WHITE PAPER

Exploring URL Vectorization to Understand User Intent for Online Advertising

Prepared by: Dr. Melodie Du, Senior Data Engineer & Data Scientist

APRIL 2023

Artificial Intelligence Designed for the Future of Advertising

Executive Summary:

In this article, Inuvo Senior Data Engineer & Data Scientist **Dr. Melodie Du** explores enhancing the understanding of webpage content at scale using machine learning through vectorization. This topic is increasingly relevant as most web-based advertising impressions move way from persistent identifier-based targeting. In a new cookie-free world, understanding consumer intent in privacy-safe ways is crucial. The main challenge is extracting meaningful concepts from webpages and scoring them consistently across similar pages.

To comprehend consumer intent, Dr. Du employs Natural Language Processing (NLP) techniques, such as breaking down text into N-grams for evaluation and scoring. This process generates up to 300 concepts per webpage. Along high-level categories, demographics, and other statistics, these concepts create a "profile" for each URL. These profiles hold valuable user intent data for targeting. With billions of web pages to index, and each profile containing extensive information, Dr. Du suggests using "feature engineering" to transform raw page profile data into a more manageable form called a "vector" that enhances processing speed and efficiency.

The vectorization process can be **compared to compressing jpeg images;** the goal is to remove unnecessary signals and reduce the URL profile's size and complexity without losing resolution. The vectorized profile should retain essential details but be significantly easier for Machine Learning (ML) algorithms to process.

Dr. Du discusses methods for vectorization, including one-hot encoding and experienced selection. However, one-hot encoding has challenges like computational expense and high variance. To address these issues, Dr. Du proposes strategies like experienced selection, term frequency - inverse document frequency (TF-IDF), and hashing to reduce dimensionality while preserving important concepts in the feature engineering phase.

Utilizing suitable feature engineering and vectorization methods can improve the efficiency and accuracy of advertising campaigns by combining advanced contextual understandings of web pages with AI powered audience insights that allow accurate targeting on all devices without the use of cookies or persistent identifiers.

Exploring URL Vectorization to Understand User Intent for Online Advertising

Prepared by: Dr. Melodie Du, Senior Data Engineer

1. The Increasing Importance of Understanding User Intent

Since the very early days of online advertising, companies have been exploring ways to infer user intent from page content. Before the rise of “behavioral” advertising, contextual systems like Google AdSense were a major source of revenue for many web publishers. However with the rise of cookie-based user profiles and social media where users were logged in, ad networks began to build stable user activity profiles over a long period of time that could be used to target individual users in real-time. These methods ultimately proved more lucrative to publishers than the early contextual models, and to this day, most advertisers rely on ID based targeting for the bulk of their ads.

In this paper, Dr. Du will explore:

1. **Computational Advertising**
2. **Web Crawling & Concept Generation**
3. **URL Profiles**
4. **Feature Engineering**
5. **Embedding**

In the past few years, however, the sea of change has shifted considerably. First, privacy legislation such as GDPR and CCPA have made individual user targeting more legally complex. Second, Apple has effectively deprecated the use of third-party cookies and individual identifiers both for user-targeting and for conversion attribution.

As a result, the majority of programmatic ad impressions no longer have a stable user profile, and attribution has become much more difficult. Today, predicting user intent without identity-based profiles is thus extremely crucial. When a company must predict intent quickly, the content and context of individual URL pages has once again become a critical component. But the technology for doing this has advanced considerably since the early days.

> **"Today, predicting user intent without identity-based profiles is (...) crucial. When a company must predict intent quickly, the content and context of individual URL pages has once again become a critical component."**

2. Web Crawling and Concept Generalization

The first step to being able to use URLs as an effective audience targeting tool is to know which URLS actually exist. The process of crawling URLs is relatively straightforward now, as there are numerous open source web crawlers online.

The difficulty is not how to find the information, but instead how to extract and group the content of the pages as meaningful concepts and score them reliably and efficiently across all the other webpages.

To begin this process, Inuvo relies on Natural Language Processing (NLP). We break the text in the webpage into N-grams (N consecutive tokens), each of N-grams is evaluated and scored accordingly within the pool of web pages that have the similar topics.

N-grams in the article refer to a sequence of words or tokens found in the text. The "N" in N-grams represents the number of words or tokens in the sequence. They are used to analyze and understand the structure and content of a text.

Let's break it down with a simple example. Consider the sentence:

"The cat is cute."

N-Gram	Analyzation Method
1-grams (unigrams)	Analyze each word separately: **"The," "cat," "is," "cute."**
2-grams (bigrams)	Pairs of consecutive words: **"The cat," "cat is," "is cute."**
3-grams (trigrams)	Three consecutive words: **"The cat is," "cat is cute."**

These N-grams are the building blocks to extract concepts. In other words, concepts are the combination of several N-grams, wherein the score of each contributes to the final score of the concept.

By filtering and evaluating, each webpage can generate up to 300 concepts. The concepts can be expanded based on our knowledge and scoring of adjacent concepts to include more insights.

3. A URL Profile is Born

The goal is to replace the general targeting of user profiles, which rely on increasingly privacy-unsafe identifiers, with targeting based on URL profiles. Using the detailed concept map built of the NLP process described above, and combining those concepts with expanded (related) concepts, demographics, and other statistical models, we're able to build a complete URL profile that can subsequently be used for targeting. More importantly, **this URL profile is entirely agnostic to 3rd-party cookies and persistent unique identifiers.**

It is estimated that a majority of online ad impressions no longer carry such identifiers, so advertising technology companies that rely on identifiers simply cannot effectively bid on inventory without them. The remaining inventory, often consisting of iPhone and Mac users, offers great value to advertisers because fewer advertisers are able to reach them.

> **The remaining inventory, often consisting of iPhone and Mac users, offers great value to advertisers because fewer advertisers are able to reach them."**

4. Feature Engineering

These URL profiles can be directly used by Machine Learning (ML) advertising algorithms. However, this process can be made more efficient by employing Feature Engineering. The idea behind Feature Engineering is to run each URL profile through a series of steps in order to select/filter and transform the raw URL profile data to a standardized vector that ML models more efficiently process in batch.

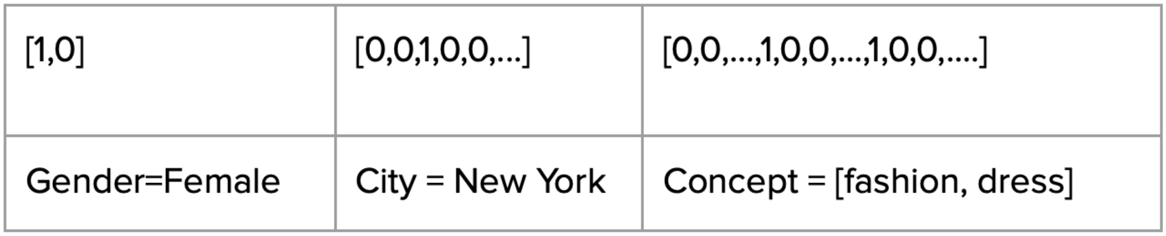
When a URL is crawled, concepts can be thought of as a set of keywords that are collected as a list. These keywords are categorical features (string) that need to be transformed to numerical numbers for the model to read and weigh more easily.

A simple metaphor for why vectorization is preferable is compression in digital photography. Raw digital images contain all of the information that the camera or sensor was able to capture, and are thus quite large. Good compression algorithms can preserve all of the visual detail of the images while reducing their size by 80-90%. A person viewing the end photo doesn't have an issue, because all of the details that have been lost do not make the image worse in any appreciable way.

> **The goal of vectorization is to preserve the important elements of the raw URL profile while making them easier for ML models to consume. "**

The most common way to vectorize the categorical feature is one-hot encoding. In a simple example, a web page has a higher association with females, is based in New York, and is mapped to concepts of fashion and dress.

The feature vector is constructed as in Figure 4.1. In this way, each position of the vector represents the existence (label as 1) of the corresponding concept or not (label as 0). The problem is the vector space is usually very large (2 million concepts in total) and sparse. It is computationally expensive and may have high variance in training models.



[1,0]	[0,0,1,0,0,...]	[0,0,...,1,0,0,...,1,0,0,....]
Gender=Female	City = New York	Concept = [fashion, dress]

Figure 4.1 One-Hot Encoding

In order to optimize such situations, several methods can be used (individually or all together) to increase the efficiency.

At the early stage of the modeling, a method called "experience selection" is always a good starting point. **Experience selection** can be thought of as a filter based on past experience.

Inuvo has experience in different campaigns. In a scenario like this, a hot list of concepts and a blacklist of concepts may be built to prioritize the concepts we would like to train in the model. This can be done by hand as well, and while hand-picking has high precision, it is also slow and has a high miss rate. Our historical data helps to make this more automated, but other methods are needed as well.

The next method that can be used is TF-IDF. **Term Frequency - Inverse Document Frequency (TF-IDF)** is a technique originally used in Natural Language Processing (NLP), and is defined by the frequency of a concept divided by the total number of occurrences in all the websites we are crawling (Equation 4.1).

$$TF \cdot IDF = \frac{TF}{DF}$$

Equation 4.1

Let's look at a simple example where two concepts - **baseball** and **weather** - have each been seen on espn.com a total 20 times. Let's assume that **baseball** has a total number of 40,000 occurrences outside of ESPN, while **weather** has been found 100,000 times. Since the proportion of observed instances over total is larger for **baseball** (20 / 40,000) than **weather** (20 / 100,000), the concept of **baseball** has a higher probability of representing espn.com.

Handling the removal of information should always be done cautiously, as it may still hold valuable insights that just aren't abundant enough to be statistically significant at the moment.

One way to address this is to use hashing to allow these concepts to share a lower-dimensional vector space during the modeling process. For instance, one could choose 8,000 less frequent concepts and map them to a 20-element vector. Although some concepts will end up with the same feature vector, their rarity and low statistical significance won't harm the overall accuracy of the model. Instead, they can still contribute positively to the model's performance.

5. Embedding

Embedding is a process that converts high-dimensional sparse vectors into lower-dimensional dense vectors. **This technique has gained prominence with the rise of deep learning, as it forms the foundation of these advanced algorithms.** Moreover, embedding has numerous applications beyond neural networks, making it a versatile and essential tool in the field.

The Importance of Embedding

Embedding is a method that uses a low-dimensional vector to represent various objects, such as vectors, items, words, movies, and anything else with relationships to other objects that you want to model. Remember when using one-hot encoding in vectorization, the resulting vector is very sparse. Embedding systematically maps the input to a denser vector space that is more suitable for modeling.

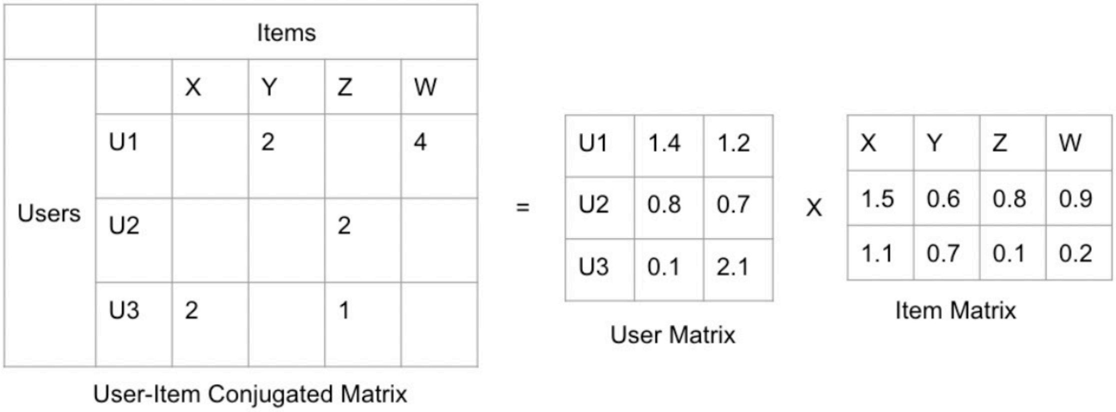
Based on how you define the relationships between items, the embedding itself contains valuable information that can be used for quickly filtering candidates and identifying similar objects.

Methods for Embedding

Pre-Deep Learning Methods

Matrix Factorization: Matrix factorization gained its reputation in the Netflix Prize Challenge, 2006. The main idea is to decompose the conjugated matrix of user and item to two matrices. One matrix can be represented as the item matrix, the other one is the user matrix (Figure 5.1). The dimension of the user/item matrix is a hyperparameter, which can be adjusted depending on the needs. In this example, user 1 has the embedding vector [1.4, 1.2] and item X has the embedding vector [1.5, 1.1]. The inner product of a user and an item is the predicted score of the user-item interaction. Stochastic Gradient Descent (SDG) is the industry way to calculate the vector.

Using cosine similarity, we can quickly identify similar items and users in the surrounding area. The potential disadvantage of matrix factorization (MF) is that it overlooks the properties of users and items. This is particularly true if the user or item has already been vectorized.



		Items			
		X	Y	Z	W
Users	U1		2		4
	U2			2	
	U3	2		1	

User-Item Conjugated Matrix

=

U1	1.4	1.2
U2	0.8	0.7
U3	0.1	2.1

User Matrix

X

X	Y	Z	W
1.5	0.6	0.8	0.9
1.1	0.7	0.1	0.2

Item Matrix

Figure 5.1 Matrix Factorization

Factorization Machine (FM): Another method of embedding is called Factorization Machine (FM). FM was proposed in 2010. The idea is to fit a vector of each feature of the item/user. Compared with Matrix Factorization, where each user/item is a vector, in FM, each feature is a vector. The interaction between a user and an item is actually the interaction between every pair of the features (Equation 5.1).

$$\hat{y} = \omega_0 + \sum_{i=1}^{n} \omega_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_i , v_j > x_i x_j$$

Equation 5.1

Similarly, the weights and latent vector (v) can be solved using gradient descent (GD). The factor number (the dimension of the vector) is a hyperparameter that is usually much smaller than the feature vector. FM is generally light-weight and easier to implement online than MF.

GBDT + LR: Gradient Boosting Decision Tree (GBDT) with Logistic Regression (LR) differs from matrix factorization (MF) and factorization machines (FM) because GBDT doesn't rely on fitting latent vectors. GBDT + LR can be thought of as using several decision trees to fit a sample incrementally. Each time, a training sample ends up in one of the tree's leaves (Figure 5.2).

For instance, let's say sample X represents a woman aged 20-40, living in New York with a pet, and we want to determine if she is interested in buying a car. In this example, two decision trees with a depth of three are used to fit the model. The first time, the sample falls into the fourth leaf of the first tree, and the second time, it falls into the third leaf. The resulting encoded embedding vector is [0,0,0,1,0,0,1,0]. It's important to note that the original feature vector is much sparser and higher-dimensional than the GDBT-generated embedding vectors. Finally, the embedding vectors are used as input for logistic regression to predict user behavior.

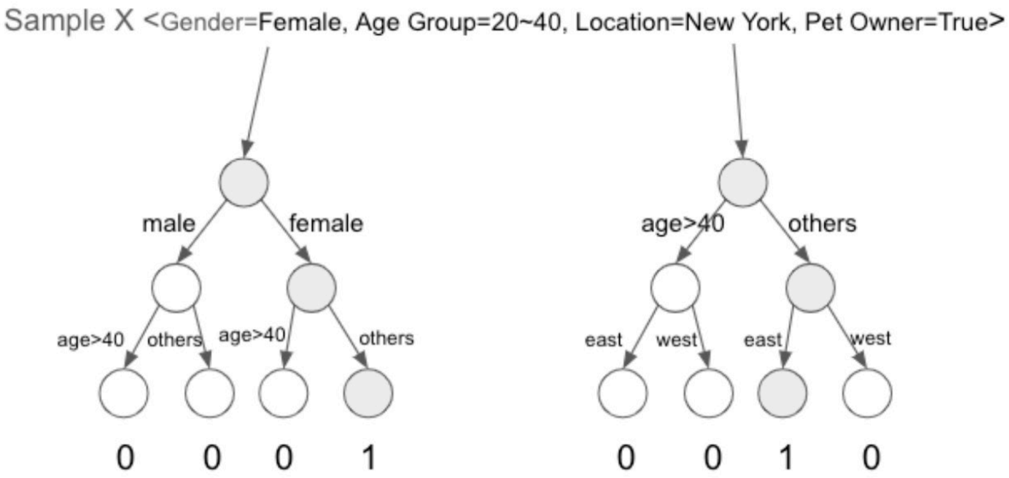

Sample X <Gender=Female, Age Group=20~40, Location=New York, Pet Owner=True>
male
female
age>40
others
age>40
others
0
0
0
1
age>40
others
east
west
east
west
0
0
1
0

Figure 5.2 GBDT for Generating Embedding Vectors

Deep Learning Methods

Deep learning has advanced rapidly in recent years, but this doesn't mean that pre-deep learning methods are obsolete. In fact, many of these earlier methods serve as building blocks for more complex models.

In deep learning, **embeddings can be trained independently or alongside the main model.** There are several advantages to training embeddings as a standalone process. First, embeddings are relatively stable and can be trained efficiently once a month. Second, the generated embeddings carry information that can be used for candidate generation or other filters. Third, the parameters in the embeddings are significantly larger than those in the model, so training them together can reduce efficiency.

Word2Vec and Item2Vec are classic methods for generating embeddings. They are based on the sequence of the target. For instance, if a user visits toyota.com after honda.com, the two sites can be considered a pair. During the training phase, one-hot encoding is used for all possible URLs. If there are 1,000 URLs, the encoded input vector will have 1,000 dimensions. In the example above, the input is the one-hot encoded vector for toyota.com, and the output is honda.com (Figure 5.3).

By updating the weights of the hidden layer, the mapping matrix (W) becomes a collection of embedding vectors.

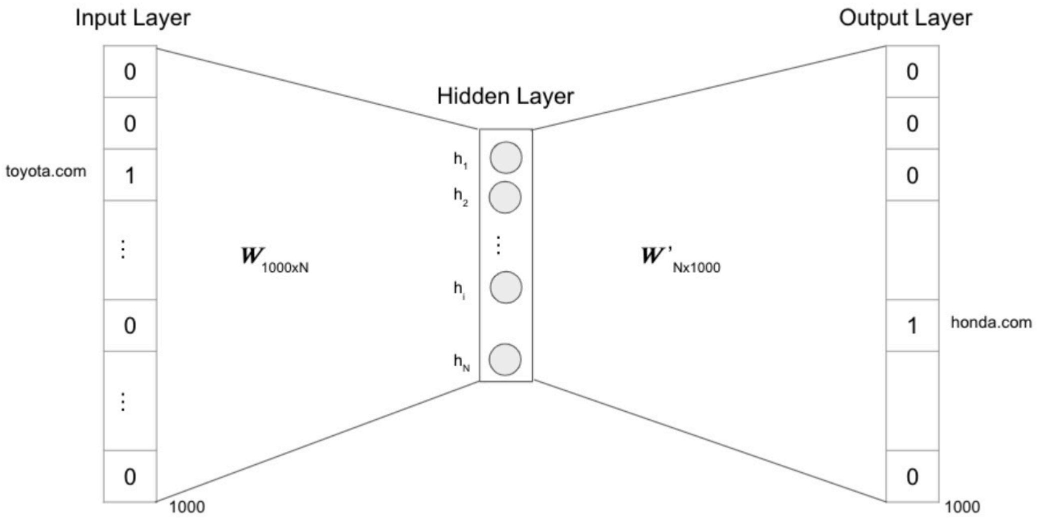


Figure 5.3 Word2Vec Neural Network Structure

Embeddings can also be generated during model training. For example, the two-tower neural network initially trains user features and item features separately, then concatenates the two vectors together as a single input for the final prediction stage (Figure 5.4).

In addition to the two-tower structure, **many classic deep learning models generate embeddings within the model, such as Deep Crossing, Factorization-Machine Supported Neural Network (FNN), and Wide & Deep.** The advantage of this approach is that the weights can be used directly to represent the input.

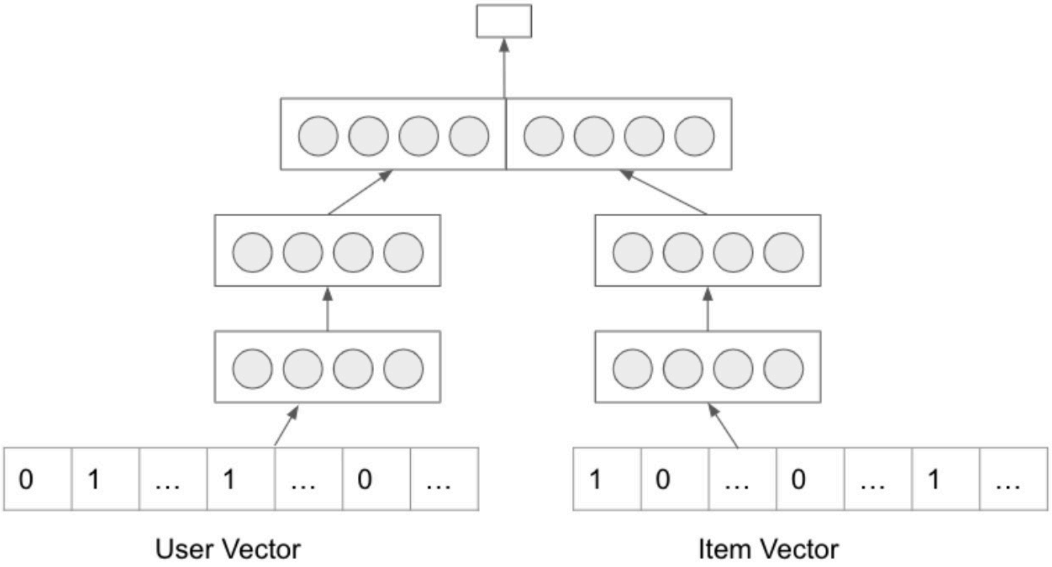


Figure 5.4 Two-Tower Structure of Neural Network

Local Sensitive Hashing for Quick Similarity Clustering/Ranking

Embeddings are low-dimensional, dense vectors that represent items (e.g., URLs). If two vectors are similar, it usually implies that the items are similar as well. Although cosine similarity can be used to calculate the distance between two vectors, it's computationally expensive, as each pair of vectors needs to be calculated and ranked. **Locality Sensitive Hashing (LSH) offers a more efficient way to calculate similarity.**

The idea is simple: if two points are closer to each other, their projections in a lower-dimensional space should also be closer. Conversely, if two points are far apart, their projections might not be as close. **By using a hashing function (like a projection function), we can quickly identify similar vectors, and within that subset, we can perform a more precise and efficient ranking.**

Final Thoughts:

Computational analysis of page content has come a long way since Google AdSense dominated the industry. As the above research shows, many new optimization and feature engineering processes have evolved over time that can be used to improve both targeting and performance. Whereas before, it was accurate to think of this targeting as "contextual," the ability to more deeply profile content and to understand adjacent content profiles make for targeting that goes well beyond the content on the page. This is especially the case when url profiles are combined with other inputs.

At Inuvo, we combine our custom url profiles with other custom machine learning models we have built for both creatives and user sessions. We produce each model independently but combine them at the time of ad execution to know when we should buy a given ad impression and to determine the very best individual ad to show when doing so.

About Dr. Melodie Du

Dr. Du is a data scientist and data engineer with a proven track record in machine learning and statistics. After joining Inuvo in 2018, she has garnered notable success in audience targeting performance improvements and building an integrated data pipeline.

As Senior Data Engineer, Dr. Du optimized the machine-learning data pipeline and updated the IntentKey concept graph contributing to the overall data-driven, decision-making capabilities. Melodie previously worked as a Software Engineer at Ucodec Inc. on High Efficiency Video Coding software.

Dr. Du has Bachelor of Engineering in Biomedical/Medical Engineering from Tsinghua University and a PhD in Biomedical/Medical Engineering from Purdue University where she served as a Data Science Research Assistant.

In her personal life, Melodie is an avid reptile rescuer, particularly of giant tortoises, with knowledge of 20+ reptile species and subspecies. She was also an ice climber before retiring from the hobby due to injury.

inuvo®

inuvo.com