) graphs. Apart from being cumbersome, this leads to a variety of ad-hoc solutions, as evidenced in the literature (Saravanan & Raajaa, 2012; Kusuma et al., 2013; Phadke et al., 2013; Kim et al., 2014a). Even more problematic is the fact that such approaches underexploit the actual information provided by the graph. For example, in the case of large call graphs, studies do take into account information about customer interactions, but most of the features related to graph structure, being computationally expensive, remain neglected (Zhu et al., 2011; Phadke et al., 2013). Furthermore, additional problems arise with the temporal nature of call graphs given that only few CP studies take it into account, thus techniques and methodologies are still largely missing. Representation learning (RL) allows to bypass the whole feature engineering process by constructing dense but high-quality representations (embeddings) of objects (texts, images, nodes) in a low dimensional space. Representational learning has been shown to be useful in domains ranging from natural language processing (NLP) (Bengio et al., 2003; Mikolov et al., 2013a), over image classification (Krizhevsky et al., 2012), to social network analytics (Grover & Leskovec, 2016). As for the latter, the uptake of graph RL techniques is mainly due to the fact that graphs are a natural and instrumental abstraction technique for all kinds of real-life relationships and entities. A recently proposed method for learning node representations in graphs, node2vec (Grover & Leskovec, 2016) is based on truncated random walks and the SkipGram model (Mikolov et al., 2013a), a renowned RL model from the NLP domain.

Node2vec, however, experiences two major drawbacks. First, it introduces two random walk-related hyperparameters which not only require additional tuning, but also negatively affect performance in case of very large graphs. Second, neither node2vec nor other approaches by which it was inspired, such as DeepWalk (Perozzi et al., 2014) and LINE (Tang et al., 2015), consider temporal graphs. This is inconvenient as many graphs modeling real-life scenarios, such as call graphs derived from Call Detail Records (CDRs), undergo constant changes throughout time. Additionally, truncated random walks, which are applied in a series of existing approaches (Perozzi et al., 2014; Grover & Leskovec, 2016; Cao et al., 2016; Dong et al., 2017; Zhang et al., 2017; Yang et al., 2015; Zhou et al., 2017; Yang et al., 2016) usually require a careful selection of input parameters such as the number and length of random walks. As such, the application of node2vec or related techniques specifically for CP, albeit looking very promising, is strongly hampered by non-existing benchmarks in the current literature with a majority of papers following the choices propounded by Perozzi et al. (2014) (e.g. Cao et al. (2016)) or by Grover & Leskovec (2016) (e.g. Nguyen et al. (2018)). A notable exception to this is the study by Dong et al. (2017) in which an analysis of parameters such as walk length and number of walks is presented. Nevertheless, this study is not considering dynamic graphs, is not looking into CP, and most importantly, only showcases applications to at least one order of magnitude smaller graphs.

Therefore, in this work, we propose *tcc2vec*, a panoptic approach aiming at devising the most appropriate RL on call graphs for CP. This work extends earlier research presented in (Mitrović et al., 2017b) and (Mitrović et al., 2017a) in which initial foundations for *ttc2vec* were proposed.

The key contributions of this work can be summarized as follows:

- We propose an integrated framework for CP modelling through RL on call graphs, relying on (1) a scalable node2vec-inspired RL algorithm that allows deriving embeddings for very large graphs, (2) a network augmentation method that enables to jointly take into account behavioral interaction and structural characteristics of customers, and (3) a windowing-based approach to include dynamic effects of call graphs into learned representations.

- We perform an in-depth experimental evaluation based on real-life datasets, leading to the specification of optimal parametrizations of *tcc2vec*. In particular, we devise a full factorial design to investigate the predictive performance effect achieved by combinations of different parameters (factors) which relate to interaction granularity, temporal granularity and characteristics of the RL method, such as the number and length of walks.
- We provide guidelines both for researchers as well as practitioners regarding the potential adoption of RL for CP in practice, and accentuate various research challenges that RL on graphs is raising in this area.

The remainder of the paper is organized as follows. In Section 2, we provide a short overview of literature concerning our topics of interest. In Sections 3 and 4, we explain our method and experimental setup, respectively. In Section 5 we present the results of our experiments, discuss them in Section 6 and we provide a conclusion and ideas for future research in Section 7.

2. Related Work

In this section we provide an overview of relevant related work on call-graph-based CP, graph RL and network dynamics.

2.1. Call Graph-based Customer Featurization for Churn Prediction

Given the specific focus on CP in telco, this subsection addresses existing approaches that make use of the call graph.

Call-Graph-based Behavioral Interaction Features using RFM. For telco CP, behavioral interaction information is predominantly quantified by means of the RFM (**R**ecency, **F**requency, **M**onetary) model (Hughes, 1994; Cheng & Chen, 2009), given its simplicity and good predictive performance (Keramati et al., 2014; Benoit & Van den Poel, 2012). Therefore, many different RFM operationalizations can be found in the literature, ranging from summary, coarse-grained to more detailed, fine-grained features. Summary RFM features usually only differ on the type of measure (R/F/M) and the dimension used (calls/seconds/SMSs), e.g. total call frequency/volume in (Dasgupta et al., 2008), seconds/frequency of use and SMS frequency in (Keramati et al., 2014). Detailed RFM features, on the contrary, show more variety as slicing along different dimensions (e.g. time (Owczarczuk, 2010; Motahari et al., 2014), direction (Dasgupta et al., 2008; Modani et al., 2013)), aggregation levels (e.g. incoming/outgoing for direction (Dasgupta et al., 2008; Modani et al., 2013; Motahari et al., 2014); peak/off-peak for time (Huang et al., 2012; Phadke et al., 2013)) is performed. In addition, applying different transformations (e.g. average (Zhang et al., 2012); ratio/percentage (Dasgupta et al., 2008; Kusuma et al., 2013; Phadke et al., 2013)) is quite common. Several studies also consider slicing according to the churner related information, which can again be considered as one of the dimensions (e.g. total interaction frequency with churners (Dasgupta et al., 2008; Kusuma et al., 2013), and total call volume to/from churner neighbors (seconds) (Dasgupta et al., 2008; Zhang et al., 2012)). We refer the interested reader to a more comprehensive tabular overview of different variations concerning detailed RFM in our previous study (Mitrović et al., 2017b).

Deriving Structural Network Features from Call Graphs. In the current literature, structural information as conveyed by the underlying call graph topology is usually characterized by centrality measures, most often using the simplest one, i.e. degree centrality. More advanced (and diversified) structural features for CP include 2^{nd} and 3^{rd}-order degree in Kusuma et al. (2013); 2^{nd}-order degree and a clustering coefficient in Zhang et al. (2012); PageRank in Huang et al. (2015); degree, closeness and eccentricity centrality, clustering coefficient, Shapley value, degree and proximity prestige in Saravanan & Raajaa (2012); PageRank, diameter, number of (strongly) connected components and cliques in Nanavati et al. (2006). However, due to computational expensiveness, especially for very large graphs, structural features are far from being fully exploited in the current literature (e.g. Zhu et al. (2011) opt to completely neglect such features, despite recognizing their value). As such, in this study, we aim at better leveraging structural information beyond hand-engineered and computationally demanding features, by relying on the learned node representations to incorporate the topological characteristics of nodes.

Combining Behavioral Interaction and Structural Call Graph Features. Extracting both behavioral interaction and structural information from call graphs and trying to leverage their predictive capacities for CP has already been investigated in several studies (Kusuma et al., 2013; Zhang et al., 2012; Huang et al., 2015; Modani et al., 2013; Motahari et al., 2014; Dasgupta et al., 2008). However, the issue of limited exploitation of structural information is a constant for these attempts as well. As such, the majority of studies exploit either only degree measures (Kusuma et al., 2013; Modani et al., 2013; Motahari et al., 2014; Dasgupta et al., 2008) or a slightly extended, but still limited number of structural features (Zhang et al., 2012; Huang et al., 2015). In Raeder et al. (2011), a compound feature combining structural and interaction information is used as the authors derive the number of calls that neighbors of one node make to the neighbors of another node. This is essentially frequency based on 2^{nd}-order neighborhood. As another example, Phadke et al. (2013) use degree centrality to normalize the values of the interaction features. Unquestionably, utilizing a set of diverse types of customer characteristics as input features for CP model building leads to better performance (Dasgupta et al., 2008; Kusuma et al., 2013; Zhang et al., 2012; Huang et al., 2015). Moreover, to the best of our knowledge, no conclusion can be drawn as to which of these two classes of features is more important for building more accurate models. However, the fact that a couple of studies show that RFM variables tend to be more important than structural features (e.g. Raeder et al. (2011) for edge classification in telco call graphs and Benoit & Van den Poel (2012) for CP in banking), is an impetus for our study to investigate the enrichment of graph topologies with RFM information.

2.2. Representation Learning on Graphs

Representation learning on graphs is a rapidly growing field of research. Stemming from embedding techniques first proposed in the natural language processing (NLP) domain (Bengio et al., 2003; Mikolov et al., 2013a), researchers have developed methods capable of learning distributed representations of nodes in graphs (Perozzi et al., 2014; Grover & Leskovec, 2016). The recent gain in attraction is mostly due to the fact that graphs are suitable for representing various types of complex real-life data. Moreover, RL techniques have been demonstrated to allow for deriving dense, high-quality but dataset- and task-agnostic representations enabling automation of feature engineering. As such, RL on graphs has been successfully applied for different unsupervised and supervised learning tasks. Examples include link prediction (Grover & Leskovec, 2016; Wang et al., 2016; Zhou et al., 2017), clustering (Cao et al., 2016; Zhang et al., 2017; Dong et al., 2017), visualization (Wang et al., 2016; Cao et al., 2016; Dong et al., 2017), and recommendation (Zhou et al., 2017). Moreover, an increasing popularity of classification applications can be observed, with noteworthy studies including (multi-)label classification in social networks (Facebook-based networks (Zhang et al., 2017), Flickr (Tang et al., 2015; Perozzi et al., 2014; Wang et al., 2016), Google (Zhang et al., 2017), YouTube (Tang et al., 2015; Perozzi et al., 2014; Wang et al., 2016), blogger networks derived from the BlogCatalog website (Grover & Leskovec, 2016; Perozzi et al., 2014; Wang et al., 2016)), in Wikipedia words co-occurrence networks (Grover & Leskovec, 2016; Tang et al., 2015), in biological networks (Protein-Protein Interaction) (Grover & Leskovec, 2016), in collaboration networks (AMiner) (Dong et al., 2017) and in citation networks (DBLP) (Tang et al., 2015).

Many previous works rely on random walks (Perozzi et al., 2014; Grover & Leskovec, 2016; Cao et al., 2016; Dong et al., 2017; Zhang et al., 2017; Yang et al., 2015; Zhou et al., 2017; Yang et al., 2016). Among these, the techniques proposed by Zhang et al. (2017) and Yang et al. (2015) are specifically tailored towards including additional node-related information (e.g. Yang et al. (2015) consider textual features). Even though our approach also looks into including additional information about nodes, the characteristics of our data are very different which makes these proposed techniques not applicable. The same holds for the technique by Dong et al. (2017) which, unlike us, examines biased random walks for heterogeneous graphs.

Deepwalk (Perozzi et al., 2014) and especially node2vec (Grover & Leskovec, 2016) are instrumental for our method. Both techniques rely on the SkipGram model (Mikolov et al., 2013b), however, node2vec was shown to perform better than DeepWalk (Perozzi et al., 2014), spectral clustering, and LINE (Tang et al., 2015), especially for classification purposes (Grover & Leskovec, 2016). Unlike our work, all these methods consider only static networks for multi-class classification. Very recently, Zhou et al. (2018) propose an approach for *dynamic* node embeddings, which, unlike random walk based approaches, uses triadic closures but underperforms DeepWalk in terms of recall on a telco dataset.

Other notable related works are GraphSage (Hamilton et al., 2017), which allows learning representations for unseen nodes, and the approach by Kipf & Welling (2017) based on convolutional neural networks. Besides being a static approach, GraphSage is not interesting for our work as it is particularly effective on graphs with rich node information, which is not our case. Furthermore, drawbacks of convolutional approaches lie both in the static aspect and scalability. The study of Cao et al. (2016), mentioned earlier, applies a stacked denoising autoencoder but likewise does not consider dynamic networks. Despite the success of deep architectures for RL, its application for CP in particular does not seem to be worth additional investigation given that methods either perform on-par (Umayaparvathi & Iyakutti, 2017) or only slightly outperform simple traditional methods (e.g. decision trees in Wangperawong et al. (2016)). The study by Castanedo et al. (2014) is a more successful application of deep learning for CP, however it does not take into account the dynamic aspect and utilizes undisclosed business-specific attributes.

2.3. Temporal Dynamics of Call Graphs

Several studies show that taking into account the temporal aspect for customer behavior analysis is indisputably more useful than a purely static approach (Hill et al., 2006; Eichinger et al., 2006; Chen et al., 2012). For capturing dynamics, most studies apply time-series techniques (Lee et al., 2011; Orsenigo & Vercellis, 2010; Chen et al., 2012), while sequence mining (Eichinger et al., 2006) and dynamic networks (Hill et al., 2006) are used as well. The latter are typically addressed in a two-phase approach whereby the network is first sliced in a series of snapshots treated as independent static graphs and only then, the extracted information is merged together (Michail, 2016; Casteigts et al., 2015; Nicosia et al., 2013). This type of approach is applied in e.g. Saravanan & Raajaa (2012) for mobile CP, Hill et al.

(2006) for fraud detection and Rahman & Al Hasan (2016) for link prediction, with certain variations. More precisely, Saravanan & Raajaa (2012) use a weekly windowing frame to generate features, Rahman & Al Hasan (2016) use dataset-defined snapshots and concatenate static features while Hill et al. (2006) use daily snapshots and apply exponential smoothing to penalize historical data. Despite being one of the few studies taking into account the dynamic aspect of CDR graphs, Hill et al. (2006); Saravanan & Raajaa (2012) still, unlike our work, perform ad-hoc featurizations of static graphs. Nevertheless, even though being few in number, there exist several other studies which reside at the intersection of dynamic networks and RL. In line with aforementioned approaches, Kim et al. (2014b); Hamilton et al. (2016); Hisano (2018) consider a sequence of static snapshots. In addition, similarly to our work, Kim et al. (2014b); Hamilton et al. (2016) then apply the SkipGram model to learn representations per snapshot. Besides considering different granularities for snapshots and being focused on the NLP domain, these studies differ from ours in reusing representations learned for the previous snapshot to initialize representations for the consecutive one. As opposed to these, Hisano (2018) instead uses stochastic gradient descent to optimize customized objective functions for link creation/dissolution for dynamic link prediction. Another very recent study makes a shift from previous approaches by using temporal random walks (Nguyen et al., 2018). Although deep architectures are out of scope of this work, it is worth mentioning that these have recently been exploited for learning dynamic embeddings as well (e.g. Goyal et al. (2017)) proposed a method based on deep autoencoders, Trivedi et al. (2017) proposed a deep recurrent architecture, and Trivedi et al. (2018) develops an attention based approach to model information propagation dynamics). Observe that mentioned works (operating on graphs) perform empirical evaluations with remarkably smaller graph sizes as compared to ours (e.g. in number of edges: <15K (Hisano, 2018), <80K (Goyal et al., 2017), <350K (Nguyen et al., 2018)).

In conclusion, to the best of our knowledge, our previous work (Mitrović et al., 2017a) is the only study in the current literature to exploit dynamic networks with node RL for CP in telco. Additionally, as evident in Table 1, our study is the only one in telco CP to incorporate all the aspects of call networks in an automatic way.

3. tcc2vec: RFM-informed Scalable Telco Customer Representation Learning in Call Graphs

This section details the main building blocks of our telco customer to vector (*tcc2vec*) RL technique, whose overall structure is given in Algorithm 1. The building blocks entail (1) an approach to enhance the original call graph topology with RFM information in order to enable the joint inclusion of both behavioral interaction and structural topology information for learning representations (lines 3-7 of Algorithm 1), (2) a scalable node2vec-inspired RL algorithm capable of dealing with networks containing millions of nodes and tens of millions of edges (line 8 of Algorithm 1), and (3) a method to include the dynamic effects of networks in the learned representation (lines 2 and 10 of Algorithm 1).

3.1. Enriching Call Graphs to Reinforce Comprehensive Capturing of Customer Behaviour

A first crucial component of *tcc2vec* includes the development of RFM-informed call graphs. The key idea is that by combining behavioral interaction information with the original graph topology, our scalable node2vec-inspired RL algorithm will be instigated to learn representations that jointly take into account these two different types of information. This is a paramount innovation in the context of telco CP, given that leveraging both types of information has posed several challenges, as explained in Section 2.1.

Table 1: Summary of recent literature about CP in Telco providing an overview related to including interaction information, including structural information, considering the dynamic aspect, using RFM to characterize interaction information, using social network analytics (SNA) and finally providing information whether the study avoids manual feature engineering. Studies which apply spreading activation functions or other variants of label propagation are considered to include structural information. For works which neither apply SNA nor consider call graphs, usage data is still treated as interaction data. Symbol ✓✗ denotes that the study is only partially compliant (e.g. in Saravanan & Raajaa (2012) only one of the features, number of calls, could be considered as RFM). Finally, symbol ✓^{DL} denotes that the study is using deep learning architectures for RL.

Authors	Interaction	Structural	Dynamic	RFM	SNA	Automated FE
Eichinger et al. (2006)	✓	✗	✓	✗	✗	✗
Nanavati et al. (2006)	✗	✓	✗	✗	✓	✗
Dasgupta et al. (2008)	✓	✓	✗	✓	✓	✗
Owczarczuk (2010)	✓	✗	✗	✓	✗	✗
Richter et al. (2010)	✓	✓	✗	✓	✓	✗
Raeder et al. (2011)	✓	✓	✓	✓	✓	✗
Chen et al. (2012)	✗	✗	✓	✗	✗	✗
Saravanan & Raajaa (2012)	✓✗	✓	✓	✓✗	✓	✗
Zhang et al. (2012)	✓	✓	✗	✓	✓	✗
Kusuma et al. (2013)	✓	✓	✗	✓	✓	✗
Modani et al. (2013)	✓	✓	✗	✓	✓	✗
Phadke et al. (2013)	✓	✓	✗	✓	✓	✗
Keramati et al. (2014)	✓	✗	✗	✓	✗	✗
Kim et al. (2014a)	✗	✓	✗	✗	✓	✗
Motahari et al. (2014)	✓	✓✗	✗	✓	✓	✗
Castanedo et al. (2014)	✓	✗	✗	✗	✗	✓^{DL}
Huang et al. (2015)	✓	✓	✗	✓	✓	✗
Wangperawong et al. (2016)	✓	✗	✓	✓	✗	✓^{DL}
Umayaparvathi & Iyakutti (2017)	✓	✗	✗	✓✗	✗	✓^{DL}
This study	✓	✓	✓	✓	✓	✓

Algorithm 1 Algorithm

```
Input: Call Detail Record (CDR), Interaction granularity $ig$, Temporal granularity $tg$,
    Walk length $wl$, Number of walks $nw$
 1: Transform CDR into $tg$ call graphs $G_1, ..., G_{tg}$
 2: for $each$ graph $G_i \in \{G_1, ..., G_{tg}\}$ do:
 3:     for $each$ node $n \in G_i$ do:
 4:         $RFM_{G_i}^{ig}(node) = Calc\_RFM\_Per\_Node$(node, $ig$);
 5:     end for;
 6:     $RFM_{G_i}^{ig} = \bigcup_{node \in G_i} RFM_{G_i}^{ig}(node)$;
 7:     $AG_i = Construct\_Artificial\_RFM\_Augmented\_Graph(G_i, ig, RFM_{G_i}^{ig})$;
 8:     $Repr_i = Learn\_Representations(AG_i, wl, nw)$;
 9: end for;
10: $Repr = ||_{i=1}^{tg} Repr_i$;
11: Train and evaluate using $Repr$ as input for predictive model;
Output: AUC and lift scores
```

Interaction Information. We first describe RFM model configurations (line 4 in Algorithm 1). Given that RFM information in telco mainly resides in the CDR data, we specify: 1) Recency (R) as the number of days between the end of the observed period and the customer's last call (within the same period); 2) Frequency (F) as the number of calls of a customer during the observed period; 3) Monetary (M) as the duration (in seconds) of customer calls during the observed period.[1]

Next, RFM variables are usually subject to a particular choice in terms of slicing or dicing according to other dimensions. We opt for two clear-cut yet powerful representations: Summary-RFM (denoted by RFM^s, i.e. ig=s in line 4 of Algorithm 1): total R/F/M per customer per observed period. Thus, the frequency per period p of customer c is calculated as: $F_{G_p}^{s}(c) = |e(c,n) : n \in \mathcal{N}_{G_p}(c) \wedge e(c,n) \in E_p|$, where $G_p = (V_p, E_p)$ is the undirected graph corresponding to period p and $\mathcal{N}_{G_p}(c)$ denotes the neighborhood of customer/node c in G_p. We proceed similarly for

[1]Duration is used as a proxy in order to avoid the necessity of billing information, which is not always available, but given that in most cases, duration and billed amount are directly related, we assume that this has no impact.

$R^s_{G_p}(c)$ and $M^s_{G_p}(c)$ substituting the cardinality operator with the maximal recency and the sum of all durations associated to corresponding edges, respectively.

Detailed-RFM (denoted by RFM^d, i.e. $ig{=}d$ in line 4 of Algorithm 1): each of the R/F/M variables is sliced based on the direction and destination dimension into three subcategories: outgoing towards home network, outgoing towards other networks and incoming, inspired by the approaches in Dasgupta et al. (2008); Modani et al. (2013); Motahari et al. (2014). Thus, the frequency per period p of customer c is now expressed by three variables:

1) $F^{d:out\_h}_{G_p}(c) = |e(c,n) \,:\, n \in \mathcal{N}_{G_p}(c) \wedge n \in \mathcal{H} \wedge e(c,n) \in E_p|$,
2) $F^{d:out\_o}_{G_p}(c) = |e(c,n) \,:\, n \in \mathcal{N}_{G_p}(c) \wedge n \in \mathcal{O} \wedge e(c,n) \in E_p|$,
3) $F^{d:in}_{G_p}(c) = |e(n,c) \,:\, c \in \mathcal{N}_{G_p}(n) \wedge e(c,n) \in E_p|$,

where $G_p = (V_p, E_p)$ represents the directed graph corresponding to period p, $\mathcal{H}$ denotes the home network, $\mathcal{O}$ denotes other network(s), and further notation is as before. R and M are treated correspondingly.

Integrating Interaction and Structural Information into RFM-Augmented Call Graphs. To conjoin interaction and structural information (Alg. 1, line 7, detailed in Alg. 2), we start with preserving the topology of the original call graph (Alg. 2, line 1). This is motivated by the RL part of *tcc2vec*, which is based on random walks and thus explicitly exploits the network topology.

Therefore, no sampling or any other manipulation which would distort structural information (or hinder the utility of interaction information) is permitted.

We proceed with the same idea and, to integrate RFM information, we consider the strategies which align best with random walk traversal of the underlying call graph. In our preliminary study (Mitrović et al., 2017b), we have demonstrated that the most straightforward option of adding RFM variables as edge weights underperforms the pure traditional RFM model (see RFM-embedded architecture in

Mitrović et al. (2017b)). However, another type of architecture, where RFM information is added in the form of additional artificial nodes (referred to as RFM-augmented graphs) showed promising preliminary results (Mitrović et al., 2017b). To devise the construction of these artificial nodes, we follow the idea frequently used in the literature related to customer segmentation and customer lifetime value modeling, where customers are usually segmented partitioning each of their R/F/M variables in five equi-frequency bins[2] (corresponding to very high, high, medium, low, very low) (Hughes, 1994; McCarty & Hastak, 2007; Cheng & Chen, 2009). Next, we assign an artificial node to each bin and connect original customer nodes with artificial nodes of their corresponding bins (lines 3-8 in Algorithm 2). Depending on the level of interaction granularity, we devise four different augmented call graph topologies, as follows:

- Augmented Graph (AG_s) with Summary interaction granularity ($ig{=}s$)
  Here the start from Summary-RFM (hence only 3 variables), and therefore, after partitioning in bins, newly obtained graph has 15 new nodes and $3 * |V|$ more edges than the original graph $G = (V, E)$.

- Augmented Graph (AG_d) with Detailed interaction granularity ($ig{=}d$)
  In this case, since we use Detailed-RFM which start with three subcategories for each of R/F/M that are further binned, up to 45 artificial nodes are used to enrich the original graph. The number of additional edges, compared to the original graph, increases with nine times the original number of nodes.

- Augmented Graph (AG_{s+ch}) with Summary+Churn interaction granularity ($ig{=}s + ch$)

[2]Exceptionally, due to the skewed distribution of R/F/M values, one can end up having less than five bins per R/F/M.

- Augmented Graph (AG_{d+ch}) with Detailed+Churn interaction granularity (ig=$d + ch$)

Graphs AG_{s+ch} and AG_{d+ch} are exactly the same as graphs AG_s and AG_d, respectively, except for adding one additional artificial node representing churn (see lines 11-14 in Algorithm 2).

Algorithm 2 *Construct_Artificial_RFM_Augmented_Graph*($G_p, ig, RFM^{ig}_{G_p}$)

```
Input: Graph $G_p$ per period $p$, Interaction granularity $ig$, RFM features of nodes in $G_p$:
    $RFM^{ig}_{G_p} = \{R^{ig}_{G_p}, F^{ig}_{G_p}, M^{ig}_{G_p}\} = \{R^{ig}_{G_p}(n), F^{ig}_{G_p}(n), M^{ig}_{G_p}(n) \mid \forall n \in G_p\}$
 1: Construct $AG_p$, $AG_p := G_p$;
 2: for $X^{ig}_{G_p} \in \{R^{ig}_{G_p}, F^{ig}_{G_p}, M^{ig}_{G_p}\}$ do:
 3:     Divide $X^{ig}_{G_p}$ into 5 bins based on $k * 20^{th}$-percentile, $k \in \{1, ..., 5\}$;
 4:     Assign an artificial node $\mathcal{X}^{ig;k}_{G_p}$ to each bin $b_k, k \in \{1, ..., 5\}$;
 5:     Add all artificial nodes $\mathcal{X}^{ig;k}_{G_p}, k \in \{1, ..., 5\}$ to graph $AG_p$;
 6:     for $each$ node $n \in G_p$ do:
 7:         Find artificial node $\mathcal{X}^{ig;k}_{G_p}$ s.t. $X^{ig}_{G_p}(n)$ belongs to its corresponding bin $b_k$;
 8:         Add edge $(n, \mathcal{X}^{ig;k}_{G_p})$ to graph $AG_p$;
 9:     end for;
10: end for;
11: if $ig \in \{s + ch, d + ch\}$ then:
12:     Add additional artificial node $\mathcal{CH}$ to graph $AG_p$;
13:     Add edge $(ch\_n, \mathcal{CH})$ to graph $AG_p$ for identified churners $ch\_n$;
14: end if;
Output: Graph $AG_p$
```

All constructed networks are undirected given that initial attempts with directed networks led to sub-par performance. This is a consequence of the sparsity of our call network which resulted in random walks getting stuck at sink nodes. Furthermore, some existing studies have a preference for undirected call networks even when their proposed methods utilize only handcrafted features (without considering random walks) (Kim et al., 2014a).

Additionally, we consider augmented graphs unweighted, for two different reasons. First, due to a very different nature of node types (existing=customer vs. artificial=RFM), it is not straightforward how to determine the corresponding weights without biasing (much) the process of walk generation in favor of one type or the other. Second, determining the weights among artificial edges themselves is intricate as the current RFM-related literature prioritizes among RFM variables in an ad-hoc and domain/dataset-dependent manner. To avoid this, we opt for following the same approach as RFM-related studies which consider R-F-M as equally important (McCarty & Hastak, 2007; Cheng & Chen, 2009).

3.2. A Scalable node2vec-inspired Representation Learning Algorithm for Call Graphs

The node2vec approach as proposed in Grover & Leskovec (2016) serves as the basis of our customer RL technique deployed to call graphs. Node2vec and earlier node RL techniques (Perozzi et al., 2014; Tang et al., 2015) rely on the similarity of node neighborhoods to word neighborhoods in NLP, allowing to apply the SkipGram neural network architecture for deriving node representations. Conceptually, the SkipGram model consists of two steps: first, extracting word contexts from a given corpus and second, based on extracted contexts maximizing the probability of collocating the words from the same context using stochastic gradient descent.

In a graph setting, random walks of fixed-length *wl* are typically used to define node neighborhood/context (Perozzi et al., 2014; Grover & Leskovec, 2016) and to avoid implicit bias induced by the selection of a start node, walks are restarted *nw* times from each node. We define our random walks as first-order Markov processes which only take into account the (immediate) previous step. That is, a random walk which reaching a particular node u looks only into its neighborhood $\mathcal{N}_u$ to determine the next node v. The transition probability is defined as:

$$Pr(u \rightarrow v) = Pr(v|u) = \frac{w_{uv}}{\sum\limits_{z \in \mathcal{N}_u} w_{uz}} \tag{1}$$

where w_{uv} denotes the weight of edge $e = (u, v)$.

To construct a uniform distribution from the above obtained discrete distribution and allow for a more efficient sampling when performing random walking, alias sampling (Kronmal & Peterson, 1979) can be applied. The only prerequisite is to generate an alias table per node in advance. Note that in the case of unweighted graph equation (1) further simplifies to uniform sampling from the set of adjacent nodes which does not even require any precomputation.

Once random walks are generated, the basic idea of the SkipGram model is used, aimed at bringing the representations of the nodes from the same context / neighborhood closer than those of the nodes found in different contexts / neighborhoods. In other words, if we denote a set of nodes in the graph by V, we are learning a function $f, f : V \to R^d$ such that: $\max_f \prod_{v \in V} \prod_{c \in C_v} Pr(f(c)|f(v))$, where C_v is a set of contexts of node v.

The conditional probability $Pr(f(c)|f(v))$ is defined using a softmax function: $Pr(f(c)|f(v)) = \frac{e^{f(c) \cdot f(v)}}{\sum_{c' \in C} e^{f(c') \cdot f(v)}}$, where C is a set of all available contexts. It yields the following objective function after switching to log likelihood:

$$\max_f \sum_{v \in V} \sum_{c \in C_v} \left(f(c) \cdot f(v) - log \sum_{c' \in C} e^{f(c') \cdot f(v)} \right)$$

However, as the second factor in the objective function (inherited from the normalization part in softmax) encompasses all possible contexts c', it is computationally very expensive and therefore, typically, approximated. To this end, we use negative sampling as suggested by Mikolov et al. (2013b) and reformulate the objective as:

$$\max_f \sum_{v \in V} \left(log\ \sigma(f(c) \cdot f(v)) + \sum_{k=1}^{K} E_{v_k \sim P_n(v)}[log\ \sigma(-f(v_k) \cdot f(v))] \right),$$

where σ is the sigmoid function, K is number of negative examples and $P_n(v)$ is noise distribution according to which negative sampling is performed.

Comparison with similar representation learning methods. Figure 1 graphically summarizes the similarities and differences between our method and its two most closely related methods: DeepWalk and node2vec.

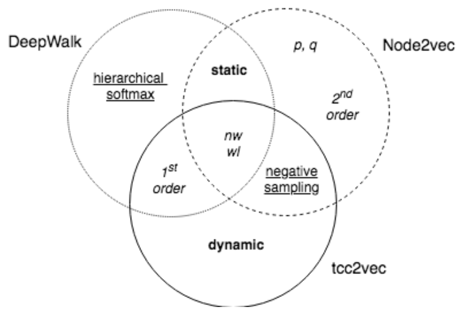


Figure 1: Graphical illustration of the comparison between our method (tcc2vec, represented by the solid circle) with its two closest existing methods (DeepWalk, dotted and Node2vec, dashed circle). We compare according to: 1) Random walk generation (in italics), i.e., parameters wl-walk length, nw-number of walks, p, q, and the order of Markov model required (1st vs. 2nd); 2) SkipGram implementation details (underlined), hence either negative sampling or hierarchical softmax and 3) type of setting/graphs considered - dynamic vs. static aspect (in bold).

As for computational complexity related to random walk generation, in case of unweighted graphs, both tcc2vec and DeepWalk require only $O(1)$ time. It is worth mentioning though that DeepWalk is primarily designed to handle unweighted graphs, hence it might not work optimally in the case of weighted graphs. In case of weighted graphs, as we require the construction of an alias table for each node, tcc2vec's space and time complexity is $O(|V|)$, while, due to considering 2^{nd}-order Markov model and introducing additional in-out and return parameters p and q, node2vec requires both node and edge alias tables which leads to $O(|P| \cdot |Q| \cdot (|V| + |E|))$ complexity, where $p \in P$ and $q \in Q$, hence the number of all possible combinations for the values of parameters p and q has to be taken into account. An additional question is how to define sets P and Q given that node2vec doesn't provide methodology to this end. Moreover, we suspect that at least the return parameter is obsolete as, considering undirected graphs, we give random walks a possibility to return, and hence no special treatment to enforce or impede backtracking is necessary in our case.

With respect to the optimization part, unlike DeepWalk which uses hierarchical softmax, a less efficient method for approximating the normalization factor in softmax probabilities, tcc2vec (and node2vec) use the more efficient negative sampling. In a nutshell, for each node, hierarchical softmax requires $O(log|V|)$ time to traverse the binary tree and calculate conditional probabilities, while negative sampling requires only $O(|K|/p_k)$, where K is a set of negative samples and p_k is the corresponding sampling distribution which given that, p_k is lower bounded and $|K|$ is constant (by default: $|K| = 5$) leads to $O(1)$ (we refer the interested reader to Mikolov et al. (2013b)).

It is worth mentioning that our edge weight-driven random walks used for context generation is similar to the incident-edges-weighted sampling procedure performed in LINE (Tang et al., 2015). However, in LINE, each context is generated from two parts, which are constructed independently from first and second-order neighborhoods and additionally, optimized separately using two different objective functions.

3.3. Including Dynamic Effects in Learned Representations

As a consequence of continuous changes in customer behaviour, call networks perpetually evolve, both in terms of (dis)appearance of nodes and links as well as their characteristics (e.g. the interaction may persist, but its intensity and frequency might change). Learning a single representation per customer would, therefore, hinder valuable fluctuations in customer behaviour which happen throughout time. Moreover, it can be expected that accurate CP is highly dependent on detecting changes in customer behavior and network topology. As such, in line with previous works considering featurization based on consecutive, non-overlapping time intervals, *tcc2vec* allows for the following time granularities (tg):

- Four weekly networks ($tg = 4$);
- Two bi-weekly networks ($tg = 2$);
- A single (monthly) network ($tg = 1$, corresponding to a static setting).

The key idea is that by including multiple representations of different consecutive time periods (see line 1 in Algorithm 1), the classification algorithm is provided with the opportunity to take into account changes in the representations over time. Once learnt, the representations corresponding to separate time periods are then stacked together using a concatenation operator (see line 10 in Algorithm 1). This setting has been adopted in previous studies as well, e.g. in Rahman & Al Hasan (2016). Illustration of complete method with $tg = 2$ and $ig = s$ is given in Figure 2.

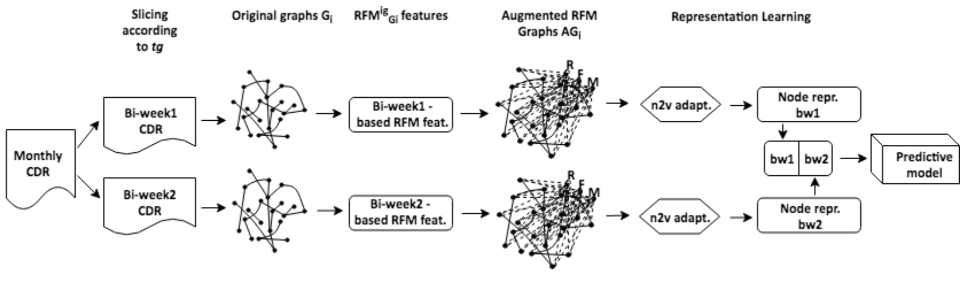


Figure 2: Graphical illustration of *tcc2vec* for $tg = 2$ and $ig = s$.

4. Experimental Evaluation

Given that a key contribution of this work consists of an investigation of crucial parameters for *tcc2vec*, this section details our experimental evaluation setup.

4.1. *Data and Tools*

Experiments were performed using two data sets (one for prepaid, one for postpaid). Both consist of four months of CDR data (only calls, no SMS or other usage types) providing only the following information: caller, callee, date/time and call duration. For confidentiality, all call numbers are encrypted. Churn labels are not available, although for postpaid, we are given ported-out dates of a several dozen of ported-out customers. More details about the datasets can be found in Table 2.

Table 2: Statistics of the datasets used, for monthly (M), bi-weekly (BW) and weekly (W) level. Notation: $|V|$ - number of nodes, $|E|$ - number of edges, *avg(deg)* - average degree, *avg(cl.coeff.)* - average clustering coefficient, $|CC|$ - number of connected components, $|max\_cq|$ - size of maximal clique, *dac* - degree assortativity coefficient.

Measure	Prepaid						
	M	BW 1	BW 2	W 1	W 2	W 3	W 4
$\lvert V\rvert$	4303541	3099439	3414621	2251195	2213089	2197238	2731485
$\lvert E\rvert$	5936423	3410857	3993270	2116619	2053684	2031695	2817743
avg(deg)	2.75886	2.20095	2.33892	1.88044	1.85594	1.84932	2.06316
avg(cl.coeff.)	0.05749	0.04231	0.04756	0.03121	0.03029	0.02958	0.03950
$\lvert CC\rvert$	138509	244535	207298	314793	322511	320098	266883
$\lvert max\_cq\rvert$	7	7	7	6	6	6	6
dac	-0.00110	-0.00124	-0.00117	-0.00145	-0.00144	-0.00141	-0.00138

Measure	Postpaid						
	M	BW 1	BW 2	W 1	W 2	W 3	W 4
$\lvert V\rvert$	4799149	3741692	4046353	2929347	2889360	2894179	3414656
$\lvert E\rvert$	9246134	5496039	6365786	3488464	3404493	3411408	4556203
avg(deg)	3.85324	2.93773	3.14643	2.38173	2.35657	2.35743	2.66862
avg(cl.coeff.)	0.06939	0.05707	0.06138	0.04450	0.04405	0.04295	0.05370
$\lvert CC\rvert$	27392	72906	55902	143390	149481	148274	97970
$\lvert max\_cq\rvert$	7	6	7	6	5	6	6
dac	-0.02290	-0.02052	-0.02123	-0.01951	-0.01937	-0.01954	-0.02073

4.2. *Definitions and Setting*

In the absence of churn labels, we define churn as customer inactivity which can be detected from CDRs, a method already used in previous works, e.g. Kusuma et al. (2013). However, previous studies tend to use the complete data from one month to predict which customers will churn immediately during the next month. These scenarios are not applicable in real business cases as they leave no time for devising appropriate retention campaigns. Therefore, in this work, we apply a differ-

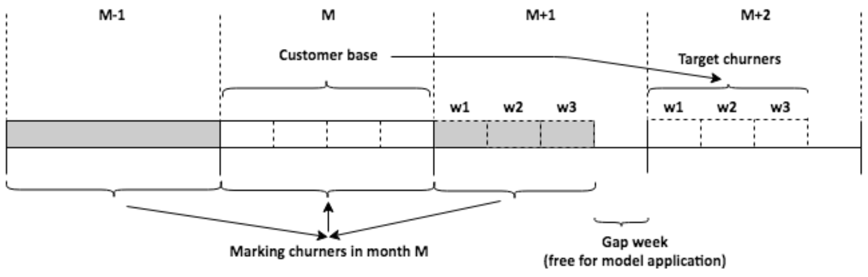


Figure 3: Graphical illustration of the four-month CDR information usage.

ent setting using four months of data, denoted as $M-1$ to $M+2$, where we consider customers from month M as our customer base and try to predict their probability of churning in month $M+2$. Since we consider time periods of different length, to label customers from month $M-1$ who are already churning during period p_i of month M, we devise following definition (for illustration for weekly scenario, see Figure 3):

- The customer appears in the CDRs during the month M, has not ported out during the month M before period p_i and had its last call in month M during period p_i,
- The customer does not appear in the CDRs during the first $n-1$ periods of month $M+1$, with n being number of periods or has been ported out during the same period.

Adopting this churn definition, we end up with around 12.2% of churners for the prepaid and 7.8% of churners for the postpaid dataset.

4.3. Experimental Design

In order to get insights in how to optimally configure *tcc2vec*, we set up a full factorial experimental design consisting of the following four factors:

- F1: Interaction granularity with four levels: summary (s), summary+churn (s+ch), detailed (d), detailed+churn (d+ch);
- F2: Temporal granularity with three levels: monthly (m), bi-weekly (b) and weekly (w);
- F3: Number of random walks per node with three levels: 10, 30 and 50;
- F4: Length of random walks with three levels: 10, 30 and 50.

This results in $4*3^3 = 108$ different combinations per dataset. The choice of values for factors F3 and F4 stems from both insights from several previous works (e.g. walk length 40 is used in Perozzi et al. (2014); Zhang et al. (2017); Cheng Yang (2017); 10 walks are used in Grover & Leskovec (2016); Cheng Yang (2017); 40 walks in Zhang et al. (2017)), as well as the need to maintain a computationally feasible setup.

Moreover, given that the number of iterations, as an additional parameter, has its own computational impact, we perform a supplementary analysis for the best performing methods of the previous experiment (per dataset) in which we vary the number of iterations with values of 1, 5, 10, 15 and 20. All computations were performed on a high performance computing platform of VSC[3] in order to fully parallelize random walk generations and representation derivations.

Our predictive models are generated using logistic regression with $l2$-regularization. Besides being a well-known classification technique for its interpretability and on-par performance with more complex techniques Owczarczuk (2010); Verbeke et al. (2010), logistic regression was also applied by previous studies performing classification based on learned node representations Grover & Leskovec

[3]`https://www.vscentrum.be/`

(2016); Perozzi et al. (2014). We use 10-fold cross validation to tune the regularization hyper-parameter. The AUC and lift scores (at 0.5%) are used for evaluation which is performed in an out-of-sample fashion.

5. Experimental Results

Comparison with baseline methods. First, we compare our method tcc2vec to DeepWalk and node2vec methods. Obtained predictive performance, in terms of AUC and lift (at 0.5%) as well as computational time required for each method is displayed in the Table 3. Obviously, tcc2vec outperforms both baselines on both datasets by huge margin in terms of AUC and lift. We provide more detailed segregation of computational time which clearly confirms the theoretical complexity evaluation provided in the last paragraph of the Section 3.2: node2vec requires a lot of time for calculating alias tables (preparatory phase) and generating random walks, while DeepWalk requires a lot of time for learning node representations. Given that tcc2vec operates on RFM-Augmented graphs, obtaining higher running time (compared with competing methods) for certain phases seems reasonable.

Interaction and temporal granularity (F1 & F2). Secondly, we focus our analysis on the first two factors. Results of a 10-fold cross validation experiment are reported in Tables 4 and 5 for the postpaid and prepaid datasets respectively. Observe that the reported values are averages over the results for all combinations of the two other factors, i.e. number of walks per node and walk length.

Regarding the impact of the interaction granularity, we can see that this factor indeed influences the classification result. In terms of AUC, both for prepaid and postpaid, we can conclude that the summary+churn interaction level provides the highest score. In terms of lift, the best results are obtained for summary+churn interaction level for prepaid and detailed interaction level for postpaid. It is also worth

Table 3: Comparison in terms of AUC (lift at 0.5%) and run time between three methods: DeepWalk, node2vec and tcc2vec for the prepaid and postpaid datasets (in case of DeepWalk, the preparatory phase refers to neighborhood extraction per node). The results are averaged across 10 folds (different from folds used for hyperparameter tuning). For all methods, monthly temporal granularity with walk length 40, number of walks 10, 128 dimensions and window size 10 is considered. For each method, its corresponding choice of the underlying undirected graph is applied: DeepWalk with its original unweighted graph, node2vec with its original weighted graph, and tcc2vec with its RFM-Augmented graph with the "s+ch" interaction granularity. For node2vec, the best result for different (p,q) parameters is provided and computational time shown refers to the average run time required for one (p,q) combination assuming that different combinations can run in parallel. The best score per dataset is marked in bold.

Dataset	Method	Predictive perf.	Computational time (in seconds)			
		AUC (Lift)	Preparatory phase	Random walk generation	Learning representations	Total time
Prepaid	DeepWalk	0.60886 (1.77355)	**184.79**	1346.20	62961.02	64492.01
	node2vec	0.62660 (1.40002)	2240.50	10150.26	17282.40	29673.16
	tcc2vec	**0.68215** (1.98502)	266.96	919.03	20975.06	22161.05
Postpaid	DeepWalk	0.66589 (2.83757)	**246.81**	1380.01	72482.02	74108.84
	node2vec	0.63975 (1.20000)	7895.24	17663.77	21706.28	47265.29
	tcc2vec	**0.76099** (3.54606)	343.24	938.04	25683.05	26964.33

noticing that in terms of AUC, both for prepaid and postpaid, summary+churn and detailed+churn interaction levels always outperform summary and detailed interaction levels, respectively. Hence, enriching the original topology with churn information adds value in terms of predictive performance. The same observation holds in terms of lift for prepaid dataset.

With respect to the temporal granularity, both in terms of AUC and lift and for both datasets, a unanimous conclusion can be derived: the weekly level always outperforms bi-weekly one, which in turn performs better than monthly.

Number of walks and walk length (F3 & F4). Thirdly, two key parameters of the RL part of *tcc2vec* are investigated, i.e. the number of walks and walk length. Results are depicted in Figures 4 and 5 for the postpaid and the prepaid datasets respectively. The graphs show average AUC scores for every combination of walk length (10, 30, 50) and number of walks (10, 30, 50) for different interaction granularity levels (that is, different RFM-augmented graphs). From these, we can conclude

Table 4: Comparison in terms of AUC and lift (at 0.5%) between different methods for the **post-paid** dataset. The results are averaged across 10 folds (different from folds used for hyperparameter tuning). Horizontally, a comparison between different levels of temporal granularity for the same interaction granularity is provided. Vertically, a comparison between the different interaction granularities (summary/summary+churn/detailed/detailed+churn) for the same temporal granularity level can be performed. Displayed are average AUC ± standard deviation and average lift ± standard deviation across different levels of other two factors (F3 and F4). The best score per row is marked in bold. The best overall score is marked in bold and underlined.

Temporal	Summary	Summary+Churn	Detailed	Detailed+Churn
	AUC (Lift)	AUC (Lift)	AUC (Lift)	AUC (Lift)
Monthly	0.76308 ± 0.00306 (3.61292 ± 0.02502)	**0.76341 ± 0.00324** (3.61359 ± 0.03348)	0.75705 ± 0.00813 (3.74623 ± 0.08124)	0.75726 ± 0.00831 (3.72948 ± 0.08526)
Bi-weekly	0.76501 ± 0.00234 (3.84050 ± 0.01890)	**0.76524 ± 0.00213** (3.83400 ± 0.01624)	0.76103 ± 0.00460 (3.85800 ± 0.01962)	0.76121 ± 0.00476 (3.84970 ± 0.02237)
Weekly	0.76819 ± 0.00240 (3.90514 ± 0.00959)	**0.76841 ± 0.00232** (3.90094 ± 0.00703)	0.76521 ± 0.00233 (3.91296 ± 0.01392)	0.76531 ± 0.00241 (3.90217 ± 0.00241)

Table 5: Comparison in terms of AUC and lift (at 0.5%) between different methods for the **pre-paid** dataset. The results are averaged across 10 folds (different from folds used for hyperparameter tuning). Horizontally, a comparison between different levels of temporal granularity for the same interaction granularity is provided. Vertically, a comparison between the different interaction granularities (summary/summary+churn/detailed/detailed+churn) for the same temporal granularity level can be performed. Displayed are average AUC ± standard deviation and average lift ± standard deviation across different levels of other two factors (F3 and F4). The best score per row is marked in bold. The best overall score is marked in bold and underlined.

Temporal	Summary	Summary+Churn	Detailed	Detailed+Churn
	AUC (Lift)	AUC (Lift)	AUC (Lift)	AUC (Lift)
Monthly	0.68214 ± 0.00276 (1.91193 ± 0.01868)	**0.68508 ± 0.00347** (2.00588 ± 0.02560)	0.68001 ± 0.00513 (1.89984 ± 0.01938)	0.68209 ± 0.00532 (1.96932 ± 0.03268)
Bi-weekly	0.68707 ± 0.00102 (1.94022 ± 0.00860)	**0.69615 ± 0.00263** (2.28452 ± 0.00816)	0.68782 ± 0.00210 (1.94715 ± 0.01084)	0.69521 ± 0.00387 (2.25625 ± 0.03734)
Weekly	0.69344 ± 0.00037 (1.96665 ± 0.00889)	**0.70313 ± 0.00207** (2.30911 ± 0.01023)	0.69398 ± 0.00104 (1.97686 ± 0.00893)	0.70190 ± 0.00253 (2.30080 ± 0.02099)

that with increasing number of walks and walk length, the AUC score increases as well, however, the relative increase becomes smaller as both parameters increase. Moreover, observe that choice of length and number of walks cannot be made interchangeably: X walks of length Y do not have the same predictive power as Y walks of length X. In fact, it can be observed that a higher number of shorter walks provides a better predictive performance than a lower number of longer walks, despite having similar computational complexities.

More detailed results corresponding to the lowest level of each factor for both

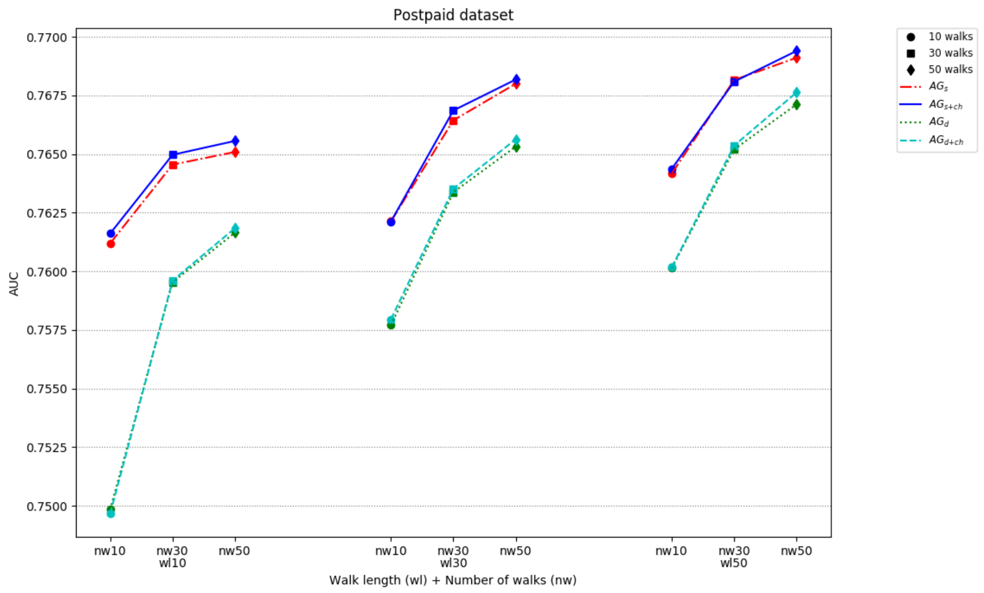


Figure 4: Average AUC scores for different values of walk length and number of walks for **postpaid** dataset. Different RFM-Augmented graphs corresponding to different levels of interaction granularity (F1) are represented separately while the average is calculated over different levels of temporal granularity (F2).

datasets can be found in the Appendix A.

Additionally, we consider all 108 different combinations per dataset as different methods and perform a Bayesian hierarchical correlated t-test Corani et al. (2016); Benavoli et al. (2016). This recently proposed test allows for comparing the performances of two methods not only for statistical significance, but also with respect to practical equivalence, which perfectly fits the purpose of this work, hence, the motivation for using it. To apply this test, we run 10 times 10-fold cross validation for each of 108 methods (using the same stratified folds) on both datasets and set the region of practical equivalence (ROPE) to 1% (as suggested in Corani et al. (2016)).

The test estimates the mean difference in AUC scores between any two meth-

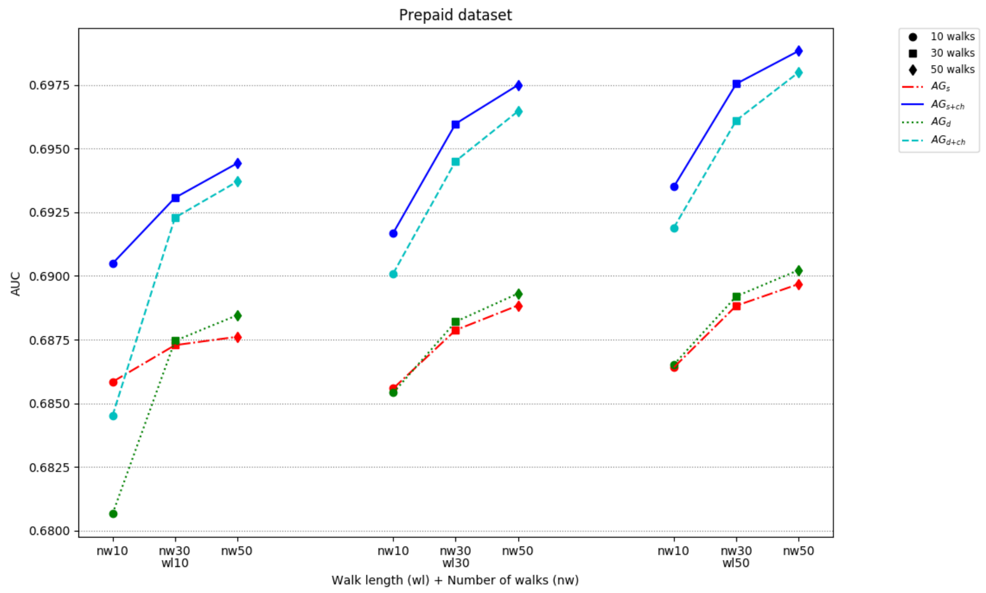
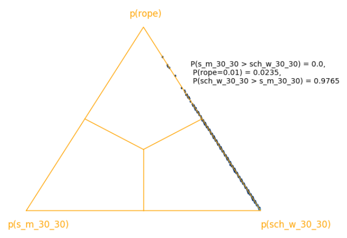
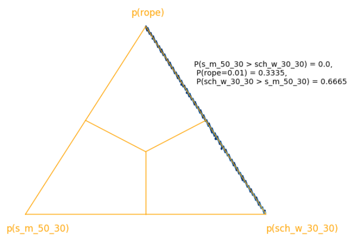


Figure 5: Average AUC scores for different values of walk length and number of walks for **prepaid** dataset. Different RFM-Augmented graphs corresponding to different levels of interaction granularity (F1) are represented separately while the average is calculated over different levels of temporal granularity (F2).

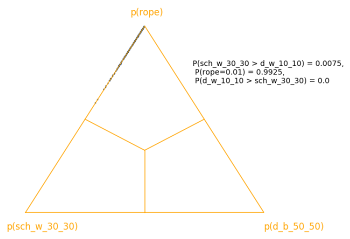
ods M_1 and M_2 and returns as a result three posterior probabilities: $P(M_1{>}M_2)$, probability that method M_1 performs practically better than M_2, $P(rope)$, probability that two methods are practically equivalent within ROPE and $P(M_2{>}M_1)$, probability that M_2 performs practically better than M_1. Results are significant if any of the posterior probabilities is ≥ 0.95 (as suggested by Corani et al. (2016)). A probability simplex (with corresponding posteriors) illustrating the test results taking *sch_w_30_30* as an example can be seen in Figure 6. Although there is no single winning method, test reveals many practically different methods. Complete results are provided in Figure A.1 in the Appendix A.



(a) Statistical significance.



(b) Positive evidence.



(c) Practical equivalence.

Figure 6: A probability simplex illustration of different possible outcomes of the Bayesian hierarchical correlated t-test on the example of *sch_w_30_30* for ROPE=1%. For readability, $s{+}ch$ is shortened to *sch*. In (a) *P(sch_w_30_30 > s_m_30_30) > 0.95* hence *sch_w_30_30* is significantly better than *s_m_30_30*. In (b) *P(sch_w_30_30 > s_m_50_30)* dominates but is <0.95 hence there is only positive evidence in favor of *sch_w_30_30* against *s_m_50_30*. In (c) $P(rope)$ dominates (and is >0.95) hence *sch_w_30_30* and *sch_w_30_30* are practically equivalent.

Number of iterations. Finally, as mentioned before, we perform a supplementary analysis of the number of iterations parameter of the SkipGram model. To this end, we only considered the best performing scenario in terms of interaction and temporal granularity (i.e. the RFM-augmented graph AG_{s+ch} and weekly temporal granularity level for both datasets with 50 walks of length 50) and subsequently performed a sensitivity analysis by instantiating the number of iterations parameter with values 1, 5, 10, 15, and 20. The results are provided in Figure 7 for both datasets. We can observe that after five iterations the performance remains stable irrespective of the chosen number of iterations in the SkipGram model.

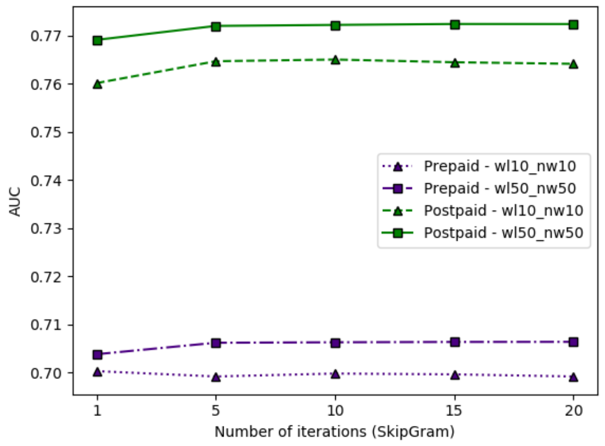


Figure 7: AUC scores for the SkipGram parameter *number of iterations* $i \in \{1, 5, 10, 15, 20\}$ using the best interaction granularity, i.e. AG_{s+ch} and optimal temporal granularity, i.e. the weekly level as an example. AUC scores are stable for varying $i \geq 5$.

6. Discussion

We would like to emphasize that preliminary results from Mitrović et al. (2017a) demonstrated that classical RFM (RFM) features perform sub-par compared to RFM-augmented networks with adapted node2vec (AG), even when the latter were calculated using fewer and shorter walks (10 walks of length 30) than what has been considered in this study. Given the evidence that the performance increases with increased number and length of walks, it is obvious that the achieved performance tremendously outperforms the one obtained using traditional RFM features, hence the reason for omitting them.

Furthermore, in this study we demonstrate that a higher number of shorter walks provides better predictive performance than a lower number of longer walks. We would like to draw attention that this insight goes in line with the homophily assumption in social networks, which implies that similar nodes tend to link together while similarity decreases with increases in distance. Moreover, this both overcomes graph sparsity and eliminates the need for force-steering random walks as done in node2vec (using parameters p and q). Hence, we can claim that our adaption of node2vec is justified not only in terms of computational efficiency, but predictive performance-wise as well.

7. Conclusion

In this work, we proposed *tcc2vec*, a panoptic approach aiming at devising the most appropriate RL on call graphs for CP. This work extends earlier research presented in Mitrović et al. (2017b) and Mitrović et al. (2017a) in which initial foundations for *tcc2vec* were proposed.

This paper contributes to the literature in three different ways. First, we developed an integrated RL-based CP framework, based on a scalable node2vec-inspired embedding algorithm for large graphs, which can learn node representations based on augmented CDR-graphs. These augmented graphs allow us to conjoin both structural and behavioral characteristics of customers. Furthermore, our approach allows to include dynamic effects into the learned representations. Second, based on a thorough experimental evaluation, we give insight into the optimal parametrization of *tcc2vec* in terms of interaction granularity, temporal granularity and characteristics of the RL method, such as the number and length of walks. Finally, this study provides guidelines both for researchers as well as practitioners regarding the potential adoption of RL for CP in practice. Furthermore, it gives insight into the various research challenges that RL on graphs is raising in this area.

For our future work, we would like to explore more sophisticated ways of capturing call dynamics (e.g. the ordering of calls, their inter-event time distribution). Additionally, it would be interesting to verify whether different ways of biasing random walks influence the predictive performance. Finally, different ways of creating artificial nodes and their effect on predictive performance could be explored as well.

References

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2016). Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *ArXiv e-prints*, *abs/1606.04316*.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*, 1137–1155.

Benoit, D. F., & Van den Poel, D. (2012). Improving customer retention in financial services using kinship network information. *Expert Systems with Applications*, *39*, 11435–11442.

Cao, S., Lu, W., & Xu, Q. (2016). Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* AAAI'16 (pp. 1145–1152). AAAI Press.

Castanedo, F., Valverde, G., Zaratiegui, J., & Vazquez, A. (2014). Using deep learning to predict customer churn in a mobile telecommunication network.

Casteigts, A., Flocchini, P., Mans, B., & Santoro, N. (2015). Shortest, fastest, and foremost broadcast in dynamic networks. *Int. J. Found. Comput. Sci.*, *26*, 499–522.

Chen, Z.-Y., Fan, Z.-P., & Sun, M. (2012). A hierarchical multiple kernel support vector machine for customer churn prediction using longitudinal behavioral data. *European Journal of operational research*, *223*, 461–472.

Cheng, C.-H., & Chen, Y.-S. (2009). Classifying the segmentation of customer value via rfm model and rs theory. *Expert systems with applications*, *36*, 4176–4184.

Cheng Yang, Z. L. C. T., Maosong Sun (2017). Fast network embedding enhancement via high order proximity approximation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17* (pp. 3894–3900).

Corani, G., Benavoli, A., Demšar, J., Mangili, F., & Zaffalon, M. (2016). Statistical comparison of classifiers through bayesian hierarchical modelling. In *Technical report IDSIA*.

Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A. A., & Joshi, A. (2008). Social ties and their relevance to churn in mobile telecom networks. In *Proceedings of the 11th conference on Extending database technology: Advances in database technology* (pp. 668–677). ACM.

Dong, Y., Chawla, N. V., & Swami, A. (2017). Metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*.

Eichinger, F., Nauck, D. D., & Klawonn, F. (2006). Sequence mining for customer behaviour predictions in telecommunications. In *Proceedings of the Workshop on Practical Data Mining at ECML/PKDD* (pp. 3–10).

Goyal, P., Kamra, N., He, X., & Liu, Y. (2017). Dyngem: Deep embedding method for dynamic graphs. In *IJCAI International Workshop on Representation Learning for Graphs*.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 855–864). ACM.

Hamilton, W., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*.

Hamilton, W. L., Leskovec, J., & Jurafsky, D. (2016). Diachronic word embeddings reveal statistical laws of semantic change. *CoRR*, *abs/1605.09096*.

Hill, S., Agarwal, D. K., Bell, R., & Volinsky, C. (2006). Building an effective representation for dynamic networks. *Journal of Computational and Graphical Statistics*, *15*, 584–608.

Hisano, R. (2018). Semi-supervised graph embedding approach to dynamic link prediction. In S. Cornelius, K. Coronges, B. Gonçalves, R. Sinatra, & A. Vespignani (Eds.), *Complex Networks IX* (pp. 109–121). Cham: Springer International Publishing.

Huang, B., Kechadi, M. T., & Buckley, B. (2012). Customer churn prediction in telecommunications. *Expert Systems with Applications*, *39*, 1414–1425.

Huang, Y., Zhu, F., Yuan, M., Deng, K., Li, Y., Ni, B., Dai, W., Yang, Q., & Zeng, J. (2015). Telco churn prediction with big data. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (pp. 607–618). ACM.

Hughes, A. (1994). *Strategic Database Marketing: The Master Plan for Starting and Managing a Profitable, Customer-Based Marketing program*. Probus Publishing Co., Chicago, IL.

Keramati, A., Jafari-Marandi, R., Aliannejadi, M., Ahmadian, I., Mozaffari, M., & Abbasi, U. (2014). Improved churn prediction in telecommunication industry using data mining techniques. *Applied Soft Computing*, *24*, 994–1012.

Kim, K., Jun, C.-H., & Lee, J. (2014a). Improved churn prediction in telecommunication industry by analyzing a large network. *Expert Systems with Applications*, *41*, 6575–6584.

Kim, Y., Chiu, Y.-I., Hanaki, K., Hegde, D., & Petrov, S. (2014b). Temporal analysis of language through neural language models. In *LTCSS@ACL*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Curran Associates, Inc.

Kronmal, R. A., & Peterson, A. V. (1979). On the alias method for generating random variables from a discrete distribution. *The American Statistician*, *33*, 214–218.

Kusuma, P. D., Radosavljevik, D., Takes, F. W., & van der Putten, P. (2013). Combining customer attribute and social network mining for prepaid mobile churn prediction. In *Proc. the 23rd Annual Belgian Dutch Conference on Machine Learning (BENELEARN)* (pp. 50–58).

Lee, H., Lee, Y., Cho, H., Im, K., & Kim, Y. S. (2011). Mining churning behaviors and developing retention strategies based on a partial least squares (pls) model. *Decision Support Systems*, *52*, 207–216.

McCarty, J. A., & Hastak, M. (2007). Segmentation approaches in data-mining: A comparison of rfm, chaid, and logistic regression. *Journal of business research*, *60*, 656–662.

Michail, O. (2016). An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, *12*, 239–280.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Mitrović, S., Baesens, B., Lemahieu, W., & De Weerdt, J. (2017a). Churn prediction using dynamic rfm-augmented node2vec. In R. Guidotti, A. Monreale, D. Pedreschi, & S. Abiteboul (Eds.), *Personal Analytics and Privacy. An Individual and Collective Perspective* (pp. 122–138). Cham: Springer International Publishing.

Mitrović, S., Singh, G., Baesens, B., Lemahieu, W., & de Weerdt, J. (2017b). Scalable rfm-enriched representation learning for churn prediction. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (pp. 79–88).

Modani, N., Dey, K., Gupta, R., & Godbole, S. (2013). Cdr analysis based telco churn prediction and customer behavior insights: A case study. In *International Conference on Web Information Systems Engineering* (pp. 256–269). Springer.

Motahari, S., Jung, T., Zang, H., Janakiraman, K., Li, X.-Y., & Hoo, K. S. (2014). Predicting the influencers on wireless subscriber churn. In *Wireless Communications and Networking Conference (WCNC), 2014 IEEE* (pp. 3402–3407). IEEE.

Nanavati, A. A., Gurumurthy, S., Das, G., Chakraborty, D., Dasgupta, K., Mukherjea, S., & Joshi, A. (2006). On the structural properties of massive telecom call graphs: findings and implications. In *Proceedings of the 15th ACM international conference on Information and knowledge management* (pp. 435–444). ACM.

Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., & Kim, S. (2018). Continuous-time dynamic network embeddings. In *3rd International Workshop on Learning Representations for Big Networks (WWW BigNet)*.

Nicosia, V., Tang, J., Mascolo, C., Musolesi, M., Russo, G., & Latora, V. (2013). Graph metrics for temporal networks. In *Temporal networks* (pp. 15–40). Springer.

Orsenigo, C., & Vercellis, C. (2010). Combining discrete svm and fixed cardinality warping distances for multivariate time series classification. *Pattern Recognition*, *43*, 3787–3794.

Owczarczuk, M. (2010). Churn models for prepaid customers in the cellular telecommunication industry using large data marts. *Expert Systems with Applications*, *37*, 4710–4712.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 701–710). ACM.

Phadke, C., Uzunalioglu, H., Mendiratta, V. B., Kushnir, D., & Doran, D. (2013). Prediction of subscriber churn using social network analysis. *Bell Labs Technical Journal*, *17*, 63–75.

Raeder, T., Lizardo, O., Hachen, D., & Chawla, N. V. (2011). Predictors of short-term decay of cell phone contacts in a large scale communication network. *Social Networks*, *33*, 245–257.

Rahman, M., & Al Hasan, M. (2016). Link prediction in dynamic networks using graphlet. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 394–409). Springer.

Richter, Y., Yom-Tov, E., & Slonim, N. (2010). Predicting customer churn in mobile networks through analysis of social groups. In *SDM* (pp. 732–741). SIAM volume 2010.

Saravanan, M., & Raajaa, G. V. (2012). A graph-based churn prediction model for mobile telecom networks. In *International Conference on Advanced Data Mining and Applications* (pp. 367–382). Springer.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 1067–1077). ACM.

Trivedi, R., Dai, H., & Wang, Y. (2017). Know-evolve: Deep temporal reasoning for dynamic knowledge graphs (supplementary material). In *ICML*.

Trivedi, R., Farajtabar, M., Biswal, P., & Zha, H. (2018). Representation learning over dynamic graphs. *CoRR*, *abs/1803.04051*.

Umayaparvathi, V., & Iyakutti, K. (2017). Automated feature selection and churn prediction using deep learning models. *International Research Journal of Engineering and Technology (IRJET)*, *4*.

Verbeke, W., Dejaeger, K., Martens, D., & Baesens, B. (2010). Customer churn prediction: does technique matter? In *Proceedings of the Joint Statistical Meeting, JSM2010, Vancouver, Canada*.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining KDD '16 (pp. 1225–1234). New York, NY, USA: ACM.

Wangperawong, A., Brun, C., Laudy, O., & Pavasuthipaisit, R. (2016). Churn analysis using deep convolutional neural networks and autoencoders. *CoRR*, *abs/1604.05377*.

Yang, C., Liu, Z., Zhao, D., Sun, M., & Chang, E. Y. (2015). Network representation learning with rich text information. In *Proceedings of the 24th International Conference on Artificial Intelligence* IJCAI'15 (pp. 2111–2117). AAAI Press.

Yang, Z., Cohen, W., & Salakhutdinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. *CoRR*, *abs/1603.08861*.

Zhang, D., Yin, J., Zhu, X., & Zhang, C. (2017). User profile preserving social network embedding. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17* (pp. 3378–3384).

Zhang, X., Zhu, J., Xu, S., & Wan, Y. (2012). Predicting customer churn through interpersonal influence. *Knowledge-Based Systems*, *28*, 97–104.

Zhou, C., Liu, Y., Liu, X., Liu, Z., & Gao, J. (2017). Scalable graph embedding for asymmetric proximity. In *AAAI Conference on Artificial Intelligence*.

Zhou, L., Yang, Y., Ren, X. T., Wu, F., & Zhuang, Y. (2018). Dynamic network embedding by modeling triadic closure process. In *AAAI Conference on Artificial Intelligence*.

Zhu, T., Wang, B., Wu, B., & Zhu, C. (2011). Role defining using behavior-based clustering in telecommunication network. *Expert Systems with Applications*, *38*, 3902–3908.

Appendix A.

Table A.1: Comparison in terms of AUC (lift at 0.5%) among different methods for the **prepaid** dataset. The results are sliced per different interaction granularity (summary,summary+churn,detailed,detailed+churn), temporal granularity (monthly,bi-weekly,weekly), walk length (10,30,50) and number of walks (10,30,50). The best score per interaction type is marked in bold. The best overall score is marked in bold and underlined.

Interaction	Temporal	Walk Length (wl) & Number of Walks (nw)								
		wl10_nw10	wl10_nw30	wl10_nw50	wl30_nw10	wl30_nw30	wl30_nw50	wl50_nw10	wl50_nw30	wl50_nw50
Summary	m	0.67801 (1.88151)	0.68099 (1.90620)	0.68210 (1.94871)	0.67847 (1.89584)	0.68311 (1.89696)	0.68495 (1.92387)	0.68047 (1.91161)	0.68475 (1.91750)	0.68642 (1.92514)
	b	0.68639 (1.93788)	0.68708 (1.94170)	0.68697 (1.94632)	0.68542 (1.92897)	0.68726 (1.95174)	0.68800 (1.93629)	0.68578 (1.95444)	0.68806 (1.92928)	0.68868 (1.93533)
	w	0.69311 (1.96272)	0.69378 (1.95062)	0.69376 (1.95715)	0.69286 (1.98056)	0.69324 (1.96877)	0.69356 (1.96304)	0.69297 (1.96798)	0.69368 (1.97705)	**0.69395** (1.97196)
Summary+Churn	m	0.68048 (1.96702)	0.68297 (2.00460)	0.68453 (2.00381)	0.68011 (1.98597)	0.68667 (2.01686)	0.68872 (2.03995)	0.68342 (1.97053)	0.68865 (2.03852)	0.69020 (2.02562)
	b	0.69176 (2.26750)	0.69442 (2.29059)	0.69579 (2.27880)	0.69342 (2.28947)	0.69726 (2.28979)	0.69895 (2.29234)	0.69492 (2.28517)	0.69892 (2.29138)	0.69991 (2.27562)
	w	0.69923 (2.31431)	0.70183 (2.31033)	0.70297 (2.31877)	0.70154 (2.29473)	0.70402 (2.29313)	0.70484 (2.31113)	0.70219 (2.29919)	0.70509 (2.31798)	<u>**0.70623**</u> (2.29393)
Detailed	m	0.66699 (1.84839)	0.68020 (1.90763)	0.68171 (1.90858)	0.67775 (1.90190)	0.68178 (1.90014)	0.68393 (1.90094)	0.67893 (1.90253)	0.68356 (1.92339)	0.68522 (1.90508)
	b	0.68290 (1.92084)	0.68781 (1.94839)	0.68901 (1.95221)	0.68603 (1.95715)	0.68864 (1.96113)	0.68935 (1.94377)	0.68723 (1.94250)	0.68933 (1.95014)	0.69012 (1.94823)
	w	0.69210 (1.98518)	0.69435 (1.98963)	0.69467 (1.98247)	0.69244 (1.95779)	0.69418 (1.97721)	0.69463 (1.97498)	0.69342 (1.97690)	0.69470 (1.96798)	**0.69533** (1.97960)
Detailed+Churn	m	0.66853 (1.88311)	0.68214 (1.96416)	0.68337 (1.96511)	0.68006 (1.97307)	0.68398 (1.99298)	0.68642 (1.99250)	0.68116 (1.97578)	0.68570 (1.97769)	0.68745 (1.99951)
	b	0.68727 (2.17243)	0.69394 (2.24122)	0.69570 (2.26702)	0.69151 (2.22530)	0.69678 (2.28071)	0.69874 (2.28581)	0.69388 (2.24807)	0.69858 (2.29075)	0.70053 (2.29489)
	w	0.69777 (2.26049)	0.70078 (2.30778)	0.70206 (2.29377)	0.69872 (2.27626)	0.70276 (2.31399)	0.70423 (2.31065)	0.70066 (2.30364)	0.70405 (2.30301)	**0.70602** (2.33756)

Table A.2: Comparison in terms of AUC (lift at 0.5%) among different methods for the **postpaid** dataset. The results are sliced per different interaction granularity (summary,summary+churn,detailed,detailed+churn), temporal granularity (monthly,bi-weekly,weekly), walk length (10,30,50) and number of walks (10,30,50). The best score per row is marked in bold. The best overall score is marked in bold and underlined.

Interaction	Temporal	Walk Length (wl) & Number of Walks (nw)								
		wl10_nw10	wl10_nw30	wl10_nw50	wl30_nw10	wl30_nw30	wl30_nw50	wl50_nw10	wl50_nw30	wl50_nw50
Summary	m	0.75855 (3.60506)	0.76192 (3.62745)	0.76212 (3.62769)	0.75876 (3.56990)	0.76461 (3.61084)	0.76631 (3.63203)	0.76182 (3.57086)	0.76653 (3.63058)	0.76712 (3.64190)
	b	0.75855 (3.60506)	0.76192 (3.62745)	0.76212 (3.62769)	0.75876 (3.56990)	0.76461 (3.61084)	0.76631 (3.63203)	0.76182 (3.57086)	0.76653 (3.63058)	0.76712 (3.64190)
	w	0.76389 (3.91427)	0.76708 (3.91331)	0.76812 (3.90608)	0.76556 (3.91499)	0.76906 (3.90006)	0.77038 (3.91258)	0.76726 (3.88537)	0.77047 (3.89476)	**0.77191** (3.90488)
Summary+Churn	m	0.75808 (3.60554)	0.76283 (3.61325)	0.76337 (3.59976)	0.75868 (3.55208)	0.76497 (3.63203)	0.76669 (3.61613)	0.76188 (3.58700)	0.76647 (3.68067)	0.76773 (3.63588)
	b	0.76207 (3.84877)	0.76491 (3.81987)	0.76509 (3.84973)	0.76210 (3.81770)	0.76629 (3.83865)	0.76730 (3.82059)	0.76386 (3.80879)	0.76701 (3.84684)	0.76847 (3.85503)
	w	0.76466 (3.90126)	0.76717 (3.89452)	0.76821 (3.89693)	0.76559 (3.89982)	0.76932 (3.90680)	0.77056 (3.89308)	0.76736 (3.89524)	0.77078 (3.91258)	**0.77199** (3.88802)
Detailed	m	0.73661 (3.52511)	0.75548 (3.78037)	0.75860 (3.77797)	0.75309 (3.72065)	0.76105 (3.75533)	0.76316 (3.78856)	0.75697 (3.77989)	0.76353 (3.78760)	0.76492 (3.80060)
	b	0.75142 (3.81650)	0.75898 (3.87020)	0.76150 (3.89428)	0.75697 (3.85141)	0.76355 (3.86225)	0.76540 (3.86731)	0.75962 (3.84732)	0.76495 (3.85840)	0.76691 (3.85430)
	w	0.76150 (3.89645)	0.76413 (3.90175)	0.76485 (3.91090)	0.76304 (3.90054)	0.76551 (3.93666)	0.76741 (3.93353)	0.76383 (3.90126)	0.76711 (3.91475)	**0.76952** (3.92077)
Detailed+Churn	m	0.73641 (3.50175)	0.75557 (3.76183)	0.75871 (3.77941)	0.75370 (3.70307)	0.76163 (3.74401)	0.76354 (3.78278)	0.75657 (3.73293)	0.76343 (3.75533)	0.76579 (3.80421)
	b	0.75121 (3.80036)	0.75923 (3.86731)	0.76168 (3.87598)	0.75703 (3.85503)	0.76332 (3.86562)	0.76571 (3.82685)	0.75999 (3.84515)	0.76517 (3.84612)	0.76754 (3.86490)
	w	0.76135 (3.88585)	0.76401 (3.90536)	0.76511 (3.93450)	0.76310 (3.90704)	0.76560 (3.89115)	0.76763 (3.89428)	0.76394 (3.90343)	0.76747 (3.88922)	**0.76957** (3.90873)

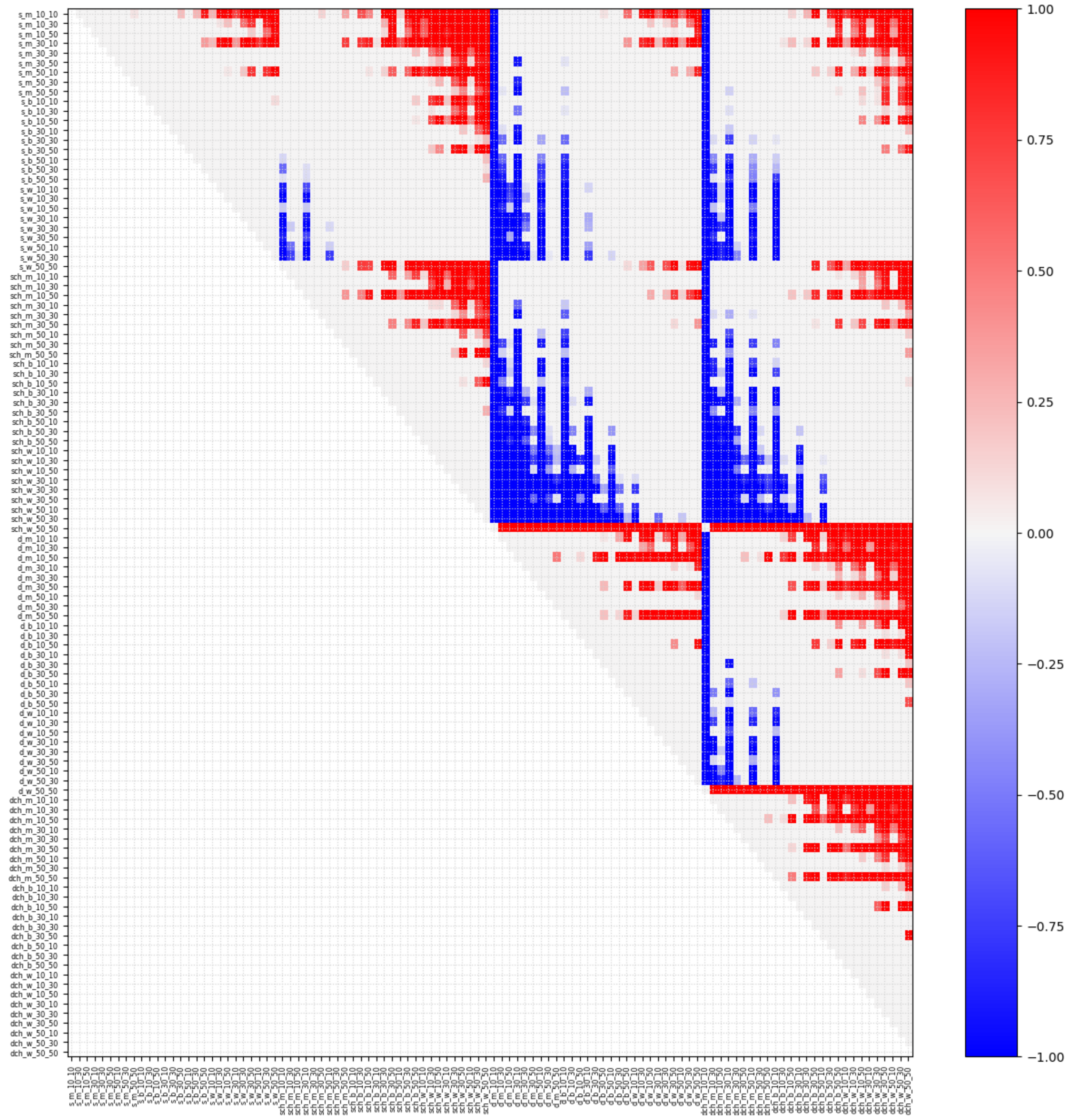


Figure A.1: Results of pairwise Bayesian hierarchical correlated t-test represented in the form of upper triangular matrix. The color bar ranges from -1 to 1, whereby negative values (in blue) denote that row method performs practically better than the column one and positive values (in red) denote that column method performs practically better than the row one. The absolute value (i.e. intensity of the color) corresponds to originally obtained posterior probability. Methods which are practically equivalent within ROPE=1% have posterior probability around 0 (in gray).