Constrained Non-Affine Alignment of Embeddings

Yuwei Wang[1], Yan Zheng[2], Yanqing Peng[1], Chin-Chia Michael Yeh[2], Zhongfang Zhuang[2],
Das Mahashweta[2], Bendre Mangesh[2], Feifei Li[1], Wei Zhang[2], Jeff M. Phillips[1]
[1]*University of Utah,* [2]*Visa Research*
{ywang, ypeng, lifeifei, jeffp}@cs.utah.edu
{yazheng, miyeh, zzhuang, mahdas, mbendre, wzhan}@visa.com

***Abstract*—Embeddings are one of the fundamental building blocks for data analysis tasks. Embeddings are already essential tools for large language models and image analysis, and their use is being extended to many other research domains. The generation of these distributed representations is often a data- and computation-expensive process; yet the holistic analysis and adjustment of them after they have been created is still a developing area. In this paper, we first propose a very general quantitatively measure for the presence of features in the embedding data based on if it can be learned. We then devise a method to remove or alleviate undesired features in the embedding while retaining the essential structure of the data. We use a Domain Adversarial Network (DAN) to generate a non-affine transformation, but we add constraints to ensure the essential structure of the embedding is preserved. Our empirical results demonstrate that the proposed algorithm significantly outperforms the state-of-art unsupervised algorithm on several data sets, including novel applications from the industry.**

***Index Terms*—embeddings, alignment**

I. Introduction

An embedding is a moderate-dimensional vector representation of an entity where many features may be captured. These embeddings are often distributed representations, meaning that in most cases the correspondence between features and vector coordinates is not a one-to-one mapping. This flexibility allows for better capturing of correlation, the representation of potentially more features than dimensions, and the emergence of a more interesting global structure. However, it also eschews the interpretability and transparency of analysis of this data, and may result in unwanted associations. Yet, due to its efficiency and effectiveness in representing data, embedding learning technology has been an essential part of a wide variety of data domains. In particular, word embedding methods such as Word2vec [1] have been widely used in natural language processing to capture the semantic and synthetic information about the words. In network management, node embeddings characterize the structures of network nodes [2]; while graph embeddings encode rich information of a graph [3]. Image embedding representations capture essential structural information about individual features [4] or an aggregate of them [5]. Recent works [6], [7], [8], [9], [10] also showed that these embedding methods could be applied to even broader ranges of data types.

Such general embedding schemes have been proven to capture data characteristics without knowing the feature values associated with these characteristics. However, it is still a challenge to recognize when and which features are captured in these embeddings, and how to explicitly identify them, or attenuate them if they are undesired. In particular, depending on the downstream task, there might exist undesired known features that have a large impact when analyzing the embedding. In this case, we would like to eliminate their influence in order to improve the performance of downstream tasks.

We provide powerful methods for recognizing and mitigating such implicitly captured and undesired embedding features. We summarize our contributions as follows:

- We provide a new method to evaluate the presence and significance of a categorical feature captured by an embedding dataset. Our evaluation is based on how easily a dataset with multiple labels can be classified.
- We then propose UCAN (Unsupervised Constrained Alignment that is Non-affine), an effective methodology for removing or alleviating the impact of such a significant feature in the embedding. It uses a variant of a Domain Adversarial Network (DAN) to find a non-affine alignment that obfuscates that feature, but also includes constraints to ensure the original meaning and structure of the embedding is retained. Unlike prior work targeting this goal, the data transformation we learn is not restricted to an affine transformation of part of the data, so it is significantly more powerful, yet the added constraint prevents data deprecation.
- We demonstrate the effectiveness of this process on several embedding datasets representing airports, language, and merchants. We identify nuisance features, and then train a generator to dampen their effect while retaining the core structure of the embedding. The usefulness is shown on several downstream tasks, outperforming state-of-the-art methods.

II. Related Work

Several existing works are focusing on transforming existing embeddings for goals such as attenuating bias. It has been observed that there exists certain (for instance, gender-associated) bias inherent in word embeddings [11], [12], [13], [14], [15]. In these lines of work, they focused on non-dominating features such as gender, ethnicity, age, and sentiment. In some cases, they describe simple projection-based [13] approaches towards removing the impact of these effects. Another line of work focuses on bilingual and multi-language embedding alignment. Cross-lingual word embeddings are appealing as they can compare the meaning of

words across languages and model transfer between languages (e.g., between resource-rich and low-resource languages, by providing a common representation space). However, if the word embedding is trained directly on merged documents from multiple languages, words from the same language tend to cluster due to similar language contexts. As a result, the language of a word becomes the dominating feature embedded in the resulting word embedding. To address this issue, several approaches have been proposed to learn bilingual dictionaries mapping from the source to the target space and align them into the same space using lexicon or a sample of lexicon [16], [17], [18]. In particular, [19] learns an initial linear mapping in an adversarial way by additionally training a discriminator to differentiate between projected and actual target language embeddings. [20] extends this line of work to represent words from multiple languages in a single distributional vector space. This line of work also applies a domain adversarial network. Our work differs from both of these lines in that it is not restricted to affine transformations and can learn more complex structures of embeddings.

There are also similar works from fairness perspective [21], [22], [23], [24]. They aim to generate fair data using adversarial learning by retaining the ability to predict the label while reducing the possibility of predicting the sensitive or protective variables. In these works, the problem setting is different from ours: the dataset comes with labels; thus, the labels are also involved in the data generation process, while our algorithm only deals with the embedding dataset and does not touch the original label. In addition, this line of work has few discussions on embedding datasets. Although [24] discusses in their paper on word embedding dataset, their experiments on this are limited to only show one specific example of the analogy task for the word embedding dataset.

III. Feature Measurement in Embeddings

Embeddings preserve the information of entities by placing similar entities close together in the embedding space, as measured by cosine distance. For example, word embedding captures the semantic and syntactic properties of words, and the embeddings of synonyms are close to each other in the embedding space. However, it is hard to tease apart features since all the features are entangled together, and there is no simple mapping between the dimensions and the features. Certain traits like polarity (e.g. good - bad) are not specifically encoded by some dimension or direction in a Word2vec embedding. However, for words with similar polarity, like 'good' and 'great', we observe that they have short cosine distances in the embedding space.

We are interested in a scenario where we can access a subset of features in the original data. In this case, we are looking for an effective way to measure how significantly an embedding is affected by a known categorical feature. We first assume that the *feature* is a binary function $F : \mathcal{D} \to \{0; 1\}$ for dataset $\mathcal{D}$ is with $0; 1$ labels. The significance of a feature indicates how easily we can distinguish the 0 and 1 points. We quantify the significance using a classifier:

Definition 1. *Consider an embedding generator $E : \mathcal{D} \to \mathsf{R}^d$ and a balanced feature $F : \mathcal{D} \to \{0; 1\}$. For a family of classifiers $\mathcal{C}$ on the embedding space, and a positive value ε with the following probability:*

$$\max_{C \in \mathcal{C}} Prob_{x \in \mathcal{D}}[C(E(x)) = F(x)] > 50\% + \varepsilon$$

We say that E embeds F with weight ε.

The idea behind this definition is straightforward. If an embedding generated by E does not contain any information about feature F, the values of $F(x)$ become random labels for $C(E(x))$. Therefore, most classifiers should achieve about 50% accuracy in expectation, on balanced data with half of each label. Conversely, if the embeddings can be classified on feature F with accuracy significantly above 50%, the embedding reflects some information of F. The higher accuracy the classifier can achieve, the more information is encoded in the embedding. While we typically cannot precisely find the actual maximum accuracy classifier $C \in \mathcal{C}$, we can use the result of a learning algorithm as a proxy.

This definition can be extended to multi-label features and classifiers. Suppose the number of labels is M, if the classification accuracy is above $1/M$, then the embedding E embeds feature F. Numerical features can also be binned to category features using predefined thresholds.

A. Imbalanced Datasets and AUC

In imbalanced datasets, the "Accuracy Paradox" occurs when we use the accuracy metric to learn the best model. Consider when a feature has 10% of the data with value 0, and 90% of the data with value 1. Then a simple classifier sets $C[E(x)] = 1$ for all x and will get the accuracy of 90%, so accuracy is not the best metric for imbalanced datasets. Thus, we use average one vs. all AUC as the metric to measure if a feature is embedded for both balanced and imbalanced data. For a feature $F : \mathcal{D} \to \{0; 1; :::; M-1\}$, and any classifier C, if $\mathrm{AUC}(C(E(x)) = F(x)) > \varepsilon$, then we say F *is embedded by* E *with weight* ε. We use this measure within this paper.

For fields using numerical values, we can set single or multiple thresholds to label values into different categories. Thus, the AUC metric can be applied to numerical features. The choice of the threshold depends on how much granularity of details needs to be preserved.

IV. Feature Attenuation and Retention

Given an undesired binary feature F on dataset $\mathcal{D}$, let $\mathcal{X}$ and $\mathcal{Y}$ be subsets of $\mathcal{D}$ divided by F, and $X; Y \subset \mathsf{R}^d$ be the two sets of corresponding embeddings. For instance, consider embeddings of various merchants generated by the similarity of the transaction sequences from their customers [10]. The subset $\mathcal{X}$ can be merchants in New York City (NYC), and $\mathcal{Y}$ can be merchants from Los Angeles (LA). To give recommendations for a customer from NYC while visiting LA, based on the embeddings, it would be useful to first remove the effect of the location feature F before doing so.

The goal of a Domain-Adversarial Network (DAN) [25] is to find a mapping function $G : \mathsf{R}^d \to \mathsf{R}^d$ (which is also our generator) for X without knowing the data of $\mathcal{X}$ and $\mathcal{Y}$, so that sets Y and $G(X)$ are indistinguishable. If there was a pattern among the features of X distinct from those of Y, it could potentially separate them. So as an added benefit, if an element $x \in \mathcal{X}$ is similar to some $y \in \mathcal{Y}$ after the feature F has been removed, then the resultant embedding $G(x)$ will tend to be similar to that of y.

We name embedding X the *source domain* (e.g., NYC merchants) and embedding Y the *target domain* (e.g., LA merchants). A discriminator $D : \mathsf{R}^d \to [0; 1]$ is trained to distinguish between elements randomly sampled from $G(X) = \{G(x_1); :::; G(x_p)\}$ (value closer to 0) and $Y = \{y_1; : : : ; y_q\}$ (values closer to 1). Generator G is trained to prevent the discriminator from making accurate predictions. As a result, this is a two-player game, where the discriminator aims at maximizing its ability to identify the origin of an embedding, and G aims at preventing the discriminator from doing so by making the distribution of $G(X)$ and Y as similar as possible on F. After the network converges, G serves as a mapping function from the source domain (NYC) to the target domain (LA), implicitly removing the effect of F.

However, merely applying this mechanism to our setting allows for too much freedom in the space of generators G. It could be that the generated dataset $G(X)$ loses all structure associated with X, and thus no longer has a use in understanding the data in the source domain!

To balance this shortcoming, in addition to adopting the domain-adversarial approach, we also add a structure preservation component to measure the similarity of the generated embedding $G(X)$ and the original embedding X by adding cosine distance to the generator loss. Ideally we would want to preserve the pairwise cosine distance: so for all $x_1; x_2 \in X$ that $\cos(G(x_1); G(x_2)) \approx \cos(x_1; x_2)$. However, this would induce intractable p^2 terms, and the p terms conserving $\cos(G(x); x)$ are a suitable proxy (by triangle inequality).

Our proposed method UCAN (Unsupervised Constrained Alignment that is Non-Affine) uses the DAN structure shown in Figure 1, where the $D : \mathsf{R}^d \to [0; 1]$ and $G : \mathsf{R}^d \to \mathsf{R}^d$ are multilayer perceptrons. Ultimately, this is formulated so D and G play the two-player minmax game with value function $V(G; D)$:

$$\min_G \max_D V(D; G) = \mathsf{E}_{y \sim p_Y(y)}[\log(D(y))] \quad (1)$$

$$+ \mathsf{E}_{x \sim p_X(x)}[\log(1 - D(G(x)))] \quad (2)$$

$$+ \quad \cdot \mathsf{E}_{x \sim p_X(x)}[1 - \cos(x; G(x))] \quad (3)$$

Since the discriminator tries to separate the target embedding (e.g., LA merchants) and mapped source embedding (e.g., NYC merchants), the discriminator loss function L_D derived from Eq. (1) and Eq. (2), and can be written as:

$$L_D = -\frac{1}{q}\sum_{i=1}^{q} \log(D(y_i))) - \frac{1}{p}\sum_{i=1}^{p} \log(1 - D(G(x_i))):$$

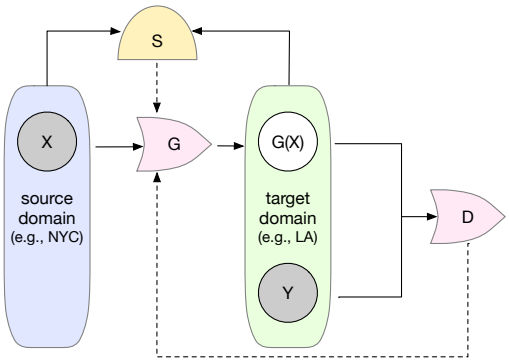

Fig. 1: DAN structure of UCAN. X represents an embedding in the source domain, $G(X)$ is the embedding mapped from the source domain to the target domain, and Y represents an embedding in the target domain. G is the generator, D is the discriminator, and S is the structure preservation component.

The generator G has two objectives: one is to fool the discriminator (e.g., so D cannot distinguish NYC merchants from LA ones), which makes the discriminator believe that the mapped embeddings are from the target distribution; the other is the structure preservation (S), which is to make the mapped embedding and the original embedding as similar as possible (e.g., so for an NYC merchant x, its mapped representation $G(x)$ still retains properties of x). We use cosine similarity as the similarity measure, as it is the standard similarity optimized in the creation of embeddings. We use a loss function L_G with two terms coming separately from Eq. (3) and Eq. (2) which is a proxy for $G(X)$ and Y having a good alignment. It is written as:

$$L_G = - \quad \cdot \frac{1}{p}\sum_{i=1}^{p} \cos(G(x_i); x_i) - \frac{1}{p}\sum_{i=1}^{p} \log(D(G(x_i))):$$

V. Experiments

We run an extensive set of experiments to validate our methods. To visually demonstrate the idea of our algorithm, we first show the experiment on synthetic datasets. Then we further show the effectiveness on three real-world datasets: two public datasets and one industry dataset. We demonstrate two real-world applications for the industry dataset, where removing a location feature by our approach improves the merchant identification accuracy and cross-city restaurant recommendation performance. The proposed algorithm significantly outperforms the state-of-the-art unsupervised linear methods on all of the datasets.

We use linear SVM as the classifier to identify the features; it could be replaced by other classifiers. The model is trained on 80% of the data, and the one vs. all AUC is calculated from the remaining 20% of the data. We repeat the experiments 10 times to get the average results. For all the experiments, F1 is the feature we want to retain, and F2 is the feature we want to remove.

TABLE I: Results on Synthetic Dataset. We report the AUC for lightness feature (F1) and color feature (F2).

	Balanced Data		Imbalanced Data	
	F1_AUC	F2_AUC	F1_AUC	F2_AUC
Raw data	0.98	0.82	0.98	0.82
= 5	0.97	0.70	0.97	0.50
= 1	0.97	0.60	0.97	0.50
= 0:5	0.97	0.57	0.97	0.50
= 0:3	0.97	0.55	0.97	0.50
= 0:1	0.97	0.52	0.97	0.50
= 0	0.59	0.78	0.88	0.50

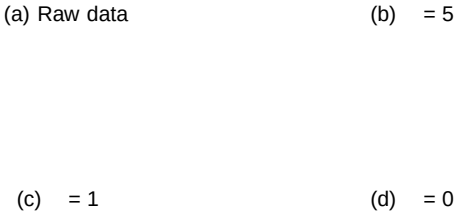

Fig. 2: Balanced synthetic data

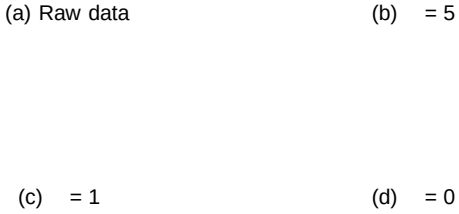

Fig. 3: Imbalanced synthetic data

A. Synthetic Datasets

Synthetic datasets have two dimensions; each corresponds to one type of binary feature. The x-axis varies with the color feature, and the y-axis varies with the lightness feature. The two binary features split the dataset into four parts, and each part is generated by a Gaussian distribution with the same covariance matrix. In this experiment, we aim to remove the color feature (F2) and retain the lightness feature (F1).

1) Balanced Dataset: Based on the color feature, we separate this dataset into two parts, each of which has 10,000 points. The original visualization of the dataset is shown in Figure 2(a). To remove the color feature, we map blue points to red points. By varying , which controls how much we preserve the structure of the original dataset, the results differ visually and analytically. From Table I, smaller (> 0) emphasizes the generator matching $G(X)$ to Y, making it more challenging for the discriminator to distinguish the target data and generated data, and thus the smaller AUC on color feature. Lightness feature is resilient to in 0:1 to 5, and retains high AUC (from 0:98 to 0:97; see Table I). We can also observe from Figure 2(b)(c) that smaller (> 0) can better align the source to target domain. However, when is too small (see = 0, so no constraint term), we give too much freedom to the generator without preserving the original data, and thus F2 is not well preserved in Figure 2(d), and the AUC of F2 is reduced dramatically. The optimal varies by datasets, but the algorithms are generally robust to this choice. Empirically, = 1 works well for most datasets (which balances the matching of $G(X)$ and Y with $G(X)$ and X), and unless speci ed, we use = 1 for the remaining experiments.

2) Imbalanced Dataset: The second synthetic dataset (see Figure 3) is generated through a similar process except for the number of each set of blue points reduces from 10,000 to 1,000; the color feature separates the whole dataset into two imbalanced parts. We still attempt to align the blue points to the red points. Color feature initially has an AUC of 0.88 and can be reduced to AUC = 0:5 with = 0:1 through = 5. When = 0, the Lightness feature's AUC remains 0.88.

We continue our experiments with the synthetic datasets which have two classes for the lightness feature (light and dark), but now three classes for the color feature (green, red and blue). The visualization of the multi-label dataset is the same as the binary dataset: the x-axis varies with Feature 1 (F1) which is the color feature, and the y-axis varies with Feature 2 (F2) which is the lightness feature. To reduce the color feature, we can x one color as the target domain and the other two colors as the source domains. For the balanced dataset, the choice of color as the target domain does not affect the results. However, for the imbalanced case where there are 1,000 green points, 10,000 red points and 10,000 blue points (illustrated in Figure 4(a)), the choice of target domain has a signi cant impact on the result. When we choose green as the target domain and map red and blue points to green points (Figure 4(b)), although red and blue points are aligned with green points, they are still not overlapped as desired. This is because blue or red points tend to align along their boundaries due to the structure-preserving components. Although smaller could solve the issue, the rule of thumb solution is choosing the domain with the most points as the target domain. For example, if we choose red points as the target domain and map green and blue points to the domain of the red points (Figure 4(c)), the three-color datasets are well overlapped. The color feature are signi cantly removed with AUC from 0.77 to 0:64 (Table III).

B. Real Datasets

We use three real-world datasets to demonstrate the effectiveness of the proposed algorithm. Since our goal is to retain the F1 feature and remove the F2 feature, if AUC score of F1 is not reduced more than 10%, the ratio score (AUC score of F1)/(AUC score of F2) is being calculated. A higher score

TABLE II: Results on airport embedding dataset. O represents the original result, C is the proposed algorithm (UCAN) result, M is the MUSE baseline and U is the UMWE baseline. We report the AUC scores for activity feature (F1) and location feature (F2).

	O_F1	O_F2	O_F1/O_F2	C_F1	C_F2	C_F1/C_F2	M_F1	M_F2	M_F1/M_F2	U_F1	U_F2	U_F1/U_F2
Brazil→ Europe	0.80	1	0.80	0.77	0.84	0.92	0.76	0.97	0.78	0.57	0.71	NA
Europe→ Brazil	0.80	1	0.80	0.79	0.86	0.92	0.82	0.95	0.86	0.57	0.80	NA
Brazil→ USA	0.85	1	0.85	0.85	0.61	1.39	0.83	0.98	0.85	0.5	1	NA
USA→ Brazil	0.85	1	0.85	0.88	0.91	0.97	0.87	0.93	0.94	0.52	0.51	NA
USA→ Europe	0.84	1	0.84	0.85	0.90	0.94	0.83	0.92	0.90	0.54	0.53	NA
Europe→ USA	0.84	1	0.84	0.83	0.62	1.34	0.80	0.99	0.81	0.49	0.97	NA

(a) Raw data with

(b) Green as target domain (c) Red as target domain

Fig. 4: Imbalanced multi-label synthetic dataset.

TABLE III: Results on imbalanced synthetic multi-label dataset. We report the AUC for lightness (F1) and color (F2).

	F1_AUC	F2_AUC
Raw data	0.98	0.81
Green_Target	0.98	0.77
Red_Target	0.98	0.64

means better performance. If AUC score of F1 is reduced more than 10%, the ratio score is set to NA. O represents the original result, C is the proposed algorithm result, M (MUSE [19]) and U (UMWE [20]) are the baseline results. For MUSE and UMWE, we use the default setting, where we run 5 million iterations with batch size 32, the number of refinements is 5 for the airport and merchant embedding dataset and 0 for the multi-language embedding dataset. For our algorithm, we set the number of iterations as 1 million, batch size as 32 for multi-language embedding dataset; the number of iterations as 30k and batch size as 512 for all the other experiments.

1) Airport Embedding Dataset: The air-traffic networks dataset includes three air-traffic networks [26]: Brazil, Europe and the USA: each node is an airport, and each edge shows the existence of commercial flights between airports. We consider two features: the level of activity (F1) with 4 classes, and country location (F2) with 3 classes. We apply the DEMO-net [27], which is a degree-specific graph neural network for node and graph classification, to train the model and get embeddings of all the nodes. Since the three networks are trained separately, if we view all the embeddings as one dataset, the location feature (F2) is definitely embedded. With the level of activity as the label used during the whole training process, the level of activity (F1) is also strongly represented in the embeddings. We use pairwise classification henceforth. From Table II, the original features are labeled as O_F1 and O_F2, for all pairs, the AUC for location feature is 1 and AUC for the activity feature is around 0.8. Our approach (labeled as C_F1 and C_F2) can reduce the weight of the location feature (F2) by about 9% 39% for different pairs with minimal degradation of activity (F1). The result of the baseline methods MUSE, UMWE are labeled as M_F1 and M_F2 & U_F1 and U_F2. Compared with our algorithm, MUSE and UMWE do not perform well. The F1/F2 score only marginally increases for MUSE and not at all for UMWE, whereas our algorithm's ratio score increases about 15% to 63%. We mark U_F1/U_F2 as NA since U_F1 is degraded so much that this measure is not useful.

2) Multi-language Embedding: For multi-language embedding, our goal is to remove the "language" feature, and retain meanings of the words from different languages. We use unsupervised word vectors that were trained using fastText [28][1]. These correspond to monolingual embeddings of dimension 300 trained on Wikipedia corpora. The languages focused on in our experiments are English (en), Spanish (es), French (fr), German (de), Russian (ru) and Italian (it). English is the source domain when translating English to other languages, while other languages are target domains. When we translate other languages into English, English is the target domain, and other languages are the source domains. We use the standard K-nearest neighbor (NN) and Cross-Lingual Similarity Scaling (CSLS) [19] as the evaluation approaches. We measure how many times one of the correct translations of a source word are retrieved, and report the precisions for $K = 1; 5; 10$ in Table IV. Our algorithm outperforms MUSE and UWME in most language pairs, except for MUSE on English to Spanish, and UWME on English to Russian. Our algorithm achieves the best result in the other pairs and is nearly best on the exceptions.

We also demonstrate the results on languages which are less similar to English, including Greek (el), Vietnamese (vi), Arabic (ar), Czech (cs) and Dutch (nl). As shown in Table V, our model can outperform MUSE and UMWE on all the language pairs. For language pair el-en, en-vi, vi-en, ar-en, MUSE cannot find a valid mapping. Thus both NN and CSLS results are close to 0. This is probably because el, vi and ar are from different language families as en. UMWE can overcome this issue by mapping them at the same time, but both NN and CSLS are not as good as our model. In conclusion, our model UCAN is very robust and can extend to any language

[1] https://github.com/facebookresearch/fastText

TABLE IV: Results on multi-Language embedding for word translation. NN and CSLS are two evaluation approaches. We use precision as metric for $K = 1, 5, 10$. C_ is the proposed algorithm (UCAN) result, and M is the MUSE baseline method and U is the UMWE baseline method.

		en-es	es-en	en-fr	fr-en	en-de	de-en	en-ru	ru-en	en-it	it-en
M_NN	K@1	69.07	64.40	53.96	61.6	61.02	51.40	24.20	32.00	56.45	59.33
	K@5	82.93	78.40	65.08	75.93	71.19	66.60	44.20	50.00	66.13	73.73
	K@10	86.87	81.8	68.25	80.60	72.88	71.93	51.40	56.20	69.35	81.13
U_NN	K@1	64.13	61.73	62.47	61.27	52.40	51.73	27.47	39.13	56.73	57.27
	K@5	76.20	75.80	75.73	75.67	74.27	67.33	52.33	57.67	71.67	71.27
	K@10	79.47	79.60	79.60	79.47	79.13	71.87	60.87	63.53	76.33	75.33
C_NN	K@1	68.93	72.00	69.60	69.00	60.27	61.80	25.73	46.60	61.33	63.07
	K@5	81.26	84.60	82.93	83.13	79.33	76.26	51.33	65.73	77.93	79.07
	K@10	85.00	88.40	85.80	87.20	83.27	80.80	59.80	70.26	82.87	82.53
M_CSLS	K@1	76.00	71.93	68.25	69.87	71.19	57.20	27.80	37.00	66.13	66.53
	K@5	86.86	83.20	79.37	82.47	79.66	72.87	49.93	57.67	79.03	79.20
	K@10	89.60	85.87	80.95	85.40	81.36	77.07	56.60	63.53	82.26	82.33
U_CSLS	K@1	70.07	68.67	69.00	68.93	59.47	58.80	32.33	45.80	63.53	65.13
	K@5	81.20	81.27	82.20	81.67	78.60	73.47	58.33	64.33	77.27	77.07
	K@10	84.60	84.20	84.87	84.93	83.20	76.93	66.27	69.93	82.00	80.33
C_CSLS	K@1	72.00	77.60	74.93	75.73	64.47	65.33	30.00	50.20	66.93	69.20
	K@5	84.46	87.60	86.40	87.47	81.27	78.60	57.20	70.20	81.80	82.73
	K@10	87.13	90.00	88.46	90.33	85.60	83.33	64.26	74.40	85.47	86.07

family, while MUSE is limited to work on languages similar to English.

3) Merchant Embedding Dataset: This embedding dataset is generated from a real-world transaction dataset from a well-known global payment company involving 70 million merchants and 260 million customers from December 1, 2017 to June 30, 2019. The merchant embedding is generated by Word2vec [1], [10], where each merchant is treated as a word and each customer as a document. The embedding is trained on all available US transaction data. For this experiment, we focus on the merchant embedding dataset in four areas: Los Angeles (LA), San Francisco (SF), Chicago (CHI), and Manhattan (MAN). We detect three prominent features in this dataset: location, frequency, and merchant category code (MCC), where the location is the feature to be removed, and the other features are retained. Similar to the airport dataset, we use pair-wise mapping to demonstrate the results. In Table VI, F1 is the MCC feature, and F2 is the location feature. Both our algorithm and baseline methods can reduce the location feature while retaining the MCC feature; our algorithm significantly outperforms MUSE and UMWE across all the location pairs. By observing the F1/F2 ratio score, our method clearly outperforms MUSE and UMWE.

Apart from MCC feature, we can also retain the frequency feature (F3) when removing the location feature. We show the results in Table VII. Both UCAN (our algorithm) and MUSE can significantly reduce the location feature while retaining the frequency feature; UCAN significantly outperforms MUSE across all the location pairs. F3/F2 is increased from 10% to 50% under the MUSE algorithm, while UCAN increases from 43% to 78%.

C. Downstream Task: On Merchant Data Set

1) False Merchant Identity Detection: In this section, we use a real-world application as a downstream task to evaluate the effectiveness of the embedding mapping algorithm for the merchant embedding dataset. Credit/debit card payment volume has proliferated in recent years with the rapid growth of small businesses and online shops. When processing these payment transactions, recognizing each merchant's real identity (i.e., merchant category) is vital to ensure the integrity of payment processing systems. For example, a high-risk merchant may pretend to be in a low-risk merchant category by reporting a fake merchant category to avoid higher processing fees associated with risky categories. Specific business type (i.e., gambling) is only permitted in some regions and territories. A merchant could report a false merchant category to avoid scrutiny from banks and regulators.

Accurate embeddings are an essential part of our merchant false identity detection [29]. The merchant category identification system monitors the transactions of each merchant and notifies the investigation team whenever the identified merchant mismatches with the merchant's self-reported category. As shown in the architecture design of our classification model (Figure 5), the two types of features we used for the classification model are each merchant's embedding (i.e., learned through Word2vec based on [10], [30]) and the time series capturing each merchant's transaction behavior. However, the major information captured by the Word2vec-based merchant embedding is the geological location of the merchant. Thus, attenuating the dominant geolocation component should enhance the classification performance from the embeddings.

To test the performance of different embedding mapping algorithms, we train our model with Los Angeles's merchants and test the model on San Francisco's merchants. This way, we can directly examine the difference between merchant embeddings contaminated with geological location information and merchant embeddings free of geological location information. The number of merchant categories in our dataset is 50. Table VIII shows result of classification experiment. We use conventional classification performance measurements like micro f1, macro f1, hit rate at 3, and hit rate at 5. As expected, attenuating the geological location information is crucial for learning a more accurate classifier. The raw

TABLE V: Results on multi-Language embedding for word translation. NN and CSLS are two evaluation approaches. We use precision as metric for $K = 1; 5; 10$. C_ is the proposed algorithm (UCAN) result, and M_ is the MUSE baseline method and U_ is the UMWE baseline method.

		en-el	el-en	en-vi	vi-en	en-ar	ar-en	en-cs	cs-en	en-nl	nl-en
M_NN	K@1	13.87	0.00	0.00	0.07	12.47	0.00	24.73	41.67	49.07	45.53
	K@5	31.13	0.00	0.13	0.07	26.33	0.07	42.00	58.33	67.80	61.87
	K@10	38.33	0.00	0.33	0.07	32.27	0.07	49.67	64.40	73.33	67.00
U_NN	K@1	18.73	26.27	3.27	1.87	14.60	22.49	26.87	31.13	43.07	33.67
	K@5	36.47	44.80	8.40	5.60	32.00	38.35	46.53	48.00	60.20	48.93
	K@10	43.27	50.53	10.73	8.53	38.87	44.51	53.20	53.07	66.07	54.13
C_NN	K@1	28:20	41:47	12:13	31:93	19:97	34:80	28:33	49:73	60:00	61:67
	K@5	46:60	59:93	23:27	45:33	41:45	51:74	50:27	67:73	76:40	75:60
	K@10	52:53	65:27	27:67	49:60	48:67	57:50	58:93	71:60	80:60	79:27
M_CSLS	K@1	22.67	0.00	0.00	0.00	15.00	0.00	31.07	47.53	59.67	55.33
	K@5	42.93	0.00	0.07	0.07	32.60	0.07	52.80	65.53	77.27	70.93
	K@10	50.80	0.00	0.07	0.07	39.00	0.07	60.07	70.67	82.73	75.00
U_CSLS	K@1	25.20	34.47	6.27	4.20	18.47	28.45	32.93	38.53	50.67	43.67
	K@5	44.00	54.33	13.07	10.93	37.20	46.92	53.67	55.67	67.60	57.93
	K@10	50.33	59.07	16.27	14.93	44.07	52.95	61.07	60.20	72.67	63.33
C_CSLS	K@1	32:40	45:80	21:80	38:33	23:93	36:81	34:13	53:60	67:47	68:67
	K@5	51:53	64:27	33:93	50:26	45:60	54:75	57:20	69:80	82:07	80:60
	K@10	57:87	69:47	38:33	54:86	52:28	59:71	64:27	74:47	85:53	83:20

TABLE VI: Results on Merchant Embedding Dataset. O_ represents the original result, C_ is the proposed algorithm (UCAN) result, M_ is the MUSE baseline and U_ is the UMWE baseline method. We report the AUC scores for MCC feature (F1) and location feature (F2).

	O_F1	O_F2	O_F1/O_F2	C_F1	C_F2	C_F1/C_F2	M_F1	M_F2	M_F1/M_F2	U_F1	U_F2	U_F1/U_F2
LA! SF	0.64	0.88	0.73	0.62	0.59	1:05	0.62	0.63	0.98	0.62	0.77	0.81
SF! LA	0.64	0.88	0.73	0.62	0.61	1:02	0.61	0.78	0.78	0.62	0.75	0.83
LA! CHI	0.58	0.95	0.61	0.57	0.57	1:00	0.58	0.71	0.82	0.57	0.79	0.72
CHI! LA	0.58	0.95	0.61	0.56	0.57	0:98	0.57	0.63	0.90	0.57	0.75	0.76
SF! CHI	0.65	0.93	0.70	0.63	0.59	1:07	0.61	0.79	0.77	0.64	0.82	0.78
CHI! SF	0.65	0.93	0.70	0.63	0.56	1:13	0.63	0.66	0.95	0.64	0.76	0.84
MAN! LA	0.61	0.90	0.68	0.59	0.54	1:09	0.59	0.82	0.72	0.60	0.75	0.80
LA! MAN	0.61	0.90	0.68	0.59	0.54	1:09	0.59	0.60	0.98	0.59	0.73	0.81
MAN! SF	0.66	0.88	0.75	0.64	0.53	1:21	0.64	0.62	1.03	0.64	0.71	0.90
SF! MAN	0.66	0.88	0.75	0.64	0.53	1:21	0.64	0.62	1.03	0.65	0.68	0.96
MAN! CHI	0.62	0.93	0.67	0.60	0.52	1:15	0.59	0.80	0.74	0.61	0.80	0.76
CHI! MAN	0.62	0.93	0.67	0.60	0.52	1:15	0.61	0.61	1.00	0.62	0.75	0.83

TABLE VII: Results on Merchant Embedding Dataset. O_ represents the original result, C_ is the proposed algorithm (UCAN) result, and M_ is the MUSE baseline. We report the AUC scores for frequency feature (F3) and location feature (F2).

	O_F3	O_F2	O_F3/O_F2	C_F3	C_F2	C_F3/C_F2	M_F3	M_F2	M_F3/M_F2	U_F3	U_F2	U_F3/U_F2
LA! SF	0.57	0.88	0.65	0.58	0.59	0:98	0.57	0.63	0.90	0.6	0.77	0.89
SF! LA	0.57	0.88	0.65	0.57	0.61	0:93	0.57	0.78	0.73	0.61	0.75	0.81
LA! CHI	0.59	0.95	0.62	0.59	0.57	1:04	0.59	0.71	0.83	0.59	0.79	0.75
CHI! LA	0.57	0.95	0.60	0.57	0.57	1:00	0.56	0.63	0.89	0.57	0.75	0.76
SF! CHI	0.57	0.93	0.61	0.57	0.59	0:97	0.57	0.79	0.72	0.62	0.82	0.76
CHI! SF	0.59	0.93	0.63	0.59	0.56	1:05	0.59	0.66	0.89	0.59	0.76	0.78
MAN! LA	0.57	0.9	0.63	0.56	0.54	1:04	0.57	0.82	0.70	0.59	0.75	0.79
LA! MAN	0.57	0.9	0.63	0.57	0.54	1:06	0.56	0.6	0.93	0.58	0.73	0.79
MAN! SF	0.58	0.88	0.66	0.59	0.53	1:11	0.58	0.62	0.94	0.6	0.71	0.85
SF! MAN	0.59	0.88	0.67	0.58	0.53	1:09	0.59	0.62	0.95	0.6	0.68	0.88
MAN! CHI	0.59	0.93	0.63	0.58	0.52	1:12	0.59	0.8	0.74	0.59	0.80	0.74
CHI! MAN	0.59	0.93	0.63	0.58	0.52	1:12	0.58	0.61	0.95	0.56	0.75	0.75

TABLE VIII: Performance of the merchant category classification model.

	Micro F1	Macro F1	Hit@3	Hit@5
Raw data	0.2884	0.2352	0.4453	0.5154
MUSE	0.3142	0.1836	0.4952	0.5931
UCAN (ours)	0.3338	0.2742	0.5592	0.6543

embedding performs the worst in almost all performance measurements compared to both MUSE and UCAN. MUSE in general performs better than the raw embedding, but the macro f1 is much worse compared to the raw embedding. Lastly, our proposed UCAN embedding mapping algorithm overall has the best performance across all performance measurements compared to the baseline method.

Since our goal is to detect merchants with false merchant categories instead of classification, we further evaluate the accuracy of the complete detection system. In our system, the particular detection rule we used is: if a merchant's self-reported merchant category is not within the k top-most likely merchant category, the merchant will be reported as a suspicious merchant where k is an adjustable threshold for our detection system. To perform the evaluation, we randomly select 10% of the test merchant and randomly change their

Fig. 5: Overall system design and architecture design for our merchant category classi er. MLP stand for multilayer perceptron and the 1D convolutional net uses a design similar to the temporal convolutional network.

Fig. 6: The performance of the merchant category identi cation system.

self-reported merchant category to emulate the process of one merchant faking its merchant identity. As we vary the detection threshold, we report the f1 score and the number of suspicious agged merchants. The experiment result is shown in Figure 6. Our attenuating method can generate better quality embeddings comparing to other methods. The resulting merchant category identi cation system can capture suspicious merchants more accurately and also report less suspicious merchants to the investigation team than the system using raw or MUSE processed embeddings.

2) Cross-City Restaurant Recommendations: Another application under merchant embeddings is to recommend restaurants to a customer across different cities based on the customer's historical transactions in the home city. We evaluate the effectiveness of removing the location feature using a real-world restaurant recommendation system [31]. For this evaluation, we choose the recommendation system's collaborative ltering model, as its performance is heavily dependent on the quality of the embedding. Figure 7 shows how different components of the recommendation system interact with each other during inference time. For a speci ed area (zip code), the recommendation model ranks the restaurants on how similar they are to the restaurants visited by a customer previously and uses the similarity score to rank the restaurants. For this experiment, we sample around 10K customers having restaurant transactions from May 2019 to Oct 2019 in SF and travel to LA from Nov 2019 to Dec 2019. Our goal is to recommend LA restaurants for the sampled customers. For historical transactions, we use customers' SF transactions from May 2019 to Oct 2019. Utilizing the recommendation model for a customer, we obtain the top 15 restaurants from the ranked list of LA restaurants and compute the score as a percentage of recommended restaurants in the real restaurant transactions made by the customer in LA from Nov 2019 to Dec 2019. For this test case, our original Word2Vec embeddings give a score of 60.36%. After removing the location feature between SF and LA, the score increases to 63.60%, while MUSE does not improve the score.

TABLE IX: Performance of the cross-city restaurant recommendations of UCAN with different .

	Score
Raw data	60.36
MUSE	60.33
= 5	63.51
= 2	64.11
= 1	63.60
= 0 :5	63.51

To test the effectiveness of UCAN under a large range of, we vary from 0.5 to 5. All the alphas can work consistently well and = 2 gives the best results. Using = 1 as the rule of thumb can generally give good results. Table IX shows that our algorithm outperforms the original Word2Vec embeddings on this task regardless of the value of (beyond our default = 1).

VI. Conclusion

We provide new and very general methods to measure and attenuate features from embeddings, and demonstrate effectiveness on four data sets and two novel downstream tasks. Our key contribution is UCAN, an alignment algorithm which using the DAN framework to learn an unrestricted mapping from data with one feature label to data with the other, but adding a simple cosine similarity constraint to retain the structure. We demonstrate that UCAN is a simple and effective method to re ne embeddings.

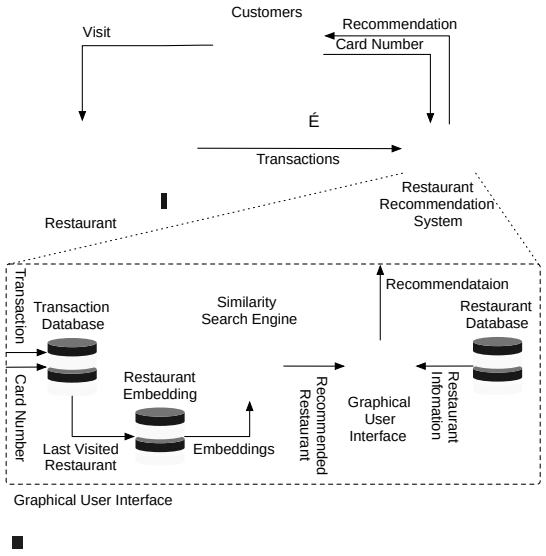

Fig. 7: Overall system design for our collaborative filtering-based restaurant recommendation system. An example of the graphical user interface in action is shown below.

ACKNOWLEDGMENT

We thank our support from NSF CCF-1350888, CNS-1514520, CNS-1564287, IIS-1816149, CCF-2115677, and from Visa Research.

REFERENCES

[1] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[2] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*. ACM, 2016, pp. 855–864.

[3] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *TKDE*, vol. 30, no. 9, pp. 1616–1637, 2018.

[4] G. Lowe, "Sift-the scale invariant feature transform," *Int. J*, vol. 2, pp. 91–110, 2004.

[5] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM-MM*, 2014, pp. 675–678.

[6] V. Tshitoyan, J. Dagdelen, L. Weston, A. Dunn, Z. Rong, O. Kononova, K. A. Persson, G. Ceder, and A. Jain, "Unsupervised word embeddings capture latent knowledge from materials science literature," *Nature*, vol. 571, no. 7763, p. 95, 2019.

[7] M. Grbovic and H. Cheng, "Real-time personalization using embeddings for search ranking at airbnb," in *KDD*. ACM, 2018, pp. 311–320.

[8] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee, "Billion-scale commodity embedding for e-commerce recommendation in alibaba," in *KDD*. ACM, 2018, pp. 839–848.

[9] X. Zhao, R. Louca, D. Hu, and L. Hong, "Learning item-interaction embeddings for user recommendations," *DAPA*, 2019.

[10] M. Du, R. Christensen, W. Zhang, and F. Li, "Pcard: Personalized restaurants recommendation from card payment transaction records," in *WWW*, 2019, pp. 2687–2693.

[11] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," in *NIPS*, 2016, pp. 4349–4357.

[12] A. Caliskan, J. J. Bryson, and A. Narayanan, "Semantics derived automatically from language corpora contain human-like biases," *Science*, vol. 356, no. 6334, pp. 183–186, 2017.

[13] S. Dev and J. Phillips, "Attenuating bias in word vectors," in *AISTATS*, 2019, pp. 879–887.

[14] C. Sweeney and M. Najafian, "Reducing sentiment polarity for demographic attributes in word embeddings using adversarial learning," in *FAT*, 2020, pp. 359–368.

[15] S. Dev, T. Li, J. Phillips, and V. Srikumar, "On measuring and mitigating biased inferences of word embeddings," *arXiv preprint arXiv:1908.09369*, 2019.

[16] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation." *CoRR*, vol. abs/1309.4168, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1309.html#MikolovLS13

[17] W. Ammar, G. Mulcaire, Y. Tsvetkov, G. Lample, C. Dyer, and N. A. Smith, "Massively multilingual word embeddings." *CoRR*, vol. abs/1602.01925, 2016. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1602.html#AmmarMTLDS16

[18] M. Faruqui and C. Dyer, "Improving vector space word representations using multilingual correlation," in *EACL*, 2014, pp. 462–471.

[19] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data." *CoRR*, vol. abs/1710.04087, 2017. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1710.html#abs-1710-04087

[20] X. Chen and C. Cardie, "Unsupervised multilingual word embeddings," in *EMNLP*, 2018, pp. 261–270.

[21] D. Madras, E. Creager, T. Pitassi, and R. Zemel, "Learning adversarially fair and transferable representations," pp. 3384–3393, 2018.

[22] H. Edwards and A. Storkey, "Censoring representations with an adversary," *ICLR*, 2016.

[23] D. Xu, S. Yuan, L. Zhang, and X. Wu, "Fairgan: Fairness-aware generative adversarial networks," in *Big Data*. IEEE, 2018, pp. 570–575.

[24] B. H. Zhang, B. Lemoine, and M. Mitchell, "Mitigating unwanted biases with adversarial learning," in *AAAI*, 2018, pp. 335–340.

[25] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *JMLR*, vol. 17, no. 1, pp. 2096–2030, 2016.

[26] D. R. Figueiredo, L. F. Ribeiro, and P. H. Saverese, "struc2vec: Learning node representations from structural identity," in *KDD*, 2017, pp. 13–17.

[27] J. Wu, J. He, and J. Xu, "Net: Degree-specific graph neural networks for node and graph classification," in *KDD*, 2019, pp. 406–415.

[28] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *TACL*, vol. 5, pp. 135–146, 2017.

[29] C.-C. M. Yeh, Z. Zhuang, Y. Zheng, L. Wang, J. Wang, and W. Zhang, "Merchant category identification using credit card transactions," *arXiv preprint arXiv:2011.02602*, 2020.

[30] C.-C. M. Yeh, D. Gelda, Z. Zhuang, Y. Zheng, L. Gou, and W. Zhang, "Towards a flexible embedding learning framework," *arXiv preprint arXiv:2009.10989*, 2020.

[31] M. Bendre, M. Das, F. Wang, and H. Yang, "GPR: Global Personalized Restaurant Recommender System Leveraging Billions of Financial Transactions," *WSDM*, 2021. [Online]. Available: https://doi.org/10.1145/3437963.3441709