Embedding for Anomaly Detection on Health Insurance Claims

Jiaqi Lu

Master of Science

Computer Science

McGill University

Montreal,Quebec

June 2019

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Masters of Science, Computer Science

DEDICATION

To my family and friends who keep encouraging me.

ACKNOWLEDGEMENTS

First I would like to express my gratitude to my academic supervisor Benjamin Fung for offering me countless suggestions and sharing his great knowledge about the field and the world to me. His guidance helps me finish this thesis.

Also, I would like to thank my colleagues during the industrial collaboration, David Lachapelle, Patrick Lemieux, Serge Illiesco, Roger Grondin, and Nadia Tahiri. I would like to thank them for explaining domain background to me, offering me numerous help in prototyping, and spending time with me sharing their experience and view about software development.

Special thanks to my family and friends, whoever in Montreal or not. Their love and support is always my motivation to move forward.

This research is supported by the Engage Grants (529904-18) from the Natural Sciences and Engineering Research Council of Canada (NSERC) with McGill Research Ethics Board's approval (146-0818).

ABSTRACT

Properly analyzing health insurance claims data could lead to significant business insights and benefits for health service providers and insurance companies. Yet, health insurance data is often high dimensional and contains complex interleave sequences of claims. Instead of conducting machine learning tasks directly on the raw data, a better approach is performing the tasks on high-quality embeddings of the raw data. Driven by the business need of our industrial partner, a Canadian technology company in the group insurance industry, in this thesis, we extract health insurance claims embeddings with neural networks in the context of anomaly detection. We propose and thoroughly examine six embedding components that are customized based on different possible assumptions made on the data. One of our proposed embedding components, *EC-ReStepRec*, significantly outperforms other candidates on two anomaly detection tasks. This is the first embedding study done on health insurance claims for anomaly detection.

ABRÉGÉ

Une analyse appropriée des données de réclamations d'assurance maladie pourrait permettre aux prestataires de services de santé et aux compagnies d'assurance de tirer des conclusions importantes sur les entreprises. Cependant, les données sur l'assurance maladie sont souvent de grande dimension et contiennent des séquences complexes de demandes entrelacées. Au lieu d'effectuer des tâches apprentissage automatique directement sur les donnes brutes, une meilleure approche consiste à effectuer les tâches sur des intégrations de haute qualité des données brutes. Soucieux de répondre aux besoins réalistes de notre partenaire industriel, une société canadienne de technologie du secteur des assurances collectives, nous extrayons dans cette thèse les intégrations de réclamations d'assurance maladie avec des réseaux de neurones dans le cadre de la détection des anomalies. Nous proposons et examinons de manière approfondie six composants d'intégration personnalisés en fonction des différentes hypothèses possibles basées sur les données. L'un des composants d'intégration proposés, *EC-ReStepRec*, surpasse de manière considérable les autres candidats sur deux tâches de détection d'anomalies. Ceci est la première étude d'intégration sur les demandes de règlement d'assurance maladie pour la détection d'anomalies.

CONTRIBUTION OF AUTHORS

The candidate and Prof. Benjamin Fung identified the research problem. The candidate formally defined the problem, designed the models, conducted the experiments, evaluated the experimental results, and wrote the entire thesis under the supervision of Prof. Benjamin Fung. Prof. Benjamin Fung proofread the final version of the thesis. The thesis is an extension of a submitted conference paper under review. Dr. William Cheung proofread the final version of the paper.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

Health insurance claims data are the bills between health service providers and insurance companies for the services obtained by a patient. A typical claiming process begins with a patient receiving health services from a provider. Next, the service provider submits a claim directly to an insurance company. The claim goes through validation checks followed by rules based on the patient's plan for pricing. Then, the insurance company pays the service provider [20]. Health insurance claims could be generally categorized into medical, pharmaceutical, and dental based on the services and service providers under request. As shown in Figure 1–1, each claim record generally contains information about the patient, the service provider, and the service. The exact attributes included in a claim depend on its category. For pharmaceutical claims, typical patient attributes include name, date of birth, address, etc. Typical service provider attributes include pharmacy code, pharmacist code, etc. Typical service attributes include medication code, quantity, date of service, etc.

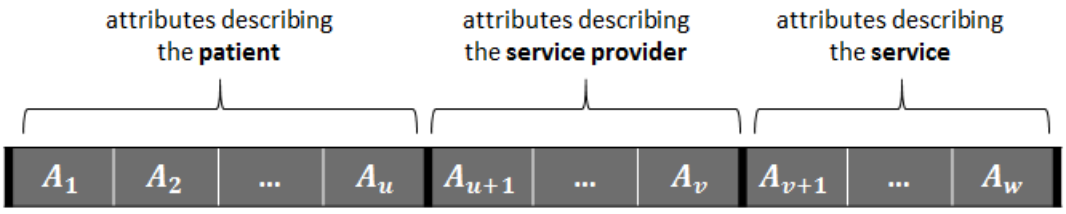


Figure 1–1: A claim record

Health insurance claims have been increasingly studied, resulting in many analytical insights that contribute to healthcare applications. Koh et al. [19] summarize the applications into four categories: evaluation of treatment effectiveness, healthcare management, customer relationship management, and anomaly detection.

Among the aforementioned applications, anomaly detection deserves special attention from insurance companies and governments. In the context of health insurance claims, there are three types of anomalies: frauds, abuses, and errors. *Frauds* indicate intentional acts of deception, misrepresentation, or concealment in order to get paid. *Abuses* indicate excessive or improper use of services that are inconsistent with acceptable business or medical practice and that result in unnecessary costs. *Errors* are unintentional mistakes made in processing claims. The boundaries between the three categories are not always clear. Frequent errors could suggest an abuse. Besides, intention is hard to be reflected in a claim itself. All three types of anomaly deserve special attention. Further manual examinations are required to determine the actions followed. The general goal is to accurately identify the anomalies.

In reality, however, it is hard to conduct analyses or perform machine learning tasks directly on health insurance claims data, which are often high dimensional and in the form of interleave sequences. Prevailing data analytical techniques are typically applied to datasets where the records are relatively small in dimension [21]. The same analytical dilemma also appears in other domains such as accounting and banking [25, 5, 36, 27, 9, 3, 37].

Traditionally, feature engineering plays an important role in addressing the issue of high dimensionality. Based on the knowledge for the target dataset, a relatively

small set of indicators would be selected as the input of the detection models. Recently, through the development of deep learning techniques, embedding has been widely studied as a solution to tackle the curse of dimensionality.

Driven by the business requirements of our industrial partner, Solution Segic Inc., a Canadian technology company that proposes solutions in the group insurance industry, we have been working on their health insurance claims data. We aim for embeddings that can effectively represent the health insurance claims data in low-dimensional space but still be descriptive, and thus the embeddings could be effectively applied in the analytical scenario of anomaly detection.

To obtain an effective embedding, we propose six embedding components for health insurance claims data. The embedding components that we present are carefully designed based on different assumptions made on the nature of the data. Each embedding component has a clear but very different learning preference. By training each embedding component as part of a deep learning model, respectively, we obtain the corresponding embeddings and evaluate them on two anomaly detection tasks of different granularity.

Our main contributions are summarized as follows:

- This is the first embedding study on health insurance claims for anomaly detection. With embedding, we effectively address the curse of dimensionality without heavily relying on domain knowledge for feature selection.
- We propose six embedding components to perform health insurance claims embedding. We thoroughly consider the possible assumptions on health insurance

claims. Based on different assumptions, we design the embedding components so that each has a distinct learning preference.

- We conduct extensive experiments on real-life health insurance claim data provided by our industrial partner. Results suggest that the embedding obtained by our proposed embedding component, *EC-ReStepRec*, is of outstanding quality and significantly outperforms other embeddings under comparison.

This thesis is organized as follows. Chapter 2 describes the works related to health insurance claims embedding. Chapter 3 is a brief overview of the machine learning concepts used in this thesis. Chapter 4 formally defines the research problem. Chapter 5 presents each proposed embedding component in detail. Chapter 6 shows the experiment on a real-life health insurance claims dataset. Chapter 7 is the evaluation of the embeddings by two anomaly detection tasks with visualization. Chapter 8 contains our interpretation of the results and lists future work directions. Chapter 9 concludes the thesis.

Chapter 2
Literature Review

2.1 Anomaly Detection

The development of information technology and the trend of digitization of insurance data boost the advancement of techniques for efficient and effective anomaly detection. Machine learning, as a group of intelligent methods that can learn from data and solve problems with minimal human intervention, has been widely studied in anomaly detection in various domain, including the healthcare sector.

Existing machine learning methods for anomaly detection in health insurance claims can be generally categorized into supervised and unsupervised learning methods.

Supervised learning methods

Supervised learning methods are good at capturing patterns within the data. With a supervised learning method, the model learns from a labeled dataset so that it can predict a label when a new sample comes in. Classical supervised machine learning methods, such as *support vector machine (SVM)* [18], *decision tree* [16], and *logistic regression* [22] have been employed for detecting anomalies. Lots of analysis software provide good support to these classical machine learning methods, which greatly decreases the technical difficulty in conducting experiments and thus the contribution of studies with classical methods mostly lies in their processing procedure and reasoning rather than modeling. Kirlidog et al. [18] use an *SVM* with

the linear kernel to predict the probability of anomaly and give a detailed analysis on the discovered anomalous records according to three criteria. Johnson et al. [16] propose a multi-stage method to classify with decision tree upon an aggregated risk measure. Liou et al. [22] employ *logistic regression*, neural network and *decision tree*, a.k.a. *classification tree* and their experiment results show that all three methods can achieve accuracy over 90%.

Artificial neural networks also have been increasingly used in detection problems in the healthcare sector because of their solid performance on diverse application problems [1, 22]. Ortega et al. [1] propose a system consisting of multiple committees of neural networks, where each committee is a sub-model corresponding to one of the entities involved in the problem.

Unsupervised learning methods

However, to gather the labeled data required by supervised learning methods is not easy, especially in the context of anomaly detection. Labeling requires domain knowledge and experiences and thus usually it is guided by professional examiners. In addition, labeling is expensive considering the human resource involved and the time cost. Therefore, unsupervised learning methods, which does not require labels, attract the attention of researchers.

Typical unsupervised learning methods include clustering [23, 17], outlier detection methods [33] and association rule based methods [31]. Liu et al. [23] and Joudaki et al. [17] both employ clustering techniques. Joudaki et al. [17] cluster using indicators which are carefully created based on logical inference about suspicious behaviors. Suspicious groups are identified according to the clustering result. Liu et

al. [23] especially point out that geo-location information is potentially an important indicator of fraudulent behavior. Thornton et al. [33] mark suspicious claims by using multiple analysis techniques and outlier detection methods. The idea behind is that different techniques and analyses can reveal different types of anomaly. Shan et al. [31] achieve the same goal by mining positive and negative association rules. After domain experts verify those rules, rule-breaking claims would be regarded as risky and suspicious.

Hybrid methods

Unsupervised learning methods do not require labels and thus they are in lack of guidance comparing with supervised learning methods. Besides, even for methods in the same categories, every learning method has its own advantages and disadvantages. In order to take advantages from multiple learning algorithms, hybrid methods are proposed. Here we select a few of them and explain the ideas behind.

Shin et al. [32] first develop a scheme to compute the composite degree of anomaly (CDA) score of a service provider. The CDA score is a weighted sum of the degree of anomaly (DA) score for each selected indicator. The weights are suggested to compute in a supervised fashion, involving six statistical techniques including *correlation analysis*, *logistic regression*, and *discriminant analysis*, etc. Providers are arranged into groups based on the CDA score and then the grouped providers would be used to build a *decision tree* classifier, which classifies providers into groups with different CDA level. Providers classified into to top-scored groups will be highly suspected as anomalies. This method is flexible and can be easily customized according to user's setting. The decision tree brings the method good interpretability

on the relationship between indicators and prediction result. The involvement of the proposed CDA scoring scheme improves the interpretability of the modeling process and to some extent reduce the model's dependence on the reliable labels indicating anomaly.

Ngufor et al. [26] propose unsupervised learning algorithms for labeling noisy data streams characterized by drifting concepts and suggest that these methods could be applied whenever there is a need for unsupervised labeling of data streams for the purpose of supervised learning. In another work [38], the authors illustrate an example system, which starts with the unsupervised learning algorithm to assign labels then goes to supervised rule-based models.

The aforementioned learning methods face the same challenge of high dimensionality in real-life data. Most of the existing works address this issue based on preliminary knowledge, for example, by manual selection [32], by computing metrics or aggregated features on the raw data and then using those advanced indicators in the detection model [33, 17, 16, 22, 2]. The knowledge required to figure out the appropriate indicators mostly comes from in-depth case studies and literature reviews or from the help of experienced domain experts. Considering the advancement of deep learning techniques, learning the latent features is becoming feasible and practical. This thesis explores embedding learning for a specific domain as an alternative to traditional feature engineering.

2.2 Embedding

Embedding has been increasingly studied in different domains, such as natural language processing [25], graph analysis [5], and network analysis [36]. Generally, there are two categories of embedding learning methods:

Mathematical-based methods

These methods are unsupervised and relate to matrix computation in closed form. The computation cost is relatively low, and they are not limited to any specific domain [34, 7]. Baldassini et al. [3] obtained client embeddings on current account transactions with a *marginalized stacked denoising autoencoder* (*mSDA*) [7]. We experimentally compare our embeddings with the embedding obtained by *mSDA* on health insurance claims.

Learning-based methods

Learning-based methods dominate the state-of-the-art embedding studies. One of our baseline methods, *autoencoder*, is one of the popular methods. In a typical *autoencoder*, an encoder maps the input into an embedding, a decoder reconstructs the embedding back to the original input, and the whole model is trained to reduce the reconstruction loss. Schreyer et al. [30] introduced a few deep *autoencoders* for anomaly detection on accounting data. We implement and employ their models on health insurance claims and compare with the obtained embeddings in experiments. Alternatively, an embedding component is trained as part of a large model for a domain-specific task in a supervised way. Optimizing algorithms such as *stochastic gradient descent* (*SGD*) would be involved in these methods in order to learn the parameters [6, 25, 5, 36, 27, 9, 3, 37].

Word embedding models, as one of the most well-studied branches of embedding learning, have been adapted to health insurance claims for embeddings of medical concepts, including diseases, medicines, and procedures, and have been proven to be able to capture medical semantic relatedness [10, 8]. Yet, no related work has been done on health insurance claims embedding in the context of anomaly detection. This thesis fills this gap.

Chapter 3
Technical Background

In this chapter, we will introduce a few machine learning concepts used in this work, so that the readers can better understand the contents in the following chapters.

3.1 Classical Machine Learning Classifiers

Essentially, a classifier can be formulated as a mapping function F, given a feature space X and a finite set of discrete classes Y. There are two types of learning approaches to build a classifier.

- **Discriminative learning**

  Discriminative learning methods estimate the mapping function F directly. They could be further divided into probabilistic methods and non-probabilistic methods, depending on whether F has a probabilistic meaning. Probabilistic methods learn $P(Y|X)$ and predict the class based on probability, while non-probabilistic methods learn the mapping from the feature space to the discrete classes directly.

- **Generative learning**

  Generative learning methods are always probabilistic. They estimate $P(X|Y)$ and $P(Y)$ respectively and then estimate $P(Y|X)$ with the Bayes rule, as shown in Formula 3.1.

$$
\begin{aligned}
P(Y = k|X = x) &= \frac{P(X = x|Y = k)P(Y = k)}{P(X = x)} \\
&= \frac{P(X = x|Y = k)P(Y = k)}{\sum_k P(X = x|Y = k)P(Y = k)}
\end{aligned}
\tag{3.1}
$$

For simplicity, in this chapter, we use $f_k(x)$ to represent the class-conditional density of X in class k, that is $P(X = x|Y = k)$. We use π_k to represent the prior probability of class k, $P(Y = k)$, where $\sum_k P(Y = k) = 1$.

Additionally, there are ensemble methods, which makes use of multiple base methods in order to achieve better predictive performance than could be achieved from any single constituent method. The base methods can be generative or discriminative learning methods.

3.1.1 Generative Learning Methods

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a generative learning method which has been well applied in data classification and dimensionality reduction. While the obvious disadvantage is that *LDA* can only learn linear boundaries, the advantages of *LDA* include:

- closed-form and thus relatively low computation cost.
- inherently support multi-class classification.
- no hyperparameter to tune.

LDA models $f_k(x)$, the class-conditional density of X in class k, as a multivariate gaussian distribution with mean μ_k and covariance matrix Σ_k [35]:

$$f_k(x) = \frac{1}{(2\pi)^{d/2}\,|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)} \tag{3.2}$$

where d is the dimension. *LDA* assumes that $\boldsymbol{\Sigma_k}$ is the same for every class so we will simply use $\boldsymbol{\Sigma}$ to denote the covariance matrices.

According to Formula 3.1, after taking the logarithm and absorbing the terms that do not rely on the class k, we can define the linear discriminant function as:

$$\delta_k(x) = x^T \boldsymbol{\Sigma}^{-1} \mu_k - \frac{1}{2} \mu_k^T \boldsymbol{\Sigma}^{-1} \mu_k + \log \pi_k \tag{3.3}$$

To compute δ_k, we need to estimate π_k, μ_k, and $\boldsymbol{\Sigma}$. Suppose a set of training data of size N is available. $D = \langle (x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N) \rangle$. Then δ_k, π_k, and μ_k can be approximated as shown below.

$$\hat{\pi_k} = \frac{N_k}{N} \tag{3.4}$$

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{y^i = k} x^i \tag{3.5}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{(N-K)} \sum_k \sum_{y^i = k} \left(x^i - \hat{\mu}_k\right) \left(x^i - \hat{\mu}_k\right)^T \tag{3.6}$$

where K is the number of classes, N_k is the number of observations of class k, and $(x^i,\, y^i)$ is the i^{th} sample in the training set.

With the estimates, we can make an inference by computing $\hat{y} = \text{argmax}_k\, \hat{\delta}_k(x)$, where $\hat{y}$ is the predicted class for x, a previously unseen instance from the feature space.

Quadratic Discriminant Analysis (QDA)

The theory behind *Quadratic Discriminant Analysis (QDA)* is almost the same as *LDA*, except that *QDA* does not assume a common convariance matrix for every class [35]. $\mathbf{\Sigma}_k$ needs to be computed separately as shown in Formula 3.7.

$$\hat{\mathbf{\Sigma}_k} = \frac{1}{(N_k - 1)} \sum_{y_i = k} (x_i - \hat{\mu}_k) (x_i - \hat{\mu}_k)^T \tag{3.7}$$

The different assumption allows *QDA* to learn quadratic boundaries and thus *QDA* is more flexible.

Naïve Bayes

Naïve bayes is another prevailing generative learning method. It is under the assumption of conditional independence between every pair of features given the value of the class [39]. Therefore, for class k,

$$f_k(x) = \prod_{j=1}^{d} f_{kj} (x_j) \tag{3.8}$$

where d is the number of features, a.k.a. dimension, x_j indicates the j^{th} feature, and f_{kj} indicates the class-conditional density of the j^{th} feature in class k.

In practice, *naïve bayes* can work very well for tasks such as document classification and spam filtering. Although the strong assumption is rarely true in reality, *naïve bayes* is preferred due to the following advantages:

- only require a small training set.
- relatively low computation cost.
- less sensitive to the curse of dimensionality.

There are different types of *naïve bayes* classifier based on different assumption made on the distribution of f_{kj}. Here we just introduce *bernoulli naïve bayes* and *gaussian naïve bayes*.

Bernoulli naïve bayes assumes that f_{kj} is given by a bernoulli distribution. Each feature is assumed to be binary. For feature x_j of class k,

$$f_{kj}(x) = \pi_{kj}x + (1 - \pi_{kj})(1 - x) \tag{3.9}$$

where π_{kj} represents $P(x_j = 1|y = k)$.

Gaussian naïve bayes assumes that f_{kj} is given by a gaussian distribution. For feature x_j of class k,

$$f_{kj}(x) = \frac{1}{\sqrt{2\pi\sigma_{kj}^2}} e^{-\frac{\left(x-\mu_{kj}\right)^2}{2\sigma_{kj}^2}} \tag{3.10}$$

where μ_{kj} is the mean and σ_{kj} is the variance.

According to Formula 3.1 and Formula 3.8, the objective function is defined as:

$$\delta_k(x) = \pi_k \prod_{j=1}^{d} f_{kj}(x_j) \tag{3.11}$$

To compute δ_k, the unknown variables are estimated with a training set D of size N. We estimate π_k as Formula 3.4. If it is for *bernoulli naïve bayes*, π_{kj} is estimated as shown in Formula 3.12.

$$\hat{\pi}_{kj} = \frac{N_{kj}}{N_k} \tag{3.12}$$

where N_k is the number of observations of class k, N_{kj} is the number of observations of class k with $x_j = 1$.

If it is for *gaussian naïve bayes*, we can approximate μ_{kj} and σ_{kj} with maximum likelihood estimation. Suppose that in the training set D, there are N_k class k training samples, $D_k = \langle (x^1, k), (x^2, k), \ldots, (x^{N_k}, k) \rangle$, then

$$\hat{\mu}_{kj} = \frac{1}{N_k} \sum_i x_j^i \tag{3.13}$$

$$\hat{\sigma}_{kj}^2 = \frac{1}{N_k} \sum_i \left(x_j^i - \hat{\mu}_{jk} \right)^2 \tag{3.14}$$

where x_j^i indicates the j^{th} feature of x^i.

3.1.2 Discriminative Learning Methods

Logistic Regression

Logistic regression is a discriminant learning method. It is probably the most widely used learning algorithm as it is simple and easy to implement.

Logistic regression models $P(Y = k|X)$ linearly with the logistic function as shown in Formula 3.15.

$$\sigma_k(x) = \frac{1}{1 + e^{-(w_k^T x + b_k)}} \tag{3.15}$$

To estimate w_k and b_k, we minimize the sum of cross-entropy loss for all the training samples, which leads to the objective function:

$$argmin_{w_k,b_k} - \left[\sum_{i=1}^{N} y^i \log\left(\sigma_k\left(x^i\right)\right) + \left(1-y^i\right)\log\left(1-\sigma_k\left(x^i\right)\right)\right] \tag{3.16}$$

With the estimates, an inference can be made by computing $\hat{y} = \text{argmax}_k\, \hat{\sigma_k}(x)$., where $\hat{y}$ is the predicted class for x, a previously unseen instance from the feature space.

Support Vector Machine (SVM)

Support Vector Machine (SVM) is also a typical discriminative learning method. The advantages of *SVM* include:

- memory efficient. Only a subset of training points is required as support vectors.
- versatile. The kernel functions can be customized.
- less sensitive to the curse of dimensionality.

In the meanwhile, *SVM* also has disadvantages including:

- relatively high computation cost.
- do not directly provide probability estimates.

The main idea behind *SVM* is to find a *hyperplain* that can separate different classes in the high-dimensional space [14]. And the margin around the *hyperplain* is expected to be as wide as possible. The idea could be formally framed as an optimization problem. Given a training set of size N, $D = \langle (x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N)\rangle$, where x^i is a d-dimensional feature vector, we have the problem as defined in Formula 3.17.

$$\begin{gathered} \min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi^i \\ \text{subject to } y^i \left(w^T \phi(x^i) + b \right) \geq 1 - \xi^i \\ \xi^i \geq 0 \\ \text{where } w \in R^d, \xi \in R^N \end{gathered} \tag{3.17}$$

ϕ maps the feature vectors to a high dimensional space. ξ^1, ξ^2,..., ξ^N are slack variables. C is a penalty parameter that controls the trade-off between slack penalty and margin.

Choosing the feature mapping function ϕ is equivalent to choosing the kernel for the algorithm. A kernel is defined as: $K\left(x^i, x^j\right) \equiv \phi\left(x^i\right)^T \phi\left(x^j\right)$. Typical kernels include:

- Linear kernel:

  $K\left(x^i, x^j\right) = \left(x^i\right)^T x^j$

- Radial basis function kernel (RBF kernel):

  $K\left(x^i, x^j\right) = \exp\left(-\gamma \left\| x^i - x^j \right\|^2\right), \gamma > 0$

The $\hat{w}$ and $\hat{b}$ returned after resolving Formula 3.17 would be used in the inference stage. For binary classification, we can make an inference by computing $\hat{y} = \text{sign}\left(\hat{w}^T \phi(x^i) + \hat{b}\right)$, where $\hat{y}$ is the predicted class for an unseen instance x^i. For multi-class classification, a common practice is to train multiple pairwise *SVMs* and select the class inferred by the majority of pairwise *SVMs* as the final predicted class.

Decision Tree

Decision tree is a discriminative learning method. The main advantage of *decision tree* is that it is easy to understand and interpret. Essentially, a *decision tree* is a set of simple decision rules [15]. However, the limitation is the expressivity. *Decision trees* cannot easily express relations such as XOR and parity. Additionally, in practice, it is reported that *decision tree* is inclined to overfit.

A *decision Tree* is constructed by performing recursive binary splitting. Given a training set $D = \langle (x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N) \rangle$, where x^i is a d-dimensional feature vector. $x^i = (x_1, x_2, \ldots, x_d)$, a *decision tree* can be built by following the steps below.

Step 1: Considering all the features and all the possible cutpoint value, select a feature x_j and a cutpoint value s which leads to the greatest decrease in minimum classification error rate. The selected x_j and s constitute a test, which corresponds to an internal node.

Step 2: The data space is split into subspaces $R_1(j, s) = \{x | x_j < s\}$ and $R_2(j, s) = \{x | x_j \geq s\}$.

Step 3: Recursively repeat Step 1 and Step 2 on each subspace, until all the training samples are in the same class. Then create a leaf node with that class label and quit.

The classification error rate is the percentage of the samples in a space that do not belong to the most common class. In addition to the classification error rate, it is also common to use entropy in practice.

The inference process is described as below. With $x = (x_1, x_2, \ldots, x_d)$,

Step 1: Run the test at each internal node.

Step 2: Go to the branch depending on the test outcome.

Step 3: Recursively repeat Step 1 and Step 2 until reaching a leaf node. The predicted class for x is the majority class of the training samples that fall in that leaf node. Quit.

K-nearest Neighbors (KNN)

K-nearest neighbors (KNN) assumes that samples that are closer in terms of the distance metric are more likely to belong to the same class. For an unseen instance x from the feature space, *KNN* looks for the K closest samples in the training set and predict the class for x as $\hat{y} = arg\max_k \sum_{i=1}^{K} \mathbb{I}(y^i = k)$. Therefore, to some extent, *KNN* does not actually learn or generalize anything but infer based on memory.

Distance metric plays a critical role in *KNN*. The performance mainly relies on how well the distance metric reflects the sample similarity. The most commonly used metric is euclidean distance.

Considering the nature of *KNN*, the advantages of *KNN* include:

- simple and easy to implement.
- no assumption made on data distribution.

And in the meanwhile, the main disadvantage is its high memory space requirement, as *KNN* stores all the training data for inference.

3.1.3 Ensemble Learning Methods

Random Forest

Random Forest is an ensemble learning method based on the idea of bagging. Basically, bagging means building a diverse set of base classifiers and combining the

prediction of the base classifiers to finalize the final prediction. *Random forest* is based on multiple *decision trees* with certain randomness [4].

The training procedure could be summarized as:

Step 1: Given the training set D, construct M bootstrap replicates. Every bootstrap replicate R contains the same number of samples by randomly selecting from D with replacement.

Step 2: Train *decision trees* T_i with a bootstrap replicate R_i. For each internal node, determining the test only with p randomly selected features.

Step 3: Repeat Step 2 until M *decision trees* are created.

As we briefly mentioned above, the final prediction of a *random forest* is based on the prediction of all the M *decision trees*. A common practice is to return the majority class.

Adaboost

Adaboost is an ensemble learning method based on the idea of boosting. The main idea behind boosting is to train multiple weak classifiers in order. The weight of the training samples would be adjusted according to the performance of the previous weak learner [11].

Below we summarize the training procedure of a variant of *adaboost* which supports multi-class classification, *stagewise additive modeling using a multi-class exponential loss function (SAMME)* [12].

Step 1: Given a training set of size N, $D = \langle (x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N) \rangle$, where x^i is a d-dimensional feature vector. $x^i = (x_1, x_2, \ldots, x_d)$. Initialize the weight distribution of each training sample as $w_i = \frac{1}{N}$.

Step 2: Train a weak classifier, h_t, with weight distribution w.

Step 3: Compute ϵ_t, the weighted error rate of h_t.

$$\epsilon_t = \frac{\sum_{i=1}^{N} w_i \cdot \mathbb{I}\left(y^i \neq h_t\left(x^i\right)\right)}{\sum_{i=1}^{N} w_i} \tag{3.18}$$

where $\mathbb{I}(c)$ is a conditional function which returns 1 when the condition c holds true and returns 0 otherwise.

Step 4: Compute α_t, the importance of h_t.

$$\alpha_t = \log \frac{1-\epsilon_t}{\epsilon_t} + \log(K-1) \tag{3.19}$$

where K is the number of classes.

Step 5: Adjust the weights for the training samples.

$$w_i = w_i \exp\left(\alpha_t \cdot \mathbb{I}\left(y^i \neq h_t\left(x^i\right)\right)\right) \tag{3.20}$$

Step 5: Normalize w.

Step 6: Repeat Step 2 to Step 5, until T weak classifiers are trained. Quit.

Adaboost finalizes the prediction based on the prediction of all the T weak learners weighted by importance α.

$$\arg\max_k \sum_{t=1}^{T} \alpha_t \cdot \mathbb{I}\left(h_t(x) = k\right) \tag{3.21}$$

3.2 Artificial Neural Networks

Artificial neural networks are computational models inspired by biological neural networks. They are discriminant models which have been widely used to approximate mapping functions, especially for complicated functions.

Typically, artificial neural networks are organized in layers. A typical artificial neural network is stacked by at least three layers, an input layer, a hidden layer, and an output layer. A layer consists of a number of interconnected neurons.

The neurons [29], as the small units in an artificial neural network, play an important role in learning. Every neuron could be considered as a small learning device. It takes in a vector $x = (x_1, x_2, \ldots, x_m)$ and output a scalar o while handling a set of weights $w = (w_1, w_2, \ldots, w_m)$ and a scalar bias b. As shown in Figure 3–1, the mechanism of a neuron can be described as:

$$o = activation\left(w^T x + b\right) \tag{3.22}$$

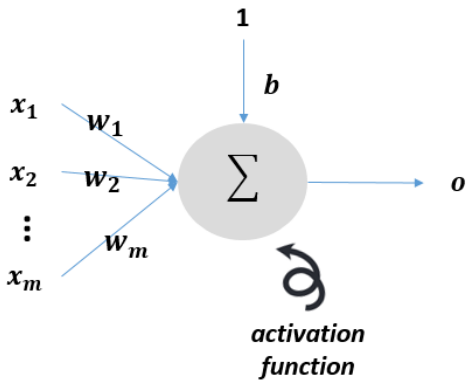


Figure 3–1: A neuron

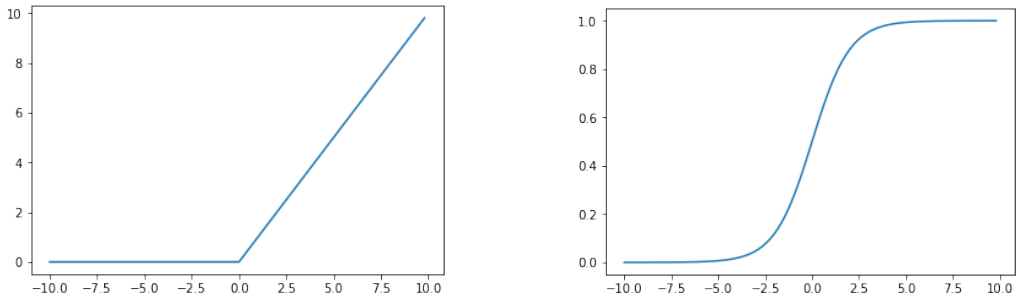
A neuron is always accompanied by an activation function. An activation function greatly impacts the behavior of a neuron and eventually impacts the learning performance of the model. Here we briefly introduce two activation functions, rectified linear unit (ReLU) and sigmoid, as they will be mentioned in the following chapters.

ReLU: $\gamma(x) = x^+ = \max(0, x)$

A non-linear activation function that only returns the positive part of the input signal. Empirically ReLU can alleviate the *vanishing gradient* problem which frequently happens in deep neural network training.

Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$

A non-linear activation function, which is also known as the logistic function. It maps an input signal into a value between 0 and 1.



(a) ReLU (b) Sigmoid

Figure 3–2: The activation functions

The most classic artificial neural network is the feed-forward fully connected neural network, which is also known as *multi-layer perceptron (MLP)*, as shown in Figure 3–3.

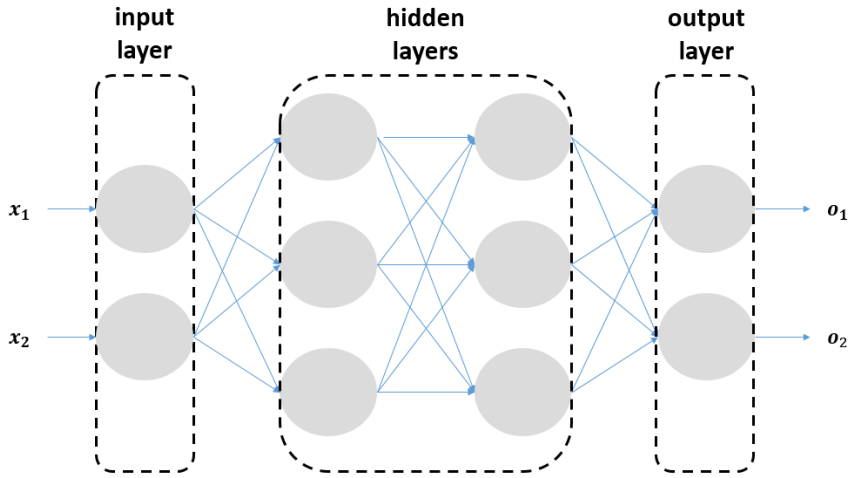


Figure 3–3: The feed-forward fully connected neural network

The intuition behind the stacked layers is that each layer processes and abstracts information that passed from the last layer with the adjusting weights and bias. Therefore, generally the more layers stacked, the more complex patterns could be captured. In addition to the number of stacked layers, another factor that greatly impacts the complexity of the learned pattern is the selection of layers.

3.2.1 The Fully Connected Layer

The fully connected layer, which is also known as the dense layer, is the most commonly used layer. All the neurons in the layer receive the input values and each neuron outputs a value.

A fully connected layer can be described as a mapping function:

$$o = activation\left(\mathbf{W}^T x + b\right) \tag{3.23}$$

x is a m-dimensional input vector, where $x = (x_1, x_2, \ldots, x_m)$. $o = (o_1, o_2, \ldots, o_p)$ is the p-dimensional output of the layer, where p is also denoted as dimension and

has to be set explicitly when defining the layer. p also corresponds to the number of neurons in the layer.

$\mathbf{W}$ is a $m \times p$ matrix of weights and b is the vector of the bias for the neurons. $\mathbf{W}$ and b is the core of the mapping function and they are the parameters to be learned during training.

We also visualize a fully connected layer in Figure 3–4 for better illustration.

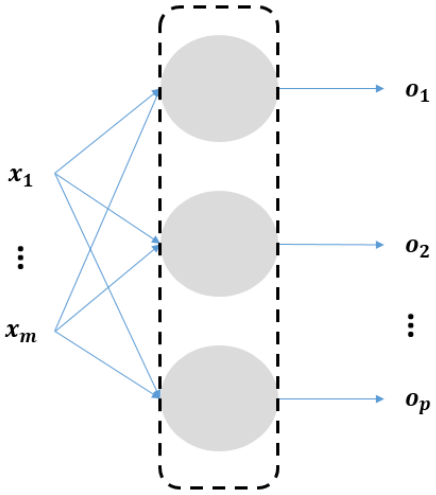


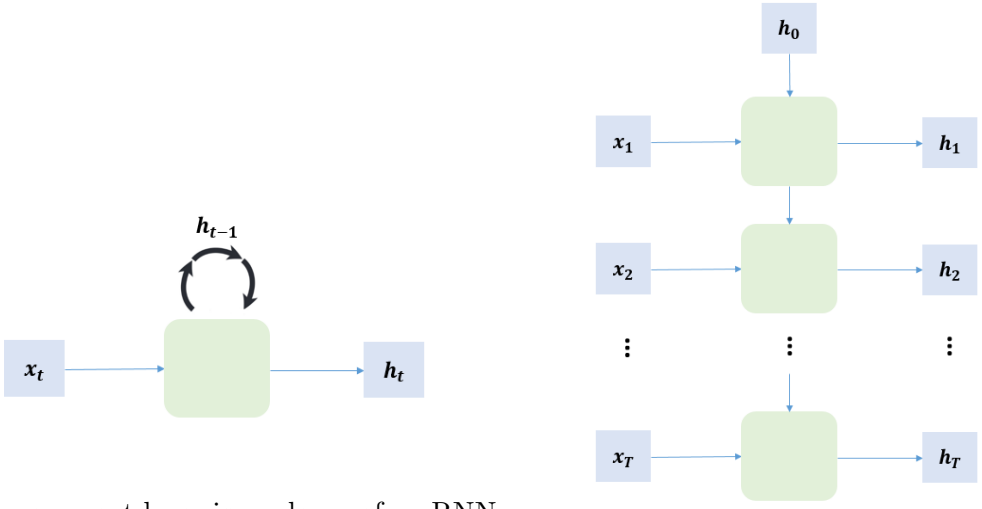
Figure 3–4: A fully connected layer (dense layer)

3.2.2 The Recurrent Layer

The recurrent layer is good at exploring sequential data and capturing temporal information. It is widely used in deep learning applications in text and speech.

A recurrent layer is made up of a recurrent neural network cell (RNN cell). An RNN cell keeps a hidden state h which gets updated over time. h is a p-dimensional vector that represents the useful information collected. At each time step, the cell

takes in an input x_t and the latest hidden state h_{t-1}, then it outputs the new hidden state h_t.



(a) An recurrent layer is made up of an RNN cell

(b) Unroll the recurrent layer over time

Figure 3–5: The recurrent layer

The mechanism of an *RNN cell* is shown in Figure 3–6. It can also be described as a mapping function:

$$h_t = activation\left(\mathbf{W}^T[x_t, h_{t-1}] + b\right) \tag{3.24}$$

t is the current time step between 1 and T, where T is the number of total time steps explicitly set when defining the layer. $x_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,m})$ is a m-dimensional input vector generated at time step t. $h_t = (h_{t,1}, h_{t,2}, \ldots, h_{t,p})$ is the p-dimensional hidden state at time step t. p also corresponds to the number of neurons needed in a cell, which should be explicitly set when defining the layer. $\mathbf{W}$

is a $(p+m) \times p$ matrix of weights and b is the vector of the bias for the neurons. It is worthy to note that $\mathbf{W}$ and b is shared between all time steps.

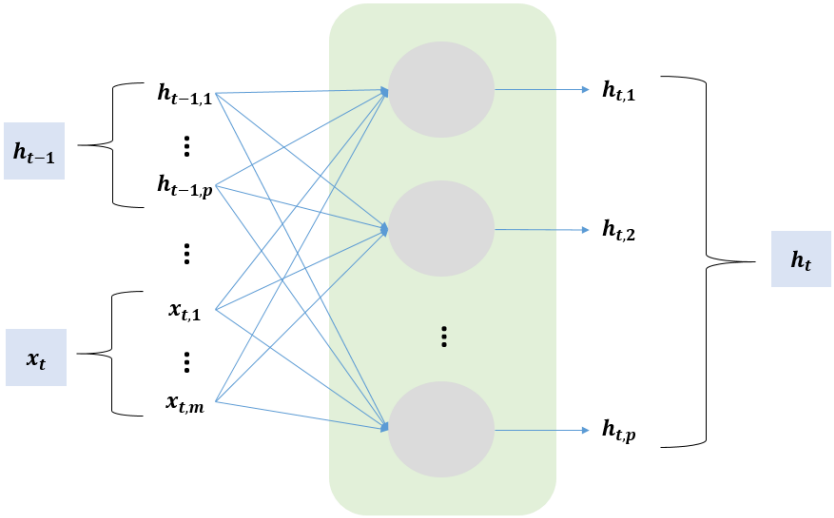
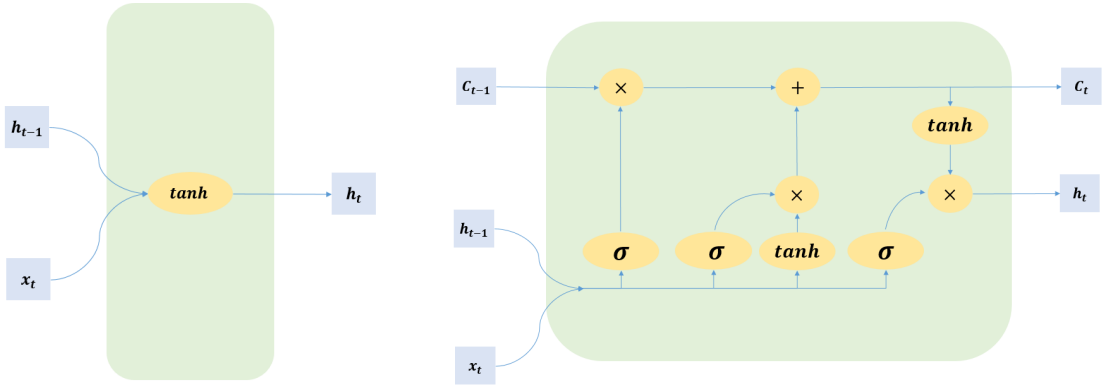


Figure 3–6: The mechanism of an *RNN cell*

3.2.3 The Long Short-Term Memory (LSTM) Layer

An LSTM layer is a variant of the vanilla recurrent layer [13]. Other than the hidden state h, an LSTM cell also manages the cell state C, which is updated additively. The cell state C can be interpreted as long-term memory. In fact, LSTM is proposed to address two main drawbacks of the vanilla recurrent neural network. One is to deal with long-term dependencies. Another is the gradient vanishing and exploding issue, resulting by the fact that the hidden state h is updated multiplicatively.



(a) A vanilla RNN cell (b) An LSTM cell [28]

Figure 3–7: Compare the complexity of a vanilla RNN cell and an LSTM cell

An LSTM cell encodes more complex computational logic inside. Figure 3–7a and Figure 3–7b visualize the operations within a vanilla RNN cell and an LSTM cell respectively. Behind the seemingly complex logic within an LSTM cell, it is three gates that control the information flow.

- Forget gate:

$$f_t = \sigma\left(\mathbf{W}_f{}^T[x_t, h_{t-1}] + b_f\right) \tag{3.25}$$

- Input gate:

$$i_t = \sigma\left(\mathbf{W}_i{}^T[x_t, h_{t-1}] + b_i\right) \tag{3.26}$$

- Output gate:

$$o_t = \sigma\left(\mathbf{W}_o{}^T[x_t, h_{t-1}] + b_o\right) \tag{3.27}$$

And then the cell outputs C_t and h_t accordingly.

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{3.28}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.29}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{3.30}$$

Chapter 4
Problem Description

Mostly, it is the characteristics of data that prohibit direct utilization and drive the involvement of embeddings. In the case of health insurance claims, the challenging characteristics are sequentiality and dimensionality.

Patient ID	A_2	...	A_u	A_{u+1}	...	A_v	Medication Code	Date of Service	...	A_w
1000	...	...	...	....	...	...	00010	20170201	...	...
1000	...	...	...	....	...	...	00010	20170225	...	...
1000	...	...	...	....	...	...	00010	20170307	...	...

Figure 4–1: An example sequence of claims

Sequentiality The claims could be processed into sequences by grouping. Figure 4–1 shows a sequence of claims identified by patient and medication code. By sorting the claims within a sequence by the date of service, the resulting sequence represents the medication history of a patient.

Generally, there exist two genres of relations in the claims:

- *Independent relation*: the relation among the attributes within the same claim.
- *Dependent relation*: the relation among the attributes across multiple claims in the same sequence.

Dependent relations, which only exist in sequences, are important in the context of health insurance claim anomaly detection. For example, they can represent

persistent behavioral patterns or ordered patterns that are likely to be suspicious but are generally hard to capture. Figure 4–2 shows an example.

Patient ID	Service	Date of Service
6010	**1001**	**20170201**
	0010	20170225
	0011	20170307
	1002	**20170311**
	1003	**20170320**

Patient ID	Service	Date of Service
6011	**1002**	**20170201**
	1001	20170225
	0011	20170307
	0010	20170311
	1003	**20170320**

Figure 4–2: An example of dependent relation: services 1001, 1002 and 1003 are usually requested in order. A patient with misordered service records is flagged suspicious.

Dimensionality The dimension for an encoded claim could be extremely large. The challenge amplifies if the data are in sequence, where multiple claims are assembled as one input. This is a challenge because the curse of dimensionality renders many traditional machine learning algorithms ineffective on many machine learning tasks.

In order to resolve those challenges, we resort to embeddings. An embedding is a relatively low-dimensional space into which high-dimensional vectors are transformed. Embeddings are helpful because they reduce the dimensionality of data while still effectively representing the relations within the original data in the mapping space. Good embeddings could well serve for various purposes. For example,

they could be the input for a specific target task or be directly visualized in order to intuitively illustrate the distribution of the original data.

Lastly, we formally define our research problem here. A claim is defined as $T=(x_1, x_2, \ldots, x_m)$, where x_i is an attribute or a feature. Given a set of sequences, $D=\langle S_1, S_2, \ldots, S_n \rangle$ where each sequence S_j is constituted by varying length of claims, $S_j=\{T_1, T_2, \ldots, T_k\}$, our problem is to find a mapping function $f : D \rightarrow R^d$ and thus every sequence S_j is mapped to a continuous vector of length d, $E=(e_1, e_2, \ldots, e_d)$, where m, n, k, and d are all positive integers. d should be significantly smaller than $m \times k$. The mapping should be of high quality so that the mapping space can effectively represent the original data.

Chapter 5
Model: Embedding Component Design

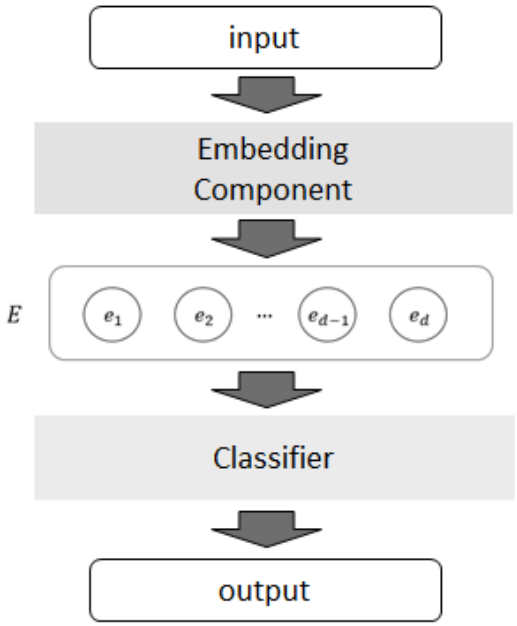


Figure 5–1: An overview of the architecture

Figure 5–1 provides an overview of the architecture, which consists of two components. The embedding component, which is the focus of this thesis, is the alternative of the traditional feature engineering process for learning an embedding. In this work, the classifier is a small fully-connected neural network responsible to classify the embedded sequences into classes depending on the user-defined customized task. In the training phase, both the embedding component and the classifiers are trained

as a whole. In the evaluation phase, only the embedding components are evaluated. The whole model takes a sequence of claims S_j as an input. T_i is the i^{th} claim in the sequence, denoted by T_i=$(x_1^i, x_2^i, \ldots, x_m^i)$.

We have explored, proposed, and evaluated different embedding components that are developed based on different assumptions that can be imposed on health insurance claim data. Each embedding component is customized for one type of assumption and thus is endowed with a specific learning preference, enabling the embedding component to explore certain relationships effectively. Here we discuss six architectures of embedding components.

5.1 *EC-Flatten*

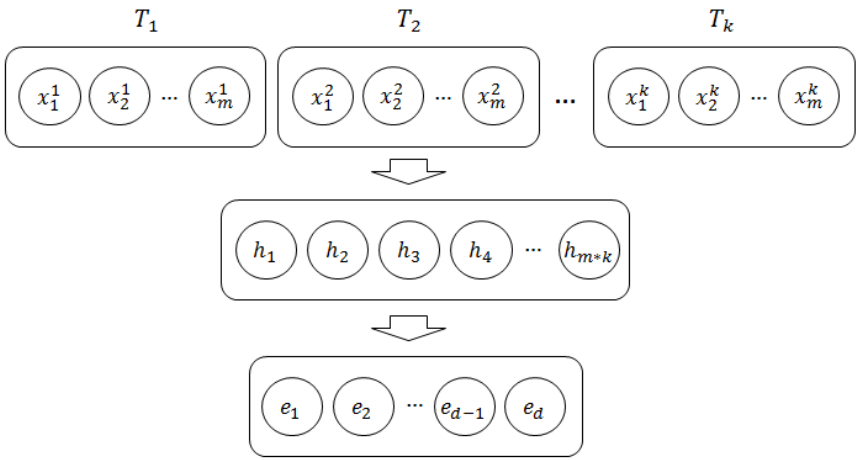


Figure 5–2: *EC-Flatten*

In *EC-Flatten*, there is no explicit assumption made in terms of the relationship between attributes, as we want to grant the model maximal flexibility. The claims in a sequence are concatenated into a one-dimensional vector. Therefore, attributes that come from the same claim and the attributes that come from different claims

are treated equally. Figure 5–2 illustrates the architecture of *EC-Flatten*, where $(h_1, h_2, \ldots, h_{m*k})$ is an intermediate output with $m \times k$ dimensions.

5.2 *EC-Recurrent*

In *EC-Recurrent* we assume that the inter-claim relationship in sequential context is important. Thus, each claim is fed into the model as one step. An abstraction persists and is updated from one step to the next. Finally, the output embedding is a global abstraction of the whole sequence. Figure 5–3 illustrates the architecture of *EC-Recurrent*, where $(h_{i,1}, h_{i,2}, \ldots, h_{i,p})$ is the p-dimensional global abstraction at step i. The final global abstraction, $(h_{k,1}, h_{k,2}, \ldots, h_{k,p})$ could be directly used as the outputted embedding so d equals to p. It also makes sense to further process $(h_{k,1}, h_{k,2}, \ldots, h_{k,p})$.

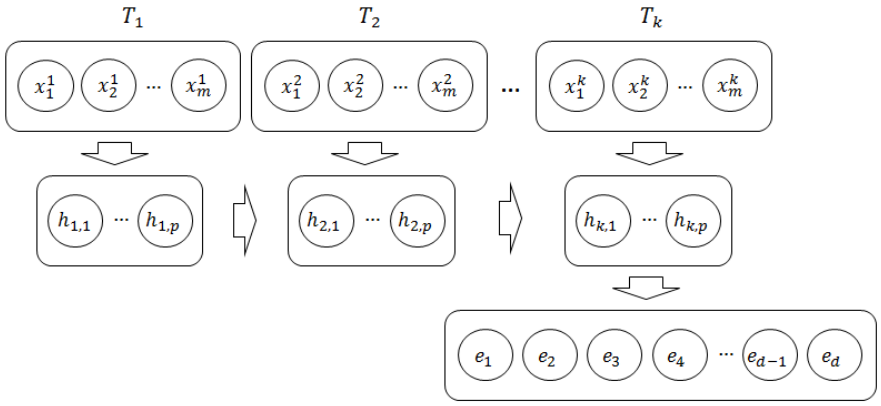


Figure 5–3: *EC-Recurrent*

5.3 *EC-Step*

In *EC-Step* we assume that claims in the same sequence do not closely rely on each other. Instead, the inter-attribute relationship within each single claim is

more important. As shown in Figure 5–4, each claim is fed into the model as one step. However, instead of allowing the information to evolve along the steps as *EC-Recurrent*, at each step the information exposed to the model is isolated. Abstraction is made step by step. $(h^1_{i,1}, h^1_{i,2}, \ldots, h^1_{i,p})$ is the p-dimensional abstraction on the input of step i. Next, the step-wise abstractions are concatenated into an intermediate output with $p \times k$ dimensions, which is $(h^2_1, h^2_2, \ldots, h^2_{p*k})$. The intermediate output is further mapped into a continuous space.

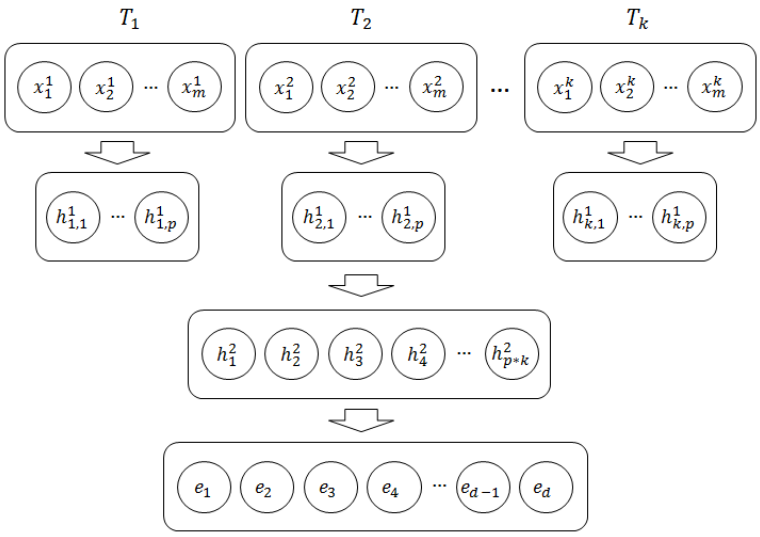


Figure 5–4: *EC-Step*

5.4 *EC-FlaRec*

EC-FlaRec is a hybrid architecture of *EC-Flatten* and *EC-Recurrent*. Therefore, while assuming the existence of inter-claim relationship, *EC-FlaRec* also benefits from certain flexibility. After concatenating the q-dimensional abstraction $(h^2_1, h^2_2, \ldots, h^2_q)$ produced by *EC-Flatten* with one intermediate layer, and the p-dimensional global abstraction $(h^1_{k,1}, h^1_{k,2}, \ldots, h^1_{k,p})$ produced by *EC-Recurrent*, the concatenated vector

is mapped to a continuous space. Figure 5–5 illustrates the architecture of *EC-FlaRec*.

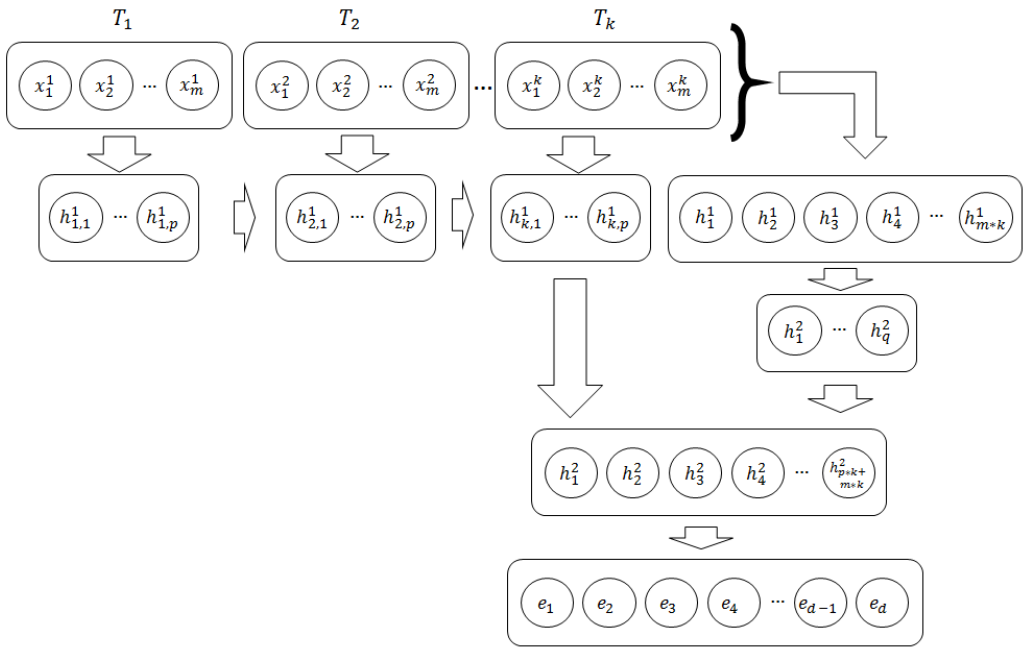


Figure 5–5: *EC-FlaRec*

5.5 *EC-StepRec*

In *EC-StepRec* we still assume that the inter-claim relationship in sequential context is critical. Yet, in addition to the global abstraction, the partial abstractions obtained during the intermediate steps are also informative. *EC-StepRec* is similar to *EC-Recurrent*, where a piece of information persists and is updated among the steps. Instead of outputting the last step abstraction only, here the abstractions obtained at each step are outputted, further abstracted, concatenated and mapped to a continuous space. Figure 5–6 illustrates the architecture of *EC-StepRec*. The first layer abstraction on the input of step i is represented as $(h_{i,1}^1, h_{i,2}^1, \ldots, h_{i,p}^1)$, where p is

the dimension. The first layer outputs are further abstracted into $(h^2_{i,1}, h^2_{i,2}, \ldots, h^2_{i,q})$, where q is the dimension. The second layer abstractions are concatenated into a $(q \times k)$-dimensional intermediate output $(h^3_1, h^3_2, \ldots, h^3_{q*k})$, which is then mapped to the embedding space.

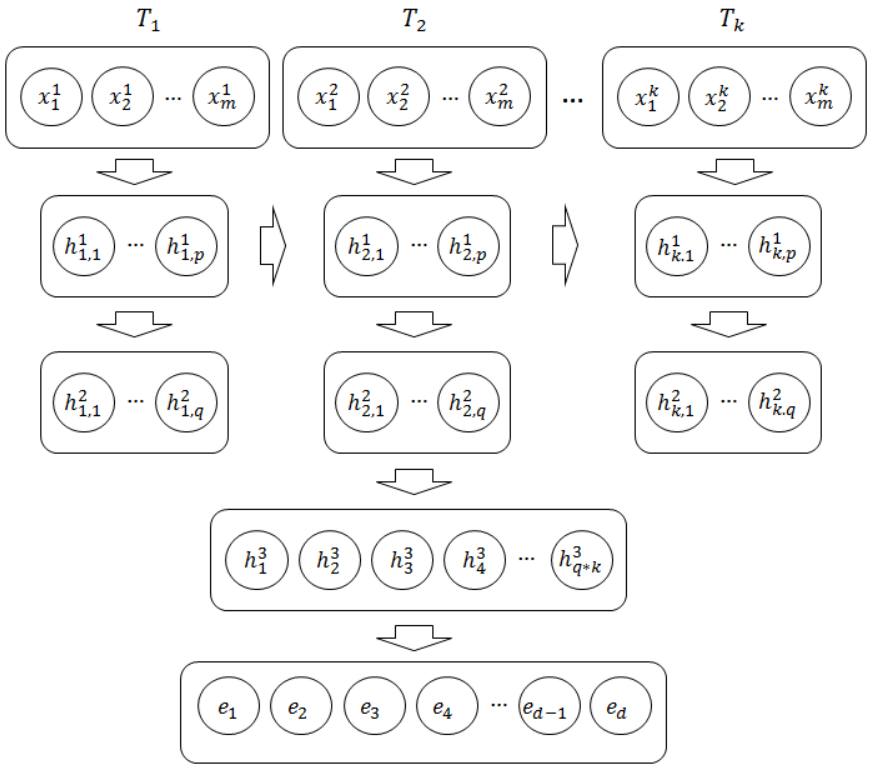


Figure 5–6: *EC-StepRec*

5.6 *EC-ReStepRec*

In *EC-ReStepRec* we assume that the sequence-wise inter-attribute relationship is important. By introducing a reshape trick, the input unit per step is no longer a claim, but the values for one attribute across all claims in sequence. Instead

of capturing the inter-claim relationship, here the intermediate layer captures the sequence-wise inter-attribute relationship. Next, similar to *EC-StepRec*, step-wise abstractions are outputted, further abstracted, concatenated, and mapped to a continuous space. Figure 5–7 illustrates the architecture of *EC-ReStepRec*. Due to the reshape trick, the input of step i is $(x_i^1, x_i^2, \ldots, x_i^k)$. The rest of the symbols used in Figure 5–7 are in line with Figure 5–6.

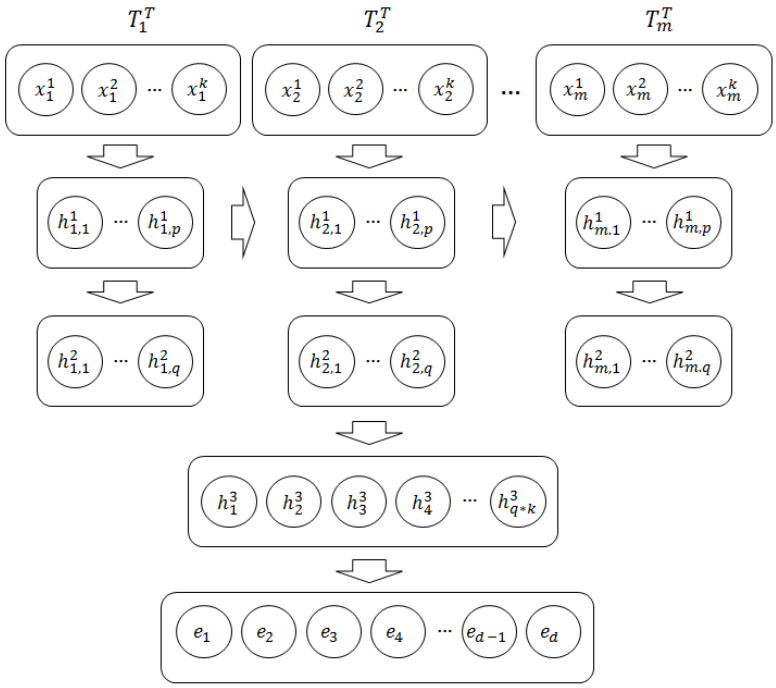


Figure 5–7: *EC-ReStepRec*

Chapter 6
Experiments

6.1 Data Preparation

6.1.1 The Pharmaceutical Claims Dataset

The experiments are performed on a pharmaceutical claims dataset provided by our industrial partner. A labeled dataset is assembled with the help of domain experts. The dataset consists of both anomalous and benign samples. Each sample is a sequence of claims of different length. The anomalous class can be further divided into two types of anomalies:

- *T1: exaggeration of claim amount*
- *T2: persistent early-refill behaviors on narcotics*

The labeling process simulates the traditional rule-based anomaly detection method. Domain experts explain the anomalous patterns. Based on the patterns we design validation rules accordingly. All claims go through the validation rules for T1 anomaly detection individually. All claims are grouped by medicine code and patient identifier first, then go through the validation rules for T2 anomaly detection as part of a sequence of claims. Upon the accepted pharmaceutical claims ranging from April 2015 through October 2018, we have 1,908 T1 anomalies, 7 T2 anomalies, and 8,760 benign cases. It is clear that the dataset is highly imbalanced. To avoid the impact of imbalance, 5,000 T1 anomalies and 2,500 T2 anomalies are simulated

by utilizing the validation rules reversely. Therefore, we finally obtain a raw dataset with 6,908 T1 anomalies, 2,507 T2 anomalies, and 8,760 benign cases.

A real-life health insurance claim database has abundant attributes. However, usable attributes are limited. Many attributes have to be excluded because of two main reasons.

- Excessive missing value: This happens frequently for non-mandatory fields of the claims.
- Unreliable filling: This happens frequently for fields whose format is ambiguous.

After consulting the domain experts, we only use mandatory and reliable attributes in our study. Additionally, we also apply necessary transformations which intuitively can help the model to learn faster and easier. The involved attributes and their descriptions are listed in Table 6–1.

Table 6–1: Attribute description

Attribute	**Description**
medication code	The identifier of the medication ordered.
quantity	The number of unit of the medication ordered.
age	The age of the patient when the claim is submitted.
claim amount	The total cost in dollar, including prescription drug cost, and pharmacist's professional fee.
transaction day of year	The day when the claim is submitted in the year. It is a number between 1 and 365 (366 if it is a leap year).
day of supply	The number of days the supply of dispensed medication will last.

6.1.2 Data Processing

$$norm(X) = \frac{X - min(X)}{max(X) - min(X)} \tag{6.1}$$

The raw data would go through a standard preprocessing procedure. The categorical attribute, medication code, is one-hot encoded into a 558-dimensional vector. Each numeric attribute is normalized to a range between 0 and 1, according to Formula (6.1). Finally, each claim is processed into a 563-dimensional vector. Since the claim sequences are of varying length, before a sample, whether an anomaly or a benign case, goes into the model, it will be either truncated or zero-padded into a sequence of 15. Note that we do not further process the sequence to take account

of classic considerations for sequences, such as time gaps, etc. Those considerations will be handled by the embedding components.

We randomly split the full dataset into a training set and a testing set. The training set accounts for 80% of the full dataset. 10% of the training set is reserved as a validation set.

6.2 Model Implementation

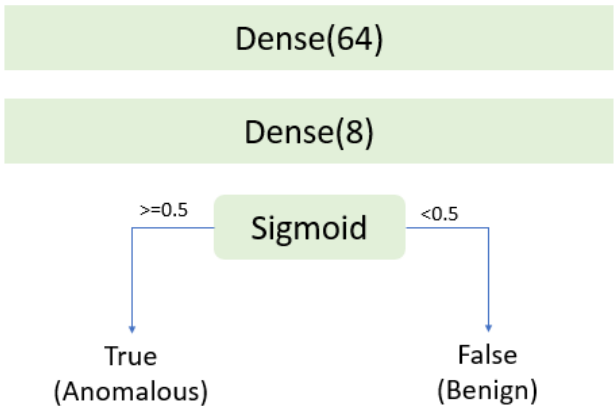


Figure 6–1: The default main classifier

To implement the framework described in Figure 5–1 we train a model, which consists of an embedding component and a main classifier, for a binary classification task that differentiates the anomalous class and the benign class. As shown in Figure 6–1, the default main classifier is a three-layer fully connected neural network, with 64 neurons, 8 neurons, and 1 sigmoid neuron in order. The output is a value between 0 and 1 which we interpret as the probability of being an anomaly.

Since our models are all implemented using TensorFlow in Python, here we illustrate our implementation of the proposed embedding components with TensorFlow

layers. To understand the most important layers, please refer to Chapter 3.2. It will not be difficult to find their equivalents in other libraries, such as PyTorch and Caffe.

Figure 6–2 illustrates the implementation of *EC-Flatten* by stacking the flatten layer and the dense layer. A flatten layer reshapes and converts an input into a one-dimensional data structure.

Figure 6–2: The implementation of *EC-Flatten*

Figure 6–3 illustrates the implementation of *EC-Recurrent* by using the LSTM layer.

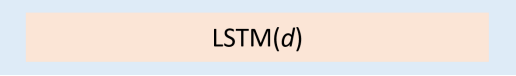


Figure 6–3: The implementation of *EC-Recurrent*

Figure 6–4 illustrates the implementation of *EC-Step*. A time-distributed dense layer is used to make step-wise abstraction in isolation. The flatten layer and the dense layer are followed.

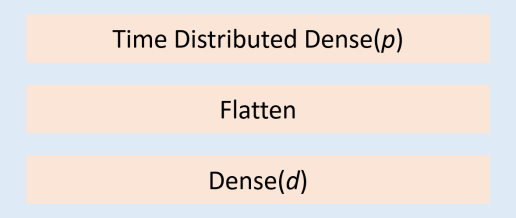


Figure 6–4: The implementation of *EC-Step*

Figure 6–5 illustrates the implementation of *EC-FlaRec*. The concatenate layer is used to merge the intermediate outputs from two previous sources.

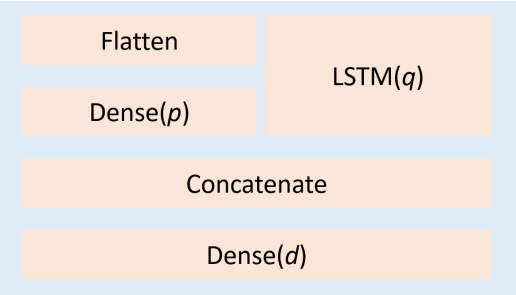


Figure 6–5: The implementation of *EC-FlaRec*

Figure 6–6 illustrates the implementation of *EC-StepRec*. It is worth noting that the LSTM layer has to be explicitly configured to return the output at every step. This is done by turning on the return_sequences parameter in TensorFlow.

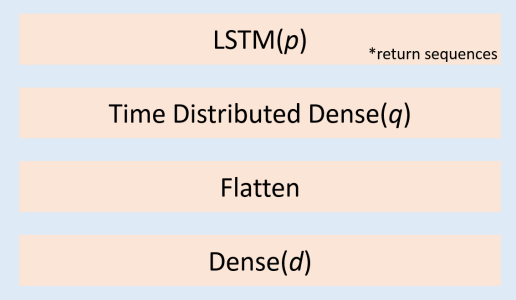


Figure 6–6: The implementation of *EC-StepRec*

Figure 6–7 illustrates the implementation of *EC-ReStepRec*. The transposition is done by the reshape layer on top. Also, the LSTM layer needs to be explicitly configured to return the output at every step.

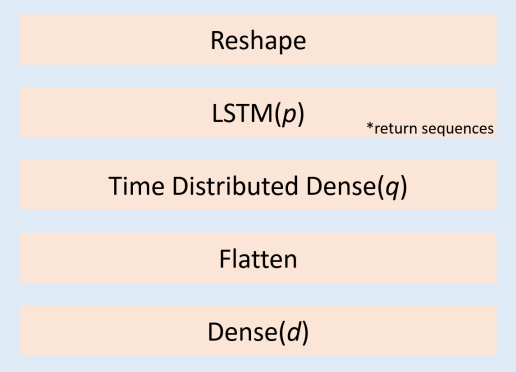


Figure 6–7: The implementation of *EC-ReStepRec*

The parameter settings for the embedding component are shown in Table 6–2. Each embedding component is regularized by dropout with 0.6 drop out rate and by batch normalization. The models are trained until convergence or reaching a running time limit.

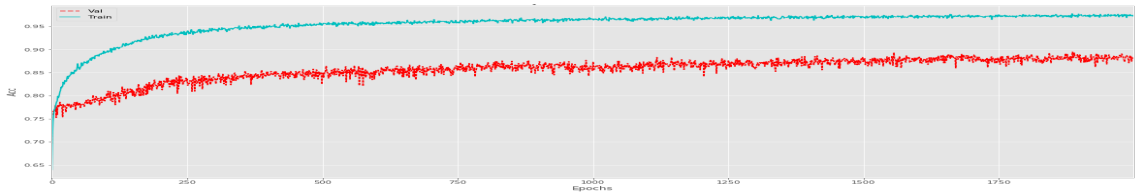
Table 6–2: Parameter settings for the embedding components

	m	k	p	q	d
EC-Flatten	563	15	/	/	128
EC-Recurrent			128	/	
EC-Step			16	/	
EC-FlaRec			128	128	
EC-StepRec			128	16	
EC-ReStepRec			128	1	

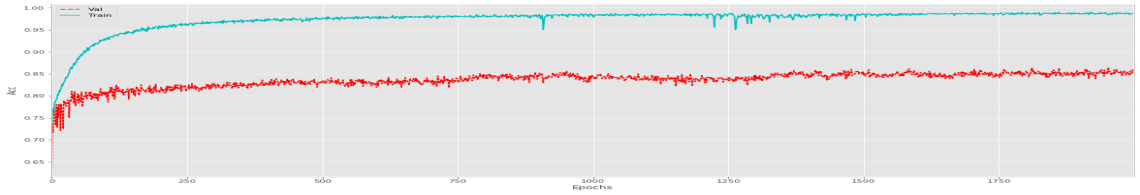
We monitor the learning progress by accuracy. Figure 6–8 shows the learning curves. The blue curve is the training accuracy and the red curve is the validation

accuracy. All the models learn well although the gap between training accuracy and validation accuracy is obvious for most models except *EC-ReStepRec*. The validation accuracy for *EC-ReStepRec* is unstable. The fluctuation might be caused by the size of the validation set. Limited by the dataset we own, we cannot reserve too many samples for validation, leading to the possibility that the distribution of the validation set and the distribution of the training set could be a bit different.

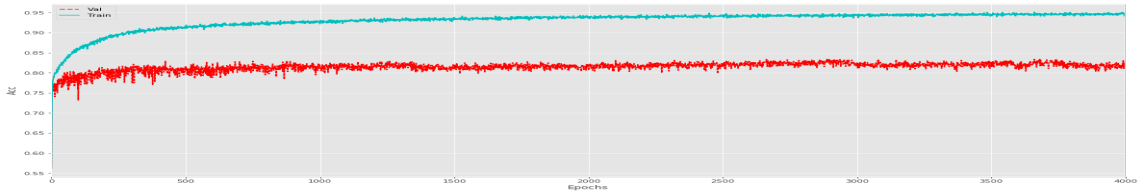
After training we collect the embedding components *EC-Flatten*, *EC-Recurrent*, *EC-Step*, *EC-FlaRec*, *EC-StepRec*, and *EC-ReStepRec* for further evaluation.



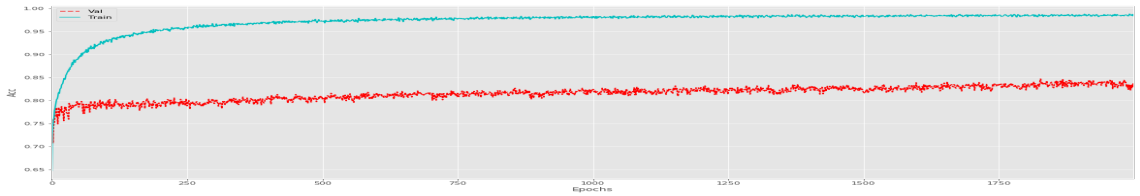
(a) *EC-Flatten*



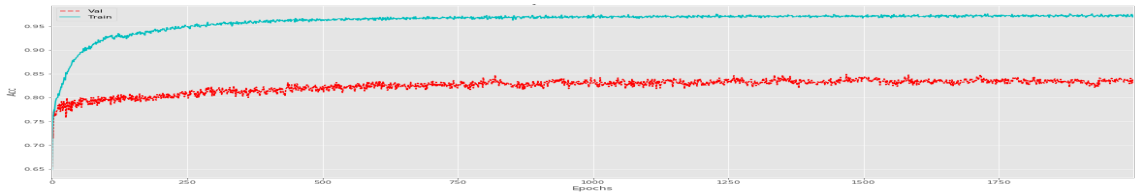
(b) *EC-Recurrent*



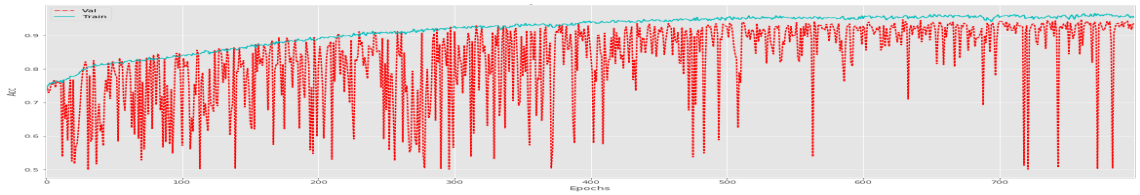
(c) *EC-Step*



(d) *EC-FlaRec*



(e) *EC-StepRec*



(f) *EC-ReStepRec*

Figure 6–8: Training models with the proposed embedding components

6.3 Baselines

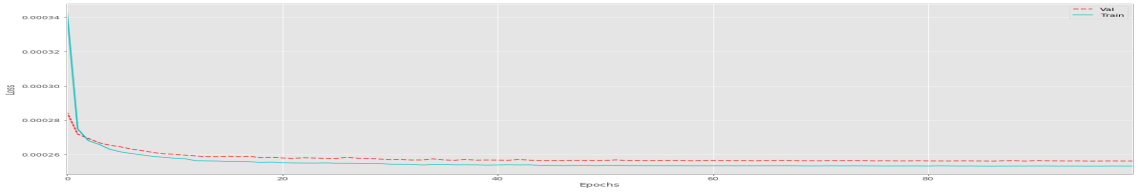
The six proposed embedding components are under comparison with each other. Additionally, as we mentioned previously, we also involve the following baseline methods, which have been employed in similar application scenarios. Note that as we defined in Chapter 4, the research problem is obtaining high-quality embedding. All the baselines are embeddings instead of anomaly detectors.

- Schreyer et al. [30] employ deep autoencoder to detect anomalies in accounting data. Here we compare with their best two deep autoencoders, *AE8* and *AE9*, as two baselines. Since our focus is the embeddings, we have to adapt the models and fix the dimension of the latent representation as 128. We implement *AE8* and *AE9* with TensorFlow. The parameter settings for the autoencoders are shown in Table 6–3.

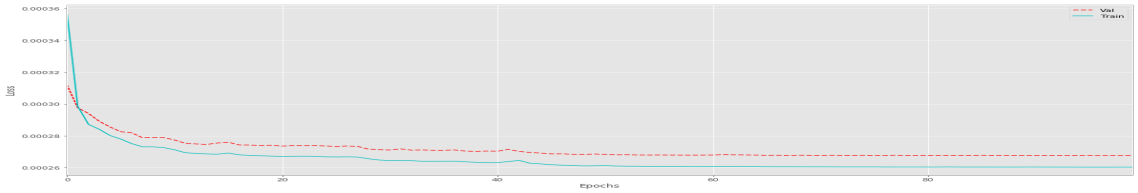
Table 6–3: Parameter settings for the embedding components

	dense layers and neurons
AE8	256-128-256
AE9	512-256-128-256-512

- Baldassini et al. [3] obtain client embeddings on current account transactions with a *marginalized stacked denoising autoencoder* (*mSDA*) [7]. Yet, *mSDA* does not reduce dimension. To be computational efficient and to also guarantee a fair comparison we first use *principal component analysis* (*PCA*) to compact the inputs into 128 dimensions and then stream the data into *mSDA*[34]. We use an open-sourced implementation of *mSDA* for the experiment.



(a) *AE8*



(b) *AE9*

Figure 6–9: Training autoencoders

We monitor the learning progress of autoencoders by reconstruction loss, which is evaluated by mean squared error (MSE). Figure 6–9 shows the learning curves. The blue curve is the training loss and the red curve is the validation loss. The loss curves dropping and converging, both *AE8* and *AE9* learn well.

Similarly, we collect the encoders of *AE8* and *AE9*, and the trained mapping of *mSDA*. Each of them maps the original dataset into a R^{128} embedding space.

Chapter 7
Evaluation

Following the convention in [5, 36], we evaluate the 9 embedding devices by two anomaly detection tasks. Also, we present their t-distributed Stochastic Neighbor Embedding (t-SNE) visualization as an intuitive evaluation [24].

7.1 Evaluation Tasks

We have two anomaly detection tasks of different granularity. Essentially, the tasks could be regarded as a binary classification task and a three-class classification task. For each embedding device, we use it to transform the original data into the R^{128} embedding space and then use the embeddings as the input of classical machine learning classifiers. We experiment with 10 classical classifiers, which we've introduced in Chapter 3.1.

- K-nearest neighbors (KNN) where $K = 5$
- Support vector machine with the linear kernel (L-SVM)
- Support vector machine with the radial basis function kernel (R-SVM)
- Decision tree (DT)
- Random forest (RF)
- Adaboost (Ada)
- Gaussian naïve bayes (NB)
- Logistic regression (LR)
- Linear discriminant analysis (LDA)

- Quadratic discriminant analysis (QDA)

The metric we use here is micro-average F1-score. It is the harmonic mean of micro-average precision $\overline{p}$ and micro-average recall $\overline{r}$.

$$\overline{p} = \frac{\sum_k TP_k}{\sum_k TP_k + \sum_k FP_k} \tag{7.1}$$

$$\overline{r} = \frac{\sum_k TP_k}{\sum_k TP_k + \sum_k FN_k} \tag{7.2}$$

$$F_1 = 2 \cdot \frac{\overline{p} \cdot \overline{r}}{\overline{p} + \overline{r}} \tag{7.3}$$

where TP_k, FP_k, FN_k indicate the number of true positives, false positives, and false negatives for class k.

We evaluate the quality of a specific embedding in two perspectives.

- **Superiority**: This is evaluated by the best micro-average F1-score achieved by any classifier with that embedding. The best micro-average F1-score for an embedding is in bold.
- **Robustness**: This is evaluated by the average micro-average F1-score achieved by all classifiers with that embedding.

7.1.1 Binary Classification Task

Table 7–1: Binary classification on test set

score(.)	*EC-Flatten*	*EC-Recurrent*	*EC-Step*	*EC-FlaRec*	*EC-StepRec*	*EC-ReStepRec*	*AE8*	*AE9*	*mSDA*
KNN(5)	82	82	**81**	**82**	**82**	92	**77**	76	**51**
L-SVM	**87**	82	81	82	82	92	76	**77**	50
R-SVM	79	82	78	82	82	**93**	76	75	50
DT	80	**83**	78	81	81	92	75	75	50
RF	75	82	77	82	82	92	74	75	50
Ada	84	82	80	82	82	92	76	76	50
NB	57	83	76	72	82	93	54	54	50
LR	85	82	81	82	82	92	76	76	51
LDA	86	82	81	82	82	92	76	76	51
QDA	51	83	78	69	82	93	56	55	49
Average	76.6	82.3	79.1	79.6	81.9	92.3	71.6	71.5	50.2

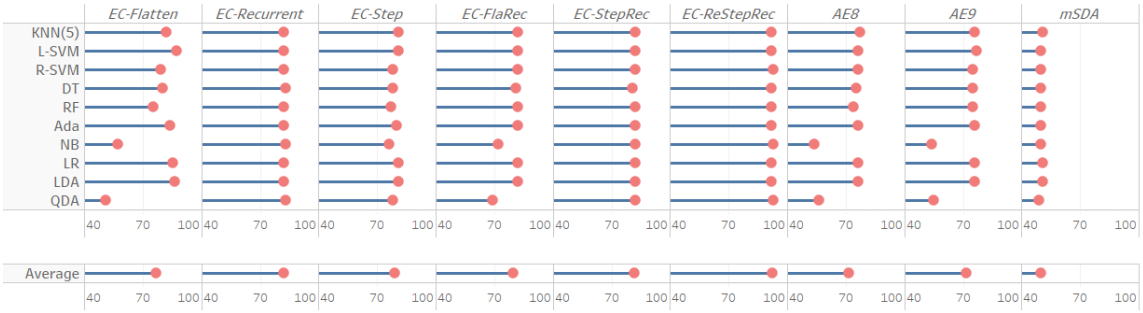


Figure 7–1: Lollipop plot visualizing the micro-average F1-scores on the binary classification task

The classifiers are trained to discriminate between the anomalous class and the benign class.

Table 7–1 reports the micro-average F1-scores on the testing set for each classifier and Figure 7–1 visualizes the same results with a lollipop plot for easier visual

comparison. Table 9–1 in Appendix shows the details of F1-scores per class. It is clear that the embedding obtained by *EC-ReStepRec* outperforms the others in both superiority and robustness, with the best micro-average F1-score as 0.93 and the best average micro-average F1-score as 0.923. We highlight the best values with square boxes.

7.1.2 Three-Class Classification Task

Table 7–2: Three-Class classification on test set

score(.)	*EC-Flatten*	*EC-Recurrent*	*EC-Step*	*EC-FlaRec*	*EC-StepRec*	*EC-ReStepRec*	*AE8*	*AE9*	*mSDA*
KNN(5)	79	79	**80**	**79**	68	**88**	**77**	76	40
L-SVM	**82**	78	80	79	**69**	79	77	**78**	40
R-SVM	66	70	72	78	69	79	76	74	40
DT	74	**81**	76	78	69	81	75	76	38
RF	62	71	64	70	69	82	73	72	40
Ada	78	81	78	79	69	80	62	58	40
NB	51	78	65	66	59	76	54	56	32
LR	81	77	79	79	69	79	76	77	40
LDA	82	80	79	78	69	79	76	76	40
QDA	53	79	63	59	69	71	58	58	**42**
Average	70.8	77.4	73.6	74.5	67.9	79.4	70.4	70.1	39.2

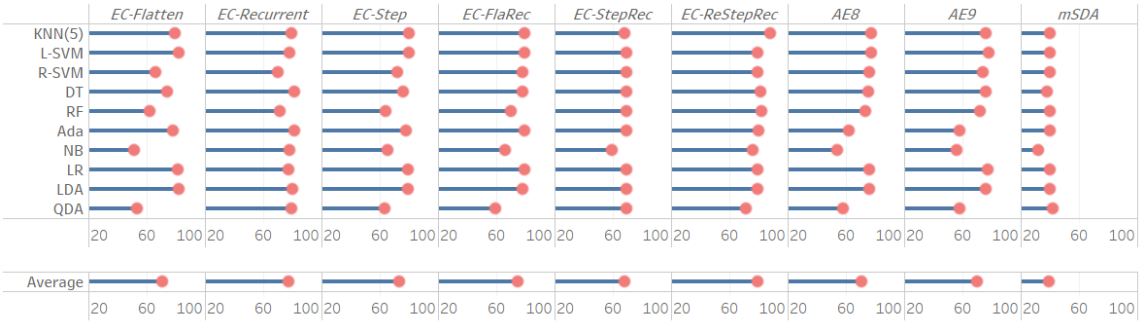


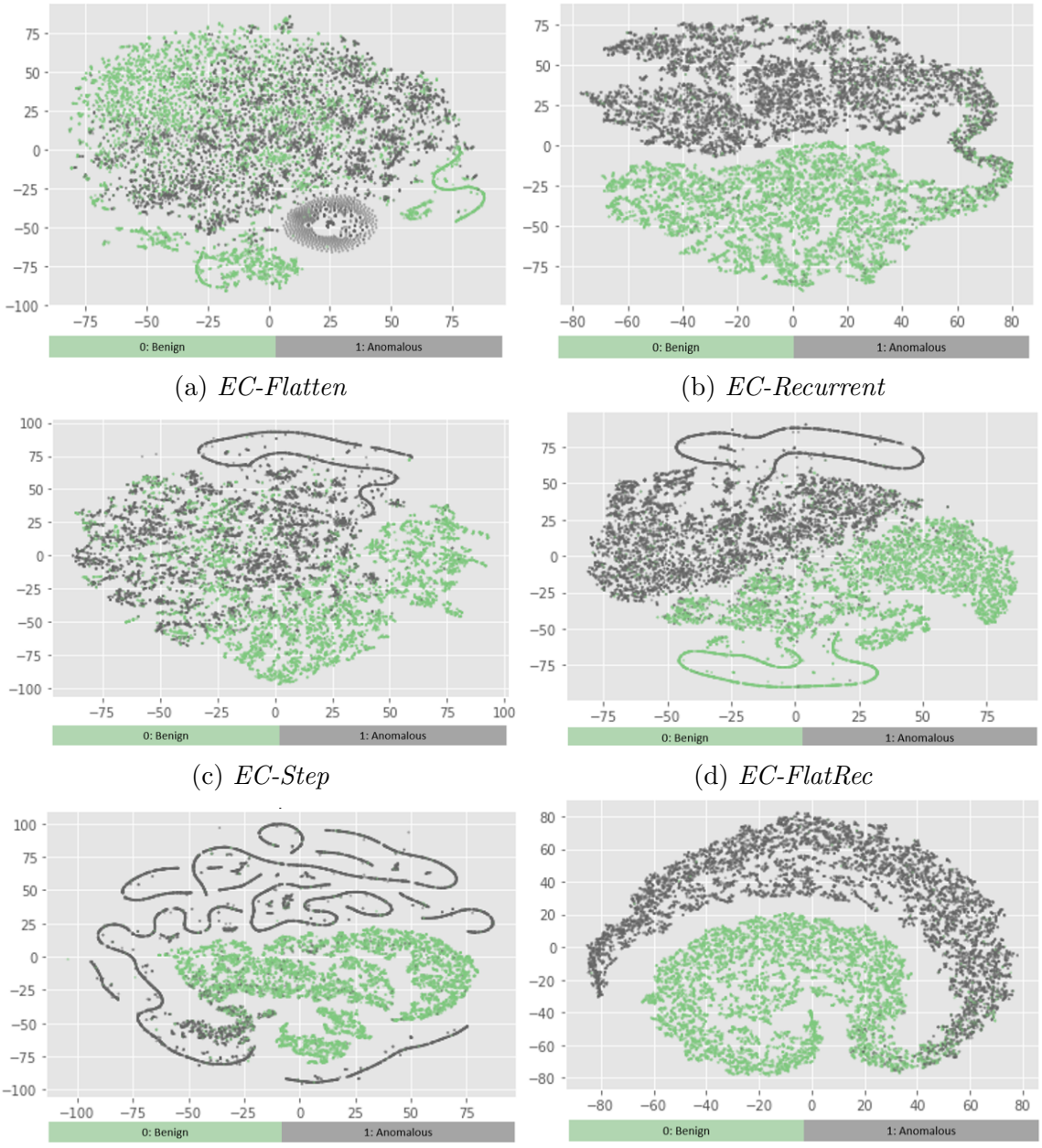
Figure 7–2: Lollipop plot visualizing the micro-average F1-scores on the three-class classification task

Those classifiers are also trained to discriminate between the T1 anomalous class, the T2 anomalous class, and the benign class. The way we evaluate embedding quality is the same as in Section 7.1.1. Table 7–2 summarizes the results and Figure 7–2 is a lollipop plot visualizing the results. A detailed table with F1-scores per class is provided in Table 9–2 in Appendix. Again, *EC-ReStepRec* achieves the best performance in terms of both superiority and robustness. The best micro-average F1-score is 0.88 and the average micro-average F1-score is 0.794.

7.2 t-SNE Visualization

t-SNE is a technique that is particularly well suited for the visualization of high-dimensional data by giving each data point a location in a two-dimensional map. It has been used in the evaluation of embedding as an intuitive evaluation method [5, 36]. Here we illustrate the t-SNE visualization on the embeddings with two levels of granularity.

7.2.1 Low Granularity Visualization



(a) *EC-Flatten* (b) *EC-Recurrent*

(c) *EC-Step* (d) *EC-FlatRec*

(e) *EC-StepRec* (f) *EC-ReStepRec*

Figure 7–3: t-SNE visualization of the *EC-Flatten*, *EC-Recurrent*, *EC-Step*, *EC-FlatRec*, *EC-StepRec*, and *EC-ReStepRec* embeddings (low granularity)

The *EC-Flatten* and *EC-Step* embeddings are the most chaotic. In Figure 7–3a, a large number of points of different classes overlap with each other in the central region. In Figure 7–3c, the points representing the embedding of a benign sample are visually dense in the region below the diagonal, but above the diagonal it is a mixture of points of two classes.

The *EC-FlatRec* and *EC-StepRec* embeddings are visually well separated. There is a clear boundary between the two classes. In Figure 7–3d, generally the points representing the embedding of a benign sample are located in the region below the diagonal and the points representing the embedding of an anomalous sample are above the diagonal. But a small number of samples are lost in the adverse region. In Figure 7–3e, the points representing the benign samples are concentrated in the central region, but the points representing the anomalous samples are dispersed elsewhere in curls, instead of in a grouped and compact region.

The *EC-Recurrent* and *EC-ReStepRec* embeddings are of the best quality. Points that belong to the same class are well grouped and the boundary between the groups is clear. In Figure 7–3b, the points of two classes have a narrow and roughly linear boundary. In Figure 7–3f, the boundary is wider but non-linear.

(a) *AE8* (b) *AE9*

Figure 7–4: t-SNE visualization of the *AE8* and *AE9* embeddings(low granularity)

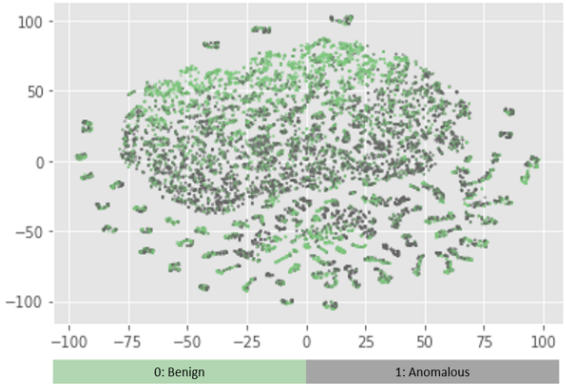
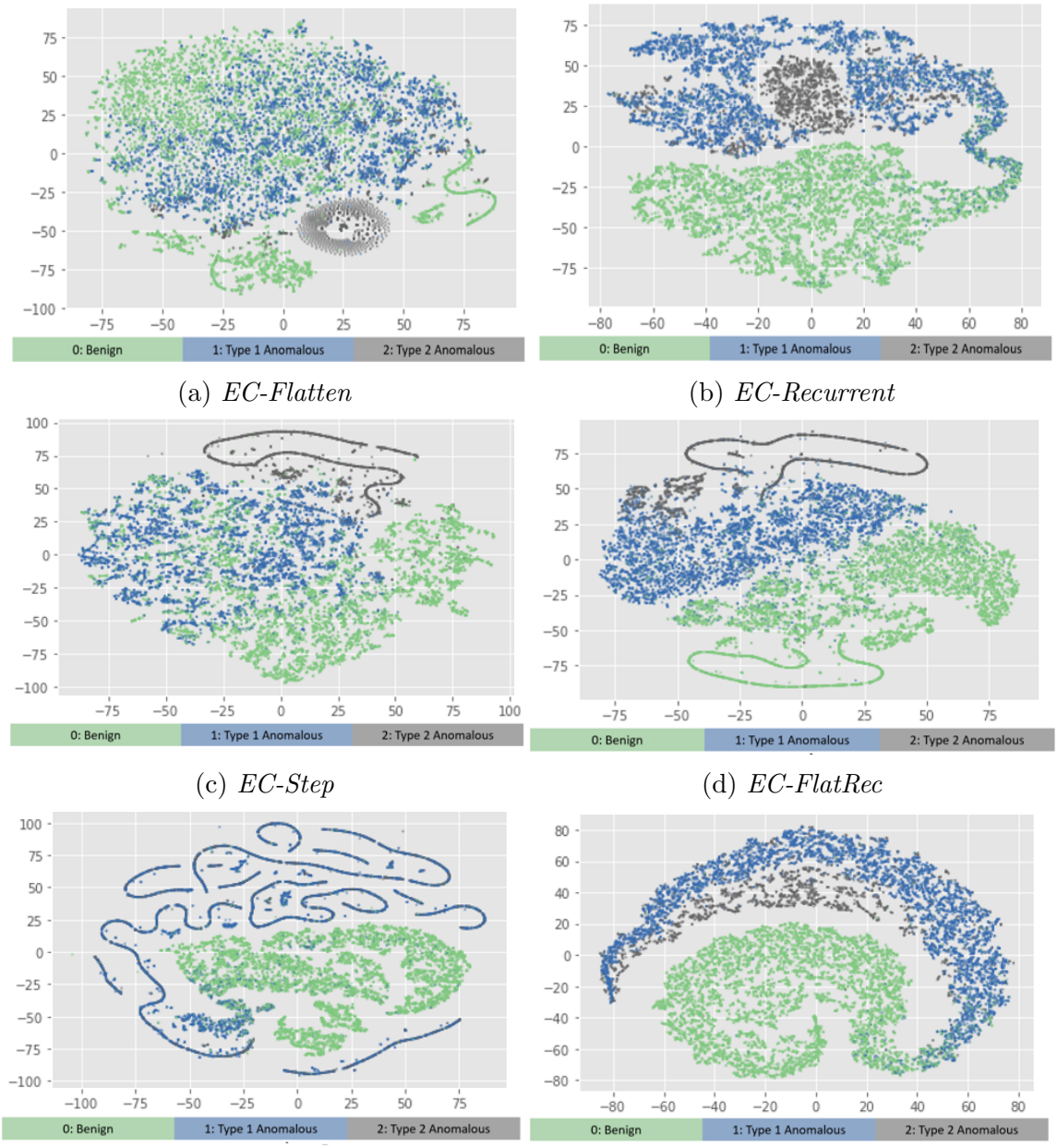


Figure 7–5: t-SNE visualization of the *mSDA* embedding(low granularity)

The visualization of the *AE8* embedding *AE9* embedding, and *mSDA* embedding barely show any clear pattern. The data points of different classes overlap severely.

7.2.2 High Granularity Visualization



(a) *EC-Flatten* (b) *EC-Recurrent*

(c) *EC-Step* (d) *EC-FlatRec*

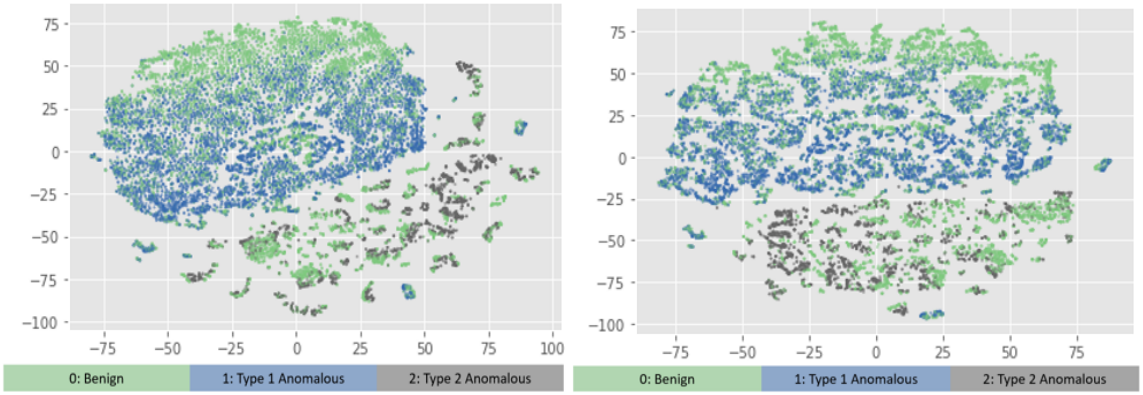
(e) *EC-StepRec* (f) *EC-ReStepRec*

Figure 7–6: t-SNE visualization of the *EC-Flatten*, *EC-Recurrent*, *EC-Step*, *EC-FlatRec*, *EC-StepRec*, and *EC-ReStepRec* embeddings (high granularity)

The *EC-Flatten*, *EC-Step* and *EC-StepRec* embeddings present obviously unsatisfying separation. In Figure 7–6a, the visualization of the *EC-Flatten* embedding, only points representing the embedding of a T2 anomalous sample are well grouped. The T1 anomalous samples and the benign samples do not form clear clusters but mix with each other. Figure 7–6c shows that the *EC-Step* embedding has the similar issue. The *EC-StepRec* embedding, as shown in Figure 7–6e, can distinguish the anomalous samples well but it cannot distinguish the T1 and T2 anomalous samples.

Compared with the aforementioned embeddings, the visualization for the *EC-StepRec* embedding and the *EC-Recurrent* embedding look better. In Figure 7–6d, it is clear that three groups correspond to three classes. However, there are quite a few dispersed points, especially those points that representing a benign sample but located in the region populated with T1 anomalous points. In Figure 7–6b, quite a few points representing T2 anomalous samples are lost in the region with dense T1 anomalous points.

The *EC-ReStepRec* embedding presents the best visualization. In Figure 7–6f, the separation between classes is clear, although the boundary between T2 anomalous points and T1 anomalous points is not wide and clear enough.



(a) *AE8* (b) *AE9*

Figure 7–7: t-SNE visualization of the *AE8* and *AE9* embeddings(high granularity)

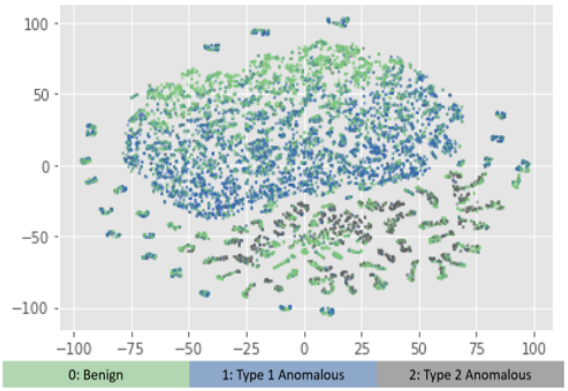


Figure 7–8: t-SNE visualization of the *mSDA* embedding(high granularity)

Figure 7–7a, Figure 7–7b, and Figure 7–8 show that the *AE8* embedding, the *AE9* embedding, and the *mSDA* embedding are capable to map the samples of two different anomalous classes into different regions. However, benign samples are distributed everywhere and overlap with the samples of the other classes.

7.3 Result Summary

Overall, the *EC-ReStepRec* embedding achieves the best performance on both tasks in terms of both superiority and robustness. The t-SNE visualizations also suggest that the *EC-ReStepRec* ccan form clusters for different classes and the clusters have relatively clear boundaries. All the evaluation results show that the *EC-ReStepRec* embedding is of the best quality.

Chapter 8
Discussion

Here we raise a few questions inspired by our experimental observations. By analyzing these questions, we interpret the results and reflect on the models.

8.1 Why does *EC-ReStepRec* outperform the other candidates?

Our evaluation results suggest that *EC-ReStepRec* yields the best embedding for anomaly detection. This could be interpreted in two perspective.

- Assumption: Right assumptions is critical to the effectiveness of models. The outstanding performance of *EC-ReStepRec* indicates that the learning preference of *EC-ReStepRec* has the best fit of the health insurance claims, which implicitly means that the assumptions corresponding to *EC-ReStepRec* describe the health insurance claim well.
- Regularization: Given the fact that our dataset is relatively small for a deep learning study, we are under the risk of overfitting. Although we use techniques such as dropout and batch normalization to alleviate it as much as possible, we cannot guarantee the techniques have led us to the safe zone. By observing the learning curves for the models with our proposed embedding components, we notice that the learning curves for *EC-ReStepRec* have the minimum gap between the training performance and validation performance. This might indicate that *EC-ReStepRec* has a good regularization effect in nature and thus it generalizes well.

8.2 Why do the baselines fail?

The performance for the baseline methods is disappointing. We have identified the possible reasons.

- *Autoencoder* for anomaly detection is commonly used when benign cases significantly dominate the dataset. However, our dataset does not really meet the condition. Besides, the *autoencoders* are trained based on *MSE*, which is a general metric. Considering the training of the embedding components, which are associated with a domain-specific task, lacking domain background in the training process of the *autoencoders* could be a reason.
- *mSDA* requires domain-specific preliminary treatments before being applied. It is possible that the standard preprocessing procedure used here is not sufficient to benefit from the capacity of *mSDA*.

.

Chapter 9
Conclusion, Lesson Learned, and Future Work

In this thesis, we present our method for health insurance claims embedding. We discuss six embedding components that are designed based on different assumptions. Our experiments on health insurance claims show that one of our proposed embedding components, *EC-ReStepRec*, achieves the best embedding for anomaly detection.

Next, we would like to share the lesson learned from this university-industry collaboration. Both the deep learning domain and the health insurance industry are complex. There was a steep learning curve for both parties at the early stage of the project. In addition to tackling technical challenges, a lot of effort was spent on gathering and labeling the data with the consideration of privacy, security, and ethical issues. Given our encouraging research results, all these efforts pay off.

Finally, we list some ideas about future work.

- As we mentioned above, limited by the resource we had, the size of data involved in this study is generally insufficient for a deep learning research. In order to further prove the effectiveness of our proposal, a dataset with more number of samples and more anomalous patterns should be accumulated and tested.
- Besides, our evaluation method is highly empirical-based and closely associated with our specific tasks. A more scientific embedding evaluation method for

health insurance claims embedding, or even further, for claim embedding is in need. There have been advanced embedding evaluation methods in branches such as natural language processing and computer vision, but for other growing application branches, it is still hard to find straightforward ways to interpret the features captured in embeddings and thus it is hard to evaluate. We encourage efforts and contributions that improve the interpretability of embeddings.

- Additionally, because the assumptions we made on the target data are quite general, it is possible that our work could also be applied to other similar datasets, for example, other transactional datasets with the characteristics of high dimensionality and sequentiality.

Appendix Classification Tasks Performance

Table 9–1: Binary classification performance on test set (0,1 indicate the F1-score on the benign class and the anomalous class respectively. m indicates the micro-average F1-score.)

score(.)		*EC-Flatten*	*EC-Recurrent*	*EC-Step*	*EC-FlaRec*	*EC-StepRec*	*EC-ReStepRec*	AE8	AE9	*mSDA*
KNN(5)	0	81	81	80	81	82	92	75	74	47
	1	83	83	82	82	83	93	78	77	55
	m	82	82	**81**	**82**	**82**	92	**77**	76	**51**
L-SVM	0	86	82	81	81	82	92	75	75	48
	1	87	83	82	83	82	93	77	79	52
	m	**87**	82	81	82	82	92	76	**77**	50
R-SVM	0	76	82	73	81	81	92	75	74	48
	1	81	83	81	82	82	93	77	77	53
	m	79	82	78	82	82	**93**	76	75	50
DT	0	78	82	75	80	81	92	71	72	42
	1	81	83	80	82	81	92	78	78	57
	m	80	**83**	78	81	81	92	75	75	50
RF	0	72	81	73	81	81	92	71	71	42
	1	78	83	79	83	82	93	77	78	56
	m	75	82	77	82	82	92	74	75	50
Ada	0	83	82	78	81	81	92	74	74	43
	1	85	83	81	83	82	93	78	79	56
	m	84	82	80	82	82	92	76	76	50
NB	0	67	82	72	67	81	92	67	67	42
	1	36	84	79	76	83	93	26	24	55
	m	57	83	76	72	82	93	54	54	50
LR	0	84	82	80	81	82	92	75	75	49
	1	86	83	82	83	82	92	76	78	52
	m	85	82	81	82	82	92	76	76	51
LDA	0	86	82	80	81	82	92	75	75	49
	1	87	83	82	82	82	92	77	78	52
	m	86	82	81	82	82	92	76	76	51
QDA	0	66	82	79	55	82	92	68	68	60
	1	14	83	76	76	83	93	27	26	28
	m	51	83	78	69	82	93	56	55	49

Table 9–2: Three-Class classification performance on test set (0, T1, T2, indicate the F1-score on the benign class, T1 anomalous class, and T2 anomalous class respectively. m indicates the micro-average F1-score.)

score(.)		*EC-Flatten*	*EC-Recurrent*	*EC-Step*	*EC-FlaRec*	*EC-StepRec*	*EC-ReStepRec*	AE8	AE9	*mSDA*
KNN(5)	0	81	81	80	81	82	92	75	74	47
	T1	76	75	76	74	62	86	75	74	41
	T2	83	87	92	87	24	75	87	87	13
	m	79	79	**80**	**79**	68	**88**	**77**	76	40
L-SVM	0	86	82	81	81	81	92	75	75	49
	T1	78	74	76	74	65	77	75	77	39
	T2	77	79	88	86	1	0	88	88	13
	m	**82**	78	80	79	**69**	79	77	**78**	40
R-SVM	0	76	82	74	81	81	92	75	74	49
	T1	66	67	71	73	65	77	74	74	39
	T2	0	13	63	83	0	0	84	76	12
	m	66	70	72	78	69	79	76	74	40
DT	0	78	82	77	80	82	92	70	72	40
	T1	69	76	73	73	65	79	79	79	43
	T2	75	89	84	84	5	30	78	79	14
	m	74	**81**	76	78	69	81	75	76	38
RF	0	75	81	69	75	82	92	71	70	46
	T1	57	68	68	70	65	80	76	77	42
	T2	0	18	1	44	0	33	69	66	8
	m	62	71	64	70	69	82	73	72	40
Ada	0	78	82	78	82	81	92	38	34	34
	T1	76	76	75	74	65	78	76	71	50
	T2	83	94	88	85	3	9	65	64	0
	m	78	81	78	79	69	80	62	58	40
NB	0	61	81	67	62	81	92	25	31	9
	T1	36	74	64	71	32	69	68	69	49
	T2	48	74	66	62	47	38	63	59	18
	m	51	78	65	66	59	76	54	56	32
LR	0	84	82	81	81	81	91	75	75	49
	T1	77	72	76	73	65	76	73	76	39
	T2	79	74	86	84	1	0	88	87	13
	m	81	77	79	79	69	79	76	77	40
LDA	0	86	82	80	81	82	92	75	74	48
	T1	79	75	75	72	65	76	74	74	39
	T2	79	89	85	83	4	26	87	85	13
	m	82	80	79	78	69	79	76	76	40
QDA	0	65	82	60	43	81	92	35	36	61
	T1	43	75	63	63	67	56	69	70	0
	T2	45	83	67	78	3	48	70	67	17
	m	53	79	63	59	69	71	58	58	**42**

References

[1] Pedro A. Ortega, Cristián J. Figueroa, and Gonzalo Ruz. A medical claim fraud/abuse detection system based on data mining: A case study in chile. volume 6, pages 224–231, 01 2006.

[2] Alejandro Correa Bahnsen, Djamila Aouada, Aleksandar Stojanovic, and Björn Ottersten. Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51:134–142, 2016.

[3] Leonardo Baldassini and Jose Antonio Rodríguez Serrano. client2vec: towards systematic baselines for banking applications. *arXiv preprint arXiv:1802.04198*, 2018.

[4] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 891–900. ACM, 2015.

[6] A P Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. An autoencoder approach to learning bilingual word representations. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS 14)*, volume 2, pages 1853–1861. MIT Press, 2014.

[7] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Coference on International Conference on Machine Learning (ICML12)*, pages 1627–1634. Omnipress, 2012.

[8] Youngduck Choi, Chill Yi-I Chiu, and David Sontag. Learning low-dimensional representations of medical concepts. *AMIA Summits on Translational Science Proceedings*, pages 41–50, 2016.

[9] Ali Batuhan Dayioglugil and Yusuf Sinan Akgul. Continuous embedding spaces for bank transaction data. In *Proceedings of the 23rd International Symposium, Foundations of Intelligent Systems (ISMIS 2017)*, pages 129–135. Springer International Publishing, 2017.

[10] Lance De Vine, Guido Zuccon, Bevan Koopman, Laurianne Sitbon, and Peter Bruza. Medical semantic similarity with a neural language model. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM 14)*, pages 1819–1822. ACM, 2014.

[11] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[12] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[14] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. 2003.

[15] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[16] Marina Evrim Johnson and Nagen Nagarur. Multi-stage methodology to detect health insurance claim fraud. *Health Care Management Science*, 19(3):249–260, 2016.

[17] Hossein Joudaki, Arash Rashidian, Behrouz Minaei-Bidgoli, Mahmood Mahmoodi, Bijan Geraili, Mahdi Nasiri, and Mohammad Arab. Improving fraud and abuse detection in general physician claims: a data mining study. *International Journal of Health Policy and Management (IJHPM)*, 5(3):165–172, 2016.

[18] Melih Kirlidog and Cuneyt Asuk. A fraud detection approach with data mining in health insurance. *Procedia-Social and Behavioral Sciences*, 62:989–994, 2012.

[19] Hian Koh and Gerald Tan. Data mining applications in healthcare. *Journal of healthcare information management : JHIM*, 19:64–72, 02 2005.

[20] Mohit Kumar, Rayid Ghani, and Zhu-Song Mei. Data mining to predict and prevent errors in health insurance claims processing. In *Proceedings of the*

16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pages 65–74. ACM, 2010.

[21] Michael Leonard and Brenda Wolfe. Mining transactional and time series data. In *Proceedings of the 30th Annual SAS® Users Group International Conference (SUGI 30)*, 2005.

[22] Fen-May Liou, Ying-Chan Tang, and Jean-Yi Chen. Detecting hospital fraud and claim abuse through diabetic outpatient services. *Health Care Management Science*, 11(4):353–358, 2008.

[23] Qi Liu and Miklos Vasarhelyi. Healthcare fraud detection: A survey and a clustering model incorporating geo-location information. In *Proceedings of the 29th World Continuous Auditing and Reporting Symposium (29WCARS), Brisbane, Australia*, 2013.

[24] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[26] Che Ngufor and Janusz Wojtusiak. Unsupervised labeling of data for supervised learning and its application to medical claims prediction. *Computer Science*, 14(2):191, 2013.

[27] Dang Nguyen, Tu Dinh Nguyen, Wei Luo, and Svetha Venkatesh. Trans2vec: learning transaction embedding via items and frequent itemsets. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 361–372. Springer, 2018.

[28] Christopher Olah. Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. [Online; accessed 19-June-2019].

[29] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[30] Marco Schreyer, Timur Sattarov, Damian Borth, Andreas Dengel, and Bernd Reimer. Detection of anomalies in large scale accounting data using deep autoencoder networks. *arXiv preprint arXiv:1709.05254*, 2017.

[31] Yin Shan, David Jeacocke, D Wayne Murray, and Alison Sutinen. Mining medical specialist billing patterns for health service management. In *Proceedings of the 7th Australasian Data Mining Conference (AusDM 2008)*, volume 87, pages 105–110. Australian Computer Society, Inc., 2008.

[32] Hyunjung Shin, Hayoung Park, Junwoo Lee, and Won Chul Jhee. A scoring model to detect abusive billing patterns in health insurance claims. *Expert Systems with Applications*, 39(8):7441–7450, 2012.

[33] Dallas Thornton, Guido van Capelleveen, Mannes Poel, Jos van Hillegersberg, and Roland M Mueller. Outlier-based health insurance fraud detection for us medicaid data. In *Proceedings of the 16th International Conference on Enterprise Information Systems (ICEIS 2014)*, pages 684–694. SCITEPRESS, 2014.

[34] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[35] Hastie Trevor, Tibshirani Robert, and Friedman Jerome. *The elements of statistical learning: data mining, inference, and prediction.* New York, NY: Springer, 2009.

[36] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1225–1234. ACM, 2016.

[37] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI18)*, pages 2532–2539, 2018.

[38] Janusz Wojtusiak, Che Ngufor, John Shiver, and Ronald Ewald. Rule-based prediction of medical claims' payments: A method and initial application to medicaid data. In *Proceedings of the 10th International Conference on Machine Learning and Applications and Workshops (ICMLA 2011)*, volume 2, pages 162–167. IEEE, 2011.

[39] Harry Zhang. The optimality of naive bayes. pages 562–567, 2004.