Web Tracking Detection with ML Classification model

Mentors: Dr. Ram Prasad Padhy, Dr. Jagadeesh Kakarla

Team Members: Yash Vardhan Goel CS20B1087 , Sachin Sai Reddy CS20B1088, Manoj CS20B1018,

Karan Kishore CS21B1002, Omkar Pujanji ME20B2031

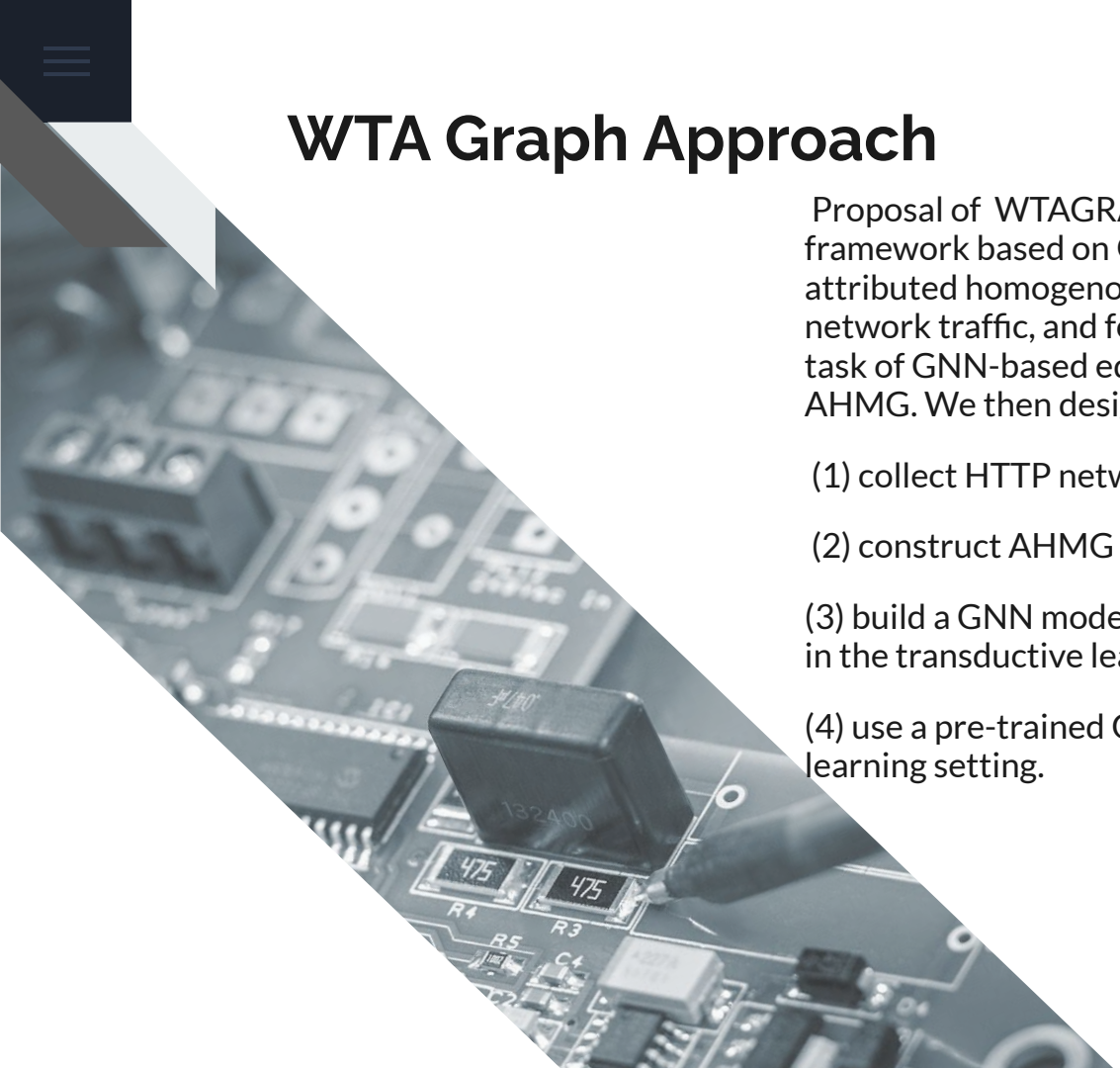
Overview

Web tracking and advertising (WTA) nowadays are ubiquitously performed on the web, continuously compromising users' privacy. Existing defense solutions, such as widely deployed blocking tools based on filter lists and alternative machine learning based solutions have limitations in terms of accuracy and effectiveness.

Common Methods of web tracking are:

- URL Tracking
- Pixel Tracking
- Cookies

WTA Graph Approach



Proposal of WTAGRAPH, a web tracking and advertising detection framework based on Graph Neural Networks (GNNs). We first construct a attributed homogenous multi-graph (AHMG) that represents HTTP network traffic, and formulate web tracking and advertising detection as a task of GNN-based edge representation learning and classification in AHMG. We then design four components in WTAGRAPH so that it can

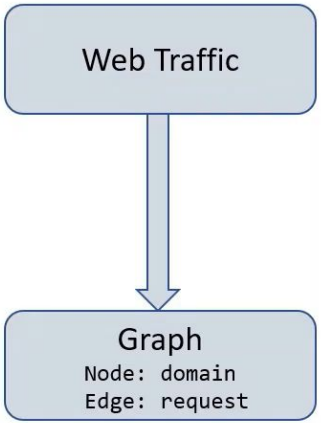
(1) collect HTTP network traffic, DOM, and JavaScript data,

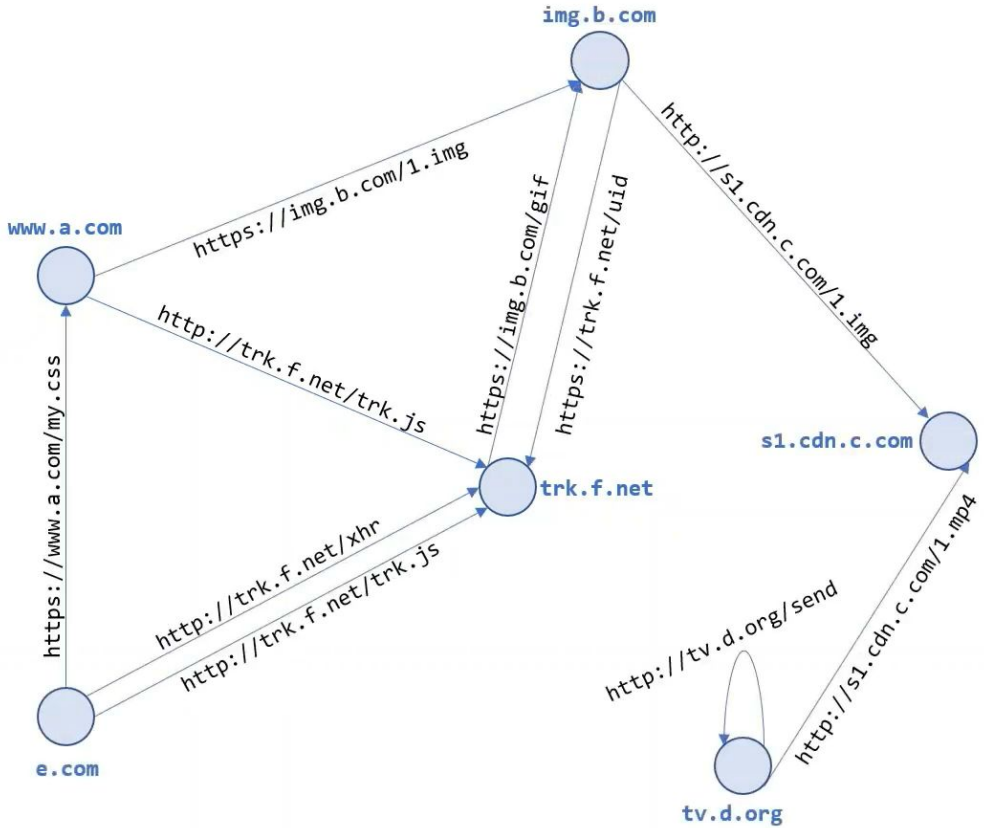
(2) construct AHMG and extract corresponding edge and node features

(3) build a GNN model for edge representation learning and WTA detectio in the transductive learning setting

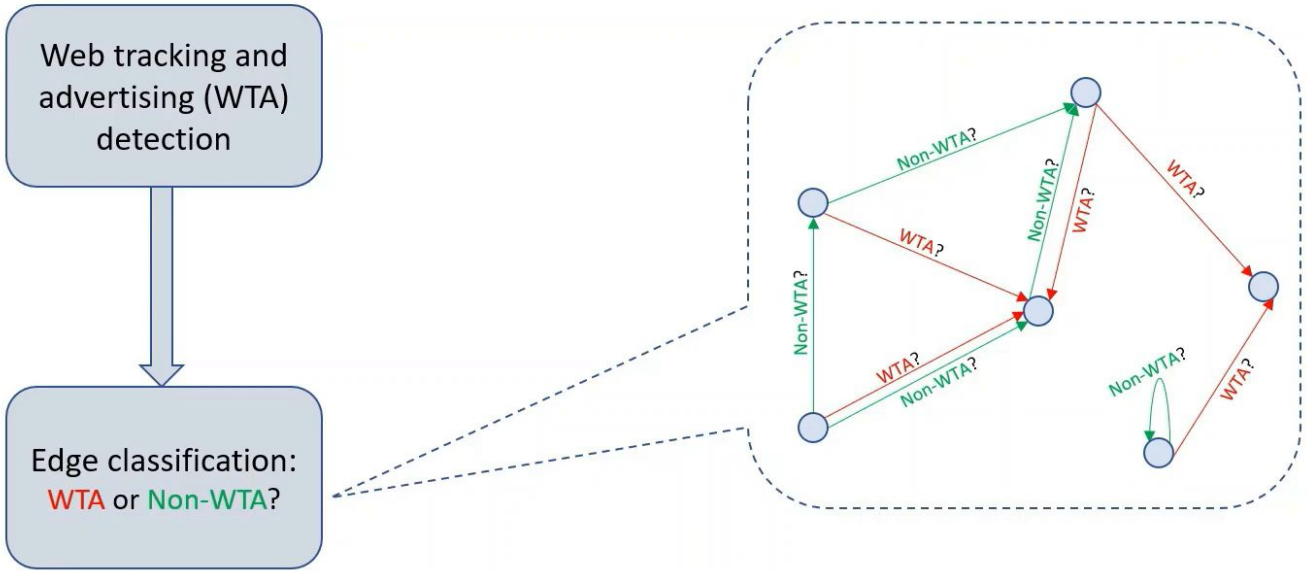
(4) use a pre-trained GNN model for WTA detection in the inductive learning setting.

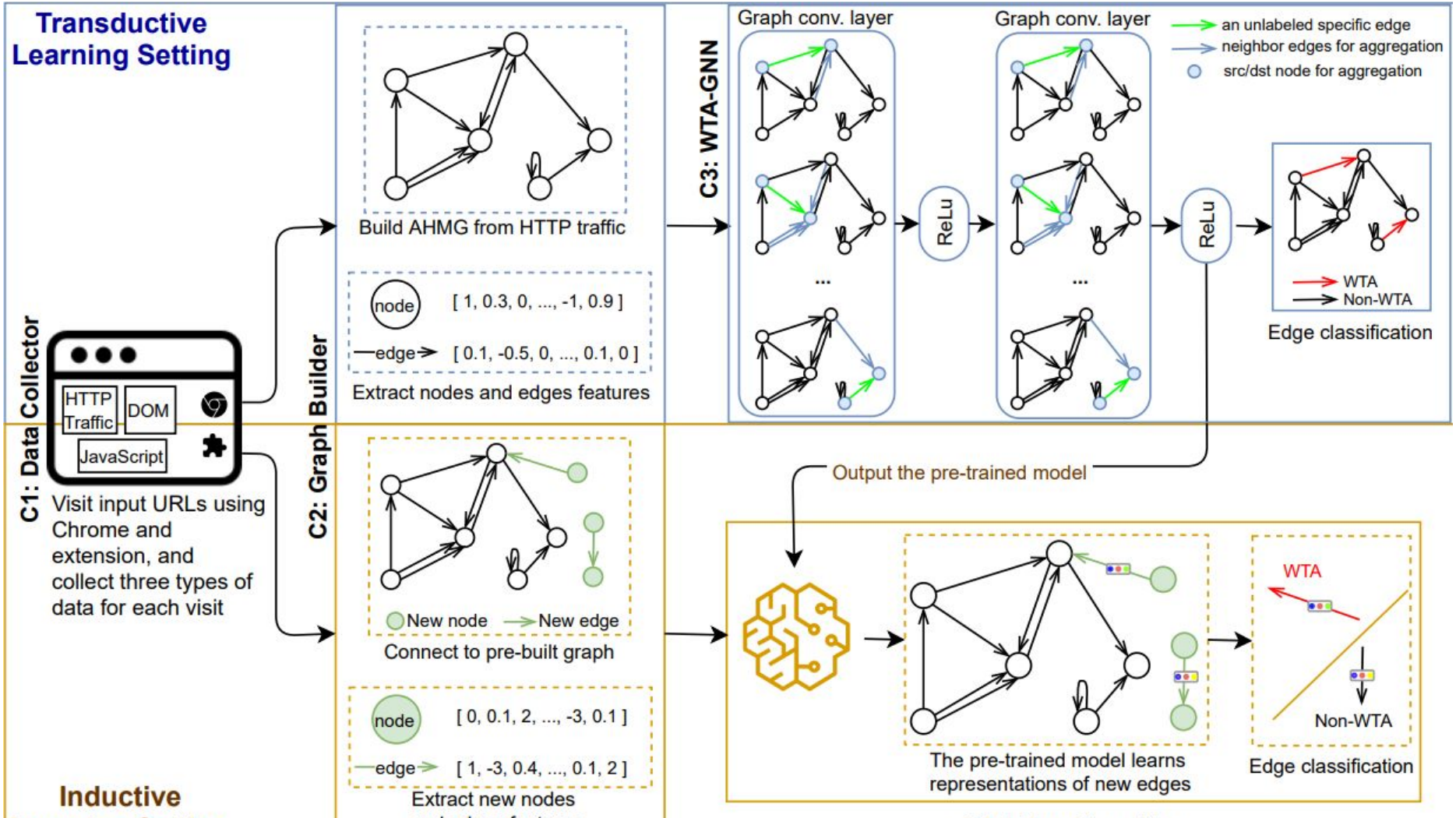
Intuition


Web Traffic
Graph
Node: domain
Edge: request


img.b.com
www.a.com
https://img.b.com/1.img
http://s1.cdn.c.com/1.img
http://trk.f.net/trk.js
https://img.b.com/gif
https://trk.f.net/uid
https://www.a.com/my.css
s1.cdn.c.com
trk.f.net
http://trk.f.net/xhr
http://trk.f.net/trk.js
http://tv.d.org/send
http://s1.cdn.c.com/1.mp4
e.com
tv.d.org

Problem Formulation




Transductive Learning Setting
C1: Data Collector
HTTP Traffic
DOM
JavaScript
Visit input URLs using Chrome and extension, and collect three types of data for each visit
C2: Graph Builder
Build AHMG from HTTP traffic
node
[1, 0.3, 0, ..., -1, 0.9]
edge
[0.1, -0.5, 0, ..., 0.1, 0]
Extract nodes and edges features
C3: WTA-GNN
Graph conv. layer
ReLu
Graph conv. layer
ReLu
an unlabeled specific edge
neighbor edges for aggregation
src/dst node for aggregation
WTA
Non-WTA
Edge classification
Output the pre-trained model
New node
New edge
Connect to pre-built graph
node
[0, 0.1, 2, ..., -3, 0.1]
edge
[1, -3, 0.4, ..., 0.1, 2]
Extract new nodes
The pre-trained model learns representations of new edges
WTA
Non-WTA
Edge classification
Inductive

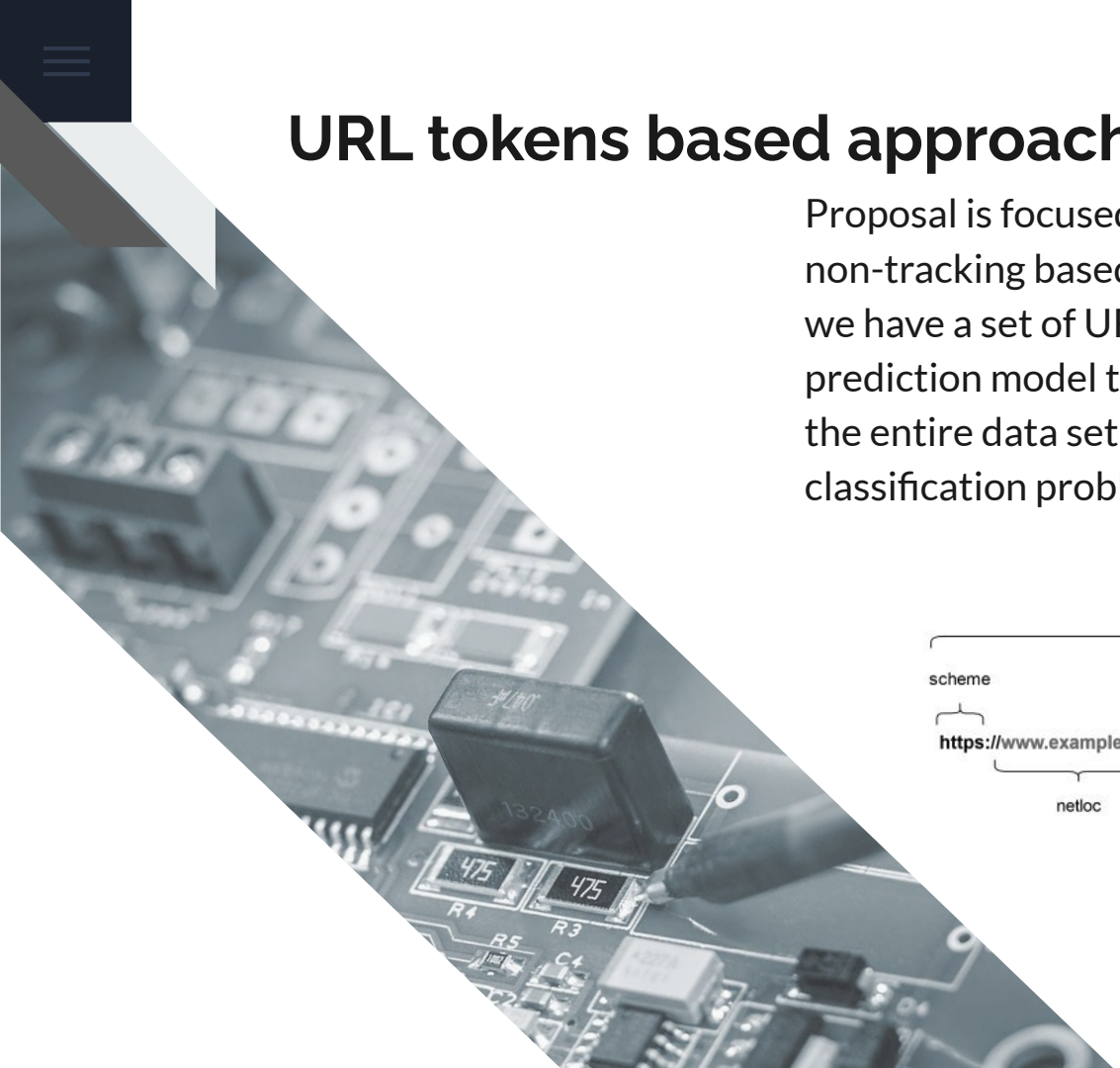
Need of URL token based tracking

- Complex combinations of JavaScript code attribution, DOM code inspection and network requests annotation to detect tracking pattern signatures.
- These approaches are very specialized and complex, and usually require major modifications to the browsers, fact that prevent common users to adopt these solutions.

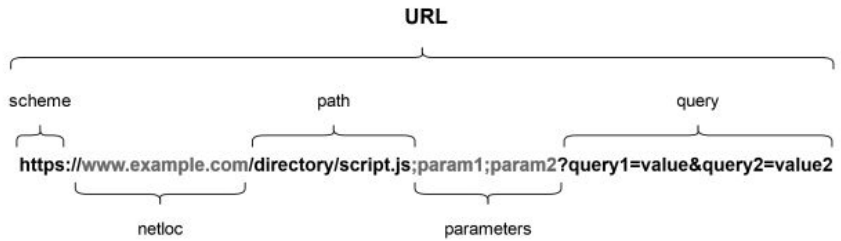
- On the other hand, URLs are very easy to access since it is not necessary to load the whole page or to explore any other exchanged data.

- Similarly, lexical features like the URL length can be measured fast, while other features such as network properties (e.g. IP Address, HTTP headers) or content properties require explicitly to download the content hosted by the URL to perform the inspection, increasing the risk and slowing down the process.
- Moreover, given that our method does not use code inspection, it is more robust against code minification and obfuscation, which is used by some JavaScript trackers to avoid detection

URL tokens based approach



Proposal is focused on classifying a given URL as tracking or non-tracking based only on the contents of the URL. To achieve i we have a set of URLs previously labeled, and we need to find a prediction model that minimizes the total number of mistakes in the entire data set. We are facing the problem as a binary classification problem



Step 1: Preprocessing

- Based on our expertise we selected a maximum length value of 200 characters, usually longer enough to contain all the URL. According to our observations, usually the most relevant information is contained in the parameters and query parts of the URL.
- If the URL is longer than 200 characters the system truncates the first elements containing the scheme, domain and path information until only 200 characters remain.
- On the contrary, if the URL is shorter than 200 characters we pad it filling it with zeros at the beginning of the array until the total length becomes 200. This way the system keeps the most important information always at the end of the array

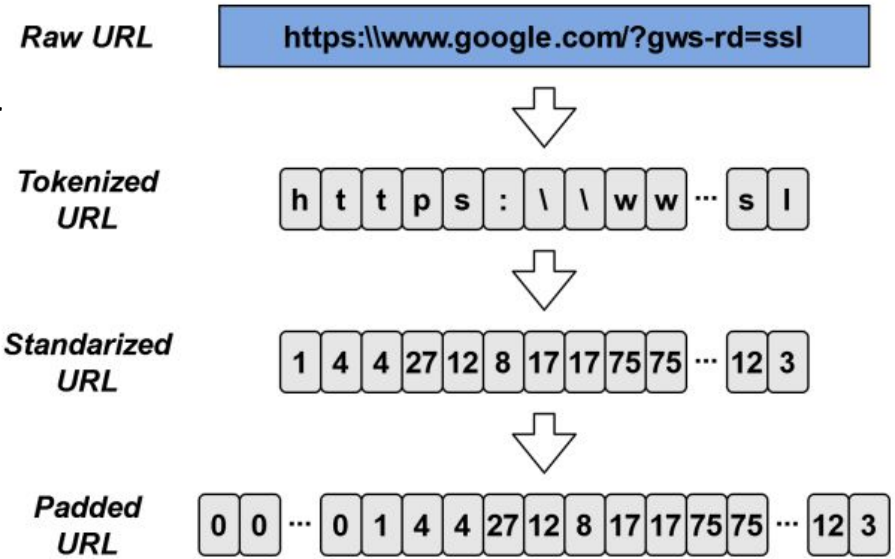
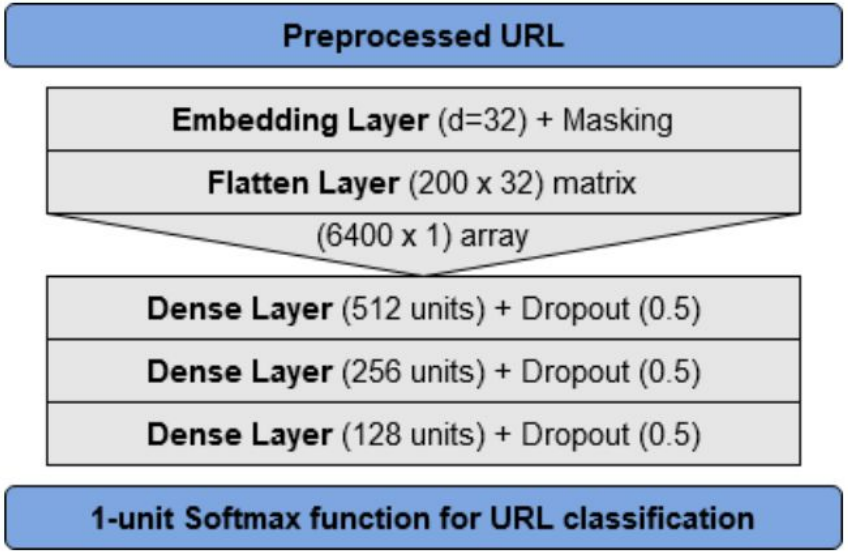


Fig. 2. URL preprocessing

Step 2: Model Architecture

- By default deep learning treat the inputs as numerical values, interpreting that 2 close values are similar.
- To fix it we add a first Embedding Layer, transforming the preprocessed URLs to a low-dimensional continuous representations
- We add a new Flatten Layer that reduces the embedding output to a 1-dimension array, In order to find the optimal number of layers and units per layer we executed a Grid search with multiple values.
- The output is composed of a 1-unit layer with a softmax activation function giving a float output $\in [0, 1]$,then rounded to the closest integer (0 for non-tracking or 1 for tracking).



Thank you!

References-

https://www.youtube.com/watch?v=Fzg47R8bjvs

https://zhiju.me/assets/files/WtaGraph_SP22.pdf