論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	One-class Text Classification with Neural Networks
著者(和文)	HuChenlong
Author(English)	Chenlong Hu
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第12070号, 授与年月日:2021年9月24日, 学位の種別:課程博士, 審査員:奥村 学,熊澤 逸夫,中山 実,篠﨑 隆宏,船越 孝太郎
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第12070号, Conferred date:2021/9/24, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

One-class Text Classification with Neural Networks

Chenlong Hu

A Doctoral Thesis

Department of Information Processing,
Interdisciplinary Graduate School of Science and Engineering,
Tokyo Institute of Technology

Supervisor: Manabu Okumura, Professor

2021

Abstract

In this study, we apply neural networks to the problem of one-class text classification. One-class (text) classification is a special classification problem that aims to learn a model on the basis of training samples only from one class. It would be beneficial in a scenario where a large number of labelled negative samples are hard or time-consuming to obtain, while positive training data samples are available.

This dissertation studies one-class text classification in two successive parts. The first part focuses on the task of acquiring semantic lexicons using a *bootstrapping* technique. *Bootstrapping* methods are efficient tools to explore knowledge from unannotated corpora with increasing text size. Specifically, we formulate *Basilisk*, an effective and commonly used bootstrapping approach, as a special one-class problem and incorporate AutoEncoder to improve its learning processes. The experimental results demonstrate that the proposed methods for guiding the bootstrapping of a semantic lexicon with AutoEncoder can boost overall performance.

The second part proposes a generic one-class text learning framework with a *multi-modal deep support vector data description*, which we call mSVDD. By extending the uni-modal deep support vector data description to a multiple modal one, we build mSVDD with multiple hyperspheres, which enable us to build a much better description for the target one-class data. Additionally, the end-to-end architecture of mSVDD can jointly handle neural feature learning and one-class text learning. We also introduce a mechanism for incorporating negative supervision in the absence of real negative data, which can be beneficial to the mSVDD model. The experimental results demonstrate that mSVDD outperforms uni-modal SVDD and can get further improvements when negative supervision is incorporated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

One-Class Classification (OCC), a special classification problem, aims to learn a model on the basis of training samples only from *one* class [106, 157, 26]. The learned model is expected to make an accurate description of the class, called *target, positive* or *normal*, and to distinguish the *target* from samples for negative classes during testing [110, 157].

Here, we formally give a definition of the problem. Let $\mathcal{X}$, which is a compact subset of $\mathbb{R}^d$, be the data space that contains all data samples from the *target* class. We also define an unknown $\mathbb{P}$ that may be a uni-modal or multi-modal distribution where the data comes from. For the observations (or data samples) $\boldsymbol{x} \in \mathcal{X}$ that is drawn from $\mathbb{P}$, the objective of one-class classification is to find a scoring function $s(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ such that:

$$s(\boldsymbol{x}) > s(\boldsymbol{x}'), \text{[1]} \tag{1.1}$$

where $\boldsymbol{x}$ denotes a sample from the target class, while $\boldsymbol{x}'$ comes from other classes. The above formulation indicates that a positive sample $\boldsymbol{x}$ should be assigned a larger score than a negative one $\boldsymbol{x}'$. In the problem setting, negative data samples $\boldsymbol{x}' \notin \mathcal{X}$ are not available in the training phase. We have to learn the scoring function on the basis of a training set comprising only the target samples, $T = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathcal{X}$, where $n \in \mathbb{N}$ is the size.

The one-class classification problem has arisen in many real-world applications, including anomaly or novelty detection [137, 15, 36], bioinformatics [2], medical di-

[1] Or $s(\boldsymbol{x}) < s(\boldsymbol{x}')$ for a measure of the distance.

agnosis [18, 88], disease outbreaks detection [168], and especially computer vision [138, 140]. It has been applied to various types of data such as audio [130], video [109], and text [99]. As for one-class text classification, it can be applied to the scenario where a large number of labelled negative samples (e.g., web pages, spam emails) are hard or time-consuming to obtain [176, 45], while positive training data samples (e.g., ham emails) are available.

One of the early studies on one-class text classification is [99], who implemented versions of one-class support vector machines (OC-SVM) [149] and showed good performances on the *Reuters* dataset [25]. As for the approaches for solving this problem, OC-SVM and support vector data description (SVDD) [158] are *boundary*-based methods [157]. These methods try to directly describe the target data by using a boundary. SVDD learns an optimal hypersphere with the minimum radius to include the most target data, while OC-SVM builds a hyperplane to maximally separate the data points from the target where outlier examples lie around.

AutoEncoder [115, 142, 98] and principal component analysis (PCA) [122, 55, 52] are *reconstruction*-based methods. These approaches aim to describe the target data with a more compact representation obtained by optimizing a reconstruction error on the target training data. *Reconstruction*-based methods assume that the learned model can well-reconstruct normal data samples with smaller reconstruction errors, while *failing* to precisely reconstruct instances from negative classes, i.e., having larger reconstruction errors.

Regarding the features for representing text data, traditionally, document-to-word co-occurrence matrices or hand-crafted features have been commonly used in most of the previous work [99, 98, 76]. In recent years, *neural networks* (*deep learning*) [80, 146, 32] have been applied to numerous applications, such as speech recognition [46], computer vision [75, 43], and natural language processing (NLP) [7, 105, 23]. An effective feature representation with neural networks is necessary for various types of data (e.g., unannotated corpora in NLP) with continuously growing size [6, 27].

Traditional feature-learning approaches assume that data representations follow a fixed priori (e.g., an implicit feature map with Gaussian kernel [53]). While hand-crafting features or choosing suitable kernels for certain tasks would be ineffective and challenging for complex data, by using gradient optimization algorithms and GPU parallelization, neural networks can be scaled up to process vast amounts of data. Moreover, neural networks with different multiple ("deep") layer structures (e.g., long short-term memory (LSTM) [147] for sequence data) can involve complex data.

For example, vector representations of words and sentences can be obtained by pre-training different neural networks, including word2vec [105] and SkipThought [70], on a large-scale corpus, e.g., BooksCorpus, which has 800M words [180].

Recent studies have developed neural networks for the problem of one-class classification. The approaches include *reconstruction*-based methods such as AuoEncoder and its variants [49, 96, 97, 141], as well as *boundary*-based methods [140, 14]. In these approaches, the models not only optimize the original objective function (e.g., reconstruction loss) but also learn a feature map $\phi(\boldsymbol{x}; \mathcal{W}) : \mathcal{X} \rightarrow \mathcal{F}$ (where $\mathcal{F}$ is the output feature space) with deep neural networks, parameterized with weights $\mathcal{W}$. For example, the deep support vector data description (deep SVDD) is a fully end-to-end boundary-based method for image data [140]. Deep SVDD learns to extract feature representations of training images with a convolutional neural network (CNN) [81] and optimizes the volume of a hypersphere in a space $\mathcal{F}$.

In this dissertation, we would like to apply neural networks to the problem of one-class text classification. This dissertation studies one-class text classification in two successive parts.

The first part addresses the task of acquiring semantic lexicons by using a *bootstrapping* technique. Bootstrapping methods are efficient tools for exploring knowledge, such as semantic lexicon, from unannotated corpora with increasing text size [132]. We focus on *Basilisk* (Bootstrapping approach to semantic lexicon induction) [159], an effective and commonly used bootstrapping approach for constructing semantic lexicon from text corpora. To start the bootstrapping process of Basilisk, all you need is to input an initial *seed list* with a limited number of seed words. Our motivation is that, since we have only these seed instances, which can be considered *positive* samples from *one* semantic category (e.g., *food* domain), we can try to solve it within the one-class learning framework. Accordingly, we address the following research issues in the first part. 1) Can we formulate the framework of Basilisk as a one-class problem? 2) How do we improve its bootstrapping process with neural networks?

The second part of the thesis investigates a wider range of *boundary*-based methods with neural networks. As introduced above, OC-SVM and SVDD, two well-known one-class classification methods, optimize a boundary to describe target data. Recently, by incorporating explicit neural feature maps $\phi(\boldsymbol{x}; \mathcal{W})$, OC-SVM and SVDD have been extended to one-class neural network (OC-NN) [14] and deep support vector data description (deep SVDD) [140], respectively. Deep SVDD learns a hypersphere model with a center while OC-NN utilizes a linear model with margin. Deep

SVDD and OC-NN outperform their kernel-based versions. Nevertheless, target data samples may have distinctive distributions that are located in different regions. Therefore, a uni-modal deep SVDD or OC-NN may not be enough to describe the target samples. In the second part, we try to extend the deep SVDD to a version with multiple modes, a *multi-modal deep support vector data description* (mSVDD), where each mode is expected to describe the target samples from a distinctive aspect. Consequently, we address three research issues in the second part: 1) How do we develop uni-modal deep support vector data description into a multi-modal version? 2) What are the relationships with other related models? 3) Due to the unavailability of training samples from negative classes, it is hard for the one-class models to learn effective discrimination information, especially for mSVDD with a multi-layer neural structure. Can we incorporate *negative supervision* to improve mSVDD in some way?

1.2 Contributions of This Thesis

The main contributions of this thesis are as follows.

Regarding the topic of AutoEncoder guided bootstrapping of the semantic lexicon:

- We use AutoEncoder to guide the learning of the bootstrapping in a totally unsupervised manner.
- We improve the candidate evaluation by incorporating AutoEncoder, which does not rely on any additional negative training data.
- We also present a new function to boost the scoring of patterns such that better candidates will be pooled in subsequent steps. The performance of candidate scoring can be improved by including our new pattern scoring function.

Regarding the topic of one-class text classification with multi-modal deep support vector data description:

- We propose a generic end-to-end one-class neural network learning framework, called mSVDD, to extend the uni-modal deep SVDD to an end-to-end multi-modal one.
- We also prove that four one-class classification models, i.e., deep SVDD [140], context vector data description (CVDD) [141], one-class neural networks (OC-NN) [14], and deep multi-sphere SVDD (DMSVDD) [29], are all special cases of the mSVDD model under certain conditions.

- We also propose approaches for effectively incorporating negative supervision information to improve the performance of the proposed mSVDD.

1.3 Outline of This Thesis

The rest of this thesis is organized as follows.

Chapter 2 This chapter introduces related work on one-class classification, bootstrapping methods, and so on.

Chapter 3 At the beginning of this chapter, we describe *Basilisk*, a commonly used framework for bootstrapping lexicons. Then, we formulate this framework as a special one-class classification problem and describe the proposed methods with AutoEncoder to guide its learning iterations. Finally, the experimental results demonstrate that our proposed methods for guiding the bootstrapping of a semantic lexicon with AutoEncoder can boost overall performance.

Chapter 4 In this chapter, we describe the framework of multi-modal deep support vector data description (mSVDD). We show how to describe the target data with multiple hyperspehres. We prove propositions to elaborate the relationships between mSVDD and other models. In addition, the methods for incorporating negative supervision for mSVDD are also introduced. Experimental results demonstrate that mSVDD outperforms uni-modal SVDD and that it can get further improvements when negative supervision is incorporated.

Chapter 5 Finally, Chapter 5 summarizes this dissertation and discusses some directions for future work.

Chapter 2

Related Work

In this chapter, we first summarize the related work for one-class classification and its corresponding methods. Then we show the related work for bootstrapping approaches. Other related topics for multi-modal deep support vector data description are also included.

2.1 One-class Classification

The first related area would be one-class learning or classification. Generally, one-class classification approaches can be divided into the following groups: density estimation-based methods, boundary-based methods, and reconstruction-based methods [157, 65, 66]. As for *density estimation*, early work uses different probability models, including Gaussian [9] and Parzen [73], to estimate the density of the training data and sets the threshold for the *target* of high probabilities in the distribution. *Density estimation*-based methods, including early work uses different probability models [120, 41, 68], estimate the density of the training data and sets the threshold for the *target* of high probabilities in the distribution.

2.1.1 Reconstruction-based Methods

Reconstruction-based methods, including AutoEncoder [115, 142, 98] and principal component analysis (PCA) [122, 52], attempt to learn a more compact representation for the description of target data. Most of these methods learn a model that is optimized to reduce the reconstruction error of normal data samples. CVDD [141], that optimizes the one-to-one reconstruction distance between context vectors and self-

attention feature vectors, can be viewed as a kind of Reconstruction-based approach.

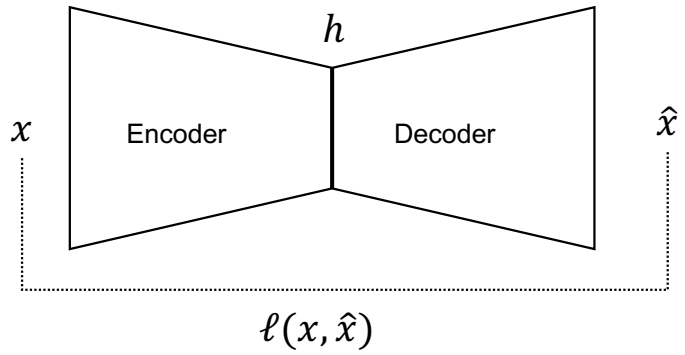


Figure 2.1: A reconstruction model with encoder and decoder. $\boldsymbol{x}$ denotes input, while $\hat{\boldsymbol{x}}$ denotes the reconstructed output.

As shown in Fig. 2.1, a *reconstruction-based* model consists of two parts, the encoder $\phi_{en}(\boldsymbol{x}; \theta_{en}) : \mathcal{X} \rightarrow \mathcal{F}$ and the decoder $\phi_{de}(\boldsymbol{x}; \theta_{de}) : \mathcal{F} \rightarrow \mathcal{X}$, where θ_{en} and θ_{en} are parameters. We call $\mathcal{F}$ the *latent* space, $\boldsymbol{h} = \phi_{en}(\boldsymbol{x}; \theta_{en})$ latent variables, or a latent representation, and $\hat{\boldsymbol{x}} = \phi_{de}(\boldsymbol{h}; \theta_{de})$ the reconstructed output which is expected to be close to $\boldsymbol{x}$, i.e., a smaller reconstruction error.

Give the unlabelled training data $T = \boldsymbol{x}_1, ..., \boldsymbol{x}_n$, we give a general definition of the objection function of *reconstruction-based* models as follow:

$$\min_{\theta_{en}, \theta_{de}} \quad \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{x}_i, \phi_{de}(\phi_{en}(\boldsymbol{x}_i; \theta_{en}); \theta_{de})) + \mathcal{R}(\theta) \tag{2.1}$$

where $\mathcal{R}$ denotes regularization item, and ℓ denotes the loss function which is usually measured in Euclidean distance. If we assume that training data $T = \boldsymbol{x}_1, ..., \boldsymbol{x}_n$ are mostly normal data samples, a reconstruction-based model trained on T is expected to have a *high* for anomalies and a *low* reconstruction error for normal samples, i.e., the samples from target *one*-class. Accordingly, the score function can be defined by the reconstruction error:

$$s(\boldsymbol{x}) = \ell(\boldsymbol{x}, \phi_{de}(\phi_{en}(\boldsymbol{x}; \theta_{en}); \theta_{de})) \tag{2.2}$$

AutoEncoder and its variants [58, 47, 49] are reconstruction models that use neural networks as the encoder and the decoder and optimize reconstruction loss by back-propagation. If we denote $\phi_{en}(\cdot; \mathcal{W}_{en})$ and $\phi_{de}(\cdot; \mathcal{W}_{de})$ are two neural networks for encoder and decoder, where $\mathcal{W}_{en}$ and $\mathcal{W}_{de}$ are weights. The objective function of

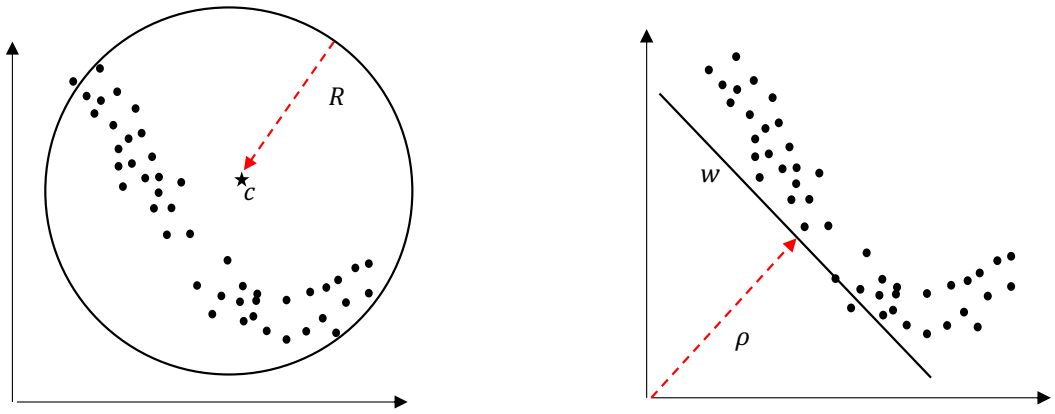
AutoEncoder is usually defined as:

$$\min_{\mathcal{W}_{en},\mathcal{W}_{de}} \quad \frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{x}_i - \phi_{de}(\phi_{en}(\boldsymbol{x}_i;\mathcal{W}_{en});\mathcal{W}_{de})\|^2 + \mathcal{R}(\mathcal{W}) \tag{2.3}$$

Regularly, AutoEncoder is regularized by forcing the hidden feature space $\mathcal{F}$ to have lower dimensionality than the input space $\mathcal{X}$, i.e., the "bottleneck" representation $h = \phi_{en}(\boldsymbol{x};\mathcal{W}_{en})$ compresses representation of the input $\boldsymbol{x}$ and limits the dimensionality of the learned manifold or subspace [49].

2.1.2 Boundary-based Methods

Boundary-based methods learn only a boundary around the target set. As mentioned in Chapter 1, one-class support vector machines (OC-SVM) seeks a hyperplane to separate the data from the origin, while support vector data description (SVDD) finds a hypersphere to enclose the data. Their neural extensions, DSVDD [140] and OC-NN [14], which we will introduce in Chapter 4, can also optimize the related boundaries in the neural feature space.



(a) SVDD represented by $(\boldsymbol{c}, R)$. (b) OC-SVM represented by $(\boldsymbol{w}, \rho)$.

Figure 2.2: Data descriptions by SVDD (left) and OC-SVM (right) without the incorporation of kernels.

Support vector data descriptions

Support vector data descriptions (SVDD) is a support vector learning method for one-class classification. It aims at constructing an optimal boundary in a feature space

that includes almost all normal target data, given only the target training samples, $T = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathcal{X}$, where $n \in \mathbb{N}$ is the size of the training data, and $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$. As shown in Fig. 2.2a, the main idea of SVDD is to optimize a hypersphere with a center $\boldsymbol{c}$ and radius R, that encloses the majority of the data. SVDD solves the following quadratic problem:

$$\begin{aligned} &\min_{R,\boldsymbol{c},\boldsymbol{\xi}} \quad R^2 + C\sum_i \xi_i \\ &s.t.\ \|\boldsymbol{x}_i - \boldsymbol{c}\|_2^2 \leq R^2 + \xi_i,\ \xi_i \geq 0, \quad \forall i = 1, ..., n, \end{aligned} \tag{2.4}$$

where ξ_i is a slack variable for allowing a flexible boundary. C is a regularization parameter, that is usually represented by $\frac{1}{\nu n}$, where $\nu \in (0, 1]$ is a parameter that controls the tradeoff between the radius of the hypersphere and the penalties ξ_i.

One-class support vector machines

One-class support vector machines (OC-SVM) [149] is another commonly used approach for solving one-class classification problems. Unlike SVDD, OC-SVM intends to seek the best hyperplane by maximizing the margin between the data points and the origin, as shown in Fig. 2.2b. Formally, given the training data $T = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathcal{X}$ like SVDD, OC-SVM solves the following optimization problem:

$$\begin{aligned} &\min_{\rho,\boldsymbol{w},\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \rho + \frac{1}{\nu n}\sum_i \xi_i \\ &s.t.\ \boldsymbol{w}^T\boldsymbol{x}_i \geq \rho - \xi_i,\ \xi_i \geq 0, \quad \forall i = 1, ..., n, \end{aligned} \tag{2.5}$$

where ρ is the margin between the origin and the hyperplane $\boldsymbol{w}$, and ξ_i is a slack variable for allowing data to cross the hyperplane boundary. As with SVDD, $\nu \in (0, 1]$ is a hyperparameter that controls the trade-off between the distance from the origin to the hyperplane and the number of data points that are allowed to cross the hyperplane.

The kernel trick [148] can be incorporated into OC-SVM and SVDD to obtain more flexible descriptions. Assuming space $\mathcal{H}$ is *reproducing kernel Hilbert space* (RKHS) [143] and $K(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}')$ is a kernel function, $\Phi(x) : \mathcal{X} \to \mathcal{H}$ becomes an implicit feature mapping function from the input space $\mathcal{X}$ to the feature space $\mathcal{H}$. In this case, the objectives of the kernel-based OC-SVM and SVDD (Eq. (2.5) and Eq. (2.4)) are to find respectively a hyperplane and a hypersphere in feature space $\mathcal{H}$. Then, both can be solved in the corresponding dual optimization problems,

which are quadratic functions involving the dot products of the transformed vectors. And every dot product can be replaced with a nonlinear kernel function, without knowing the explicit form of $\Phi(x)$.

Relationship between SVDD and OC-SVM

The identity of SVDD and OC-SVM has been mentioned in several papers [160, 157, 162]. [157](cf. Section 2.5, [157]) proved that although the hyperplane obtained by OC-SVM is not a closed boundary around the data that SVDD optimized, both can be represented by an equivalent formulation when the data samples are normalized to unit norm vectors. Also for the standard Gaussian kernel, where the data is implicitly rescaled to norm 1, these two one-class approaches are also equivalent to each other.

Several efforts have been proposed to extend kernel-induced OC-SVM and SVDD with multi-modal forms. [40] was early work to use multi-sphere SVDD, which was used for multi-class tasks. For one-class tasks, [171] used multi-sphere SVDD to encode multi-distribution target data. Multiple kernel learning was proposed for OC-SVM [21]. Two more efforts have been proposed by [79, 78], which found the optimal solution by an iterative algorithm consisting of the following two steps: 1) calculate radii and centers, and 2) calculate the assignments of data to centers. While one limitation of SVDD and OC-SVM, along with their extensions, would be that it has to perform hand-crafted feature engineering [117], the limitation can be solved by incorporating neural models into SVDD.

2.1.3 Relationship between One-class Classification and Other Problems

As a special learning problem, the relationships of one-class classification (OCC) with other problems (e.g, multi-class and binary classification) can be confusing and are not fully clear [157, 65, 125]. In this section, we would like to discuss the connection and difference between OCC and other topics from two aspects: *supervised* and *unsupervised* learning.

One-class and supervised learning models

Supervised learning is defined by its use of labeled data to train algorithms. In the view of *supervised* learning, one-class classification can be seen as a special case of

supervised classification problem, i.e., binary classification where training data samples from a single class are available and the negative class (classes) is absent, hard to obtain or poorly defined [66].

If so, the objective of OCC models is to learn classifiers using a positive set of examples, and to determine whether a test sample belongs to target class [63]. More specifically, in the context of supervised learning, OCC problem can be represented by the ordinary binary classification learning framework if we assume that most of the given training data samples are from target class and can be labelled as *positive* [1] [103, 139].

In addition, the absent negative class also needs some special assumptions about its distributions. For example, the form of support vector data descriptions (SVDD) (Eq. (2.4)) can be derived from a binary classification risk when we assume the negative data samples are *uniformly* distributed [158, 139]. As for one-class support vector machines (OC-SVM) [149], it also makes a assumption about the distribution of negative data, i.e., negative anomalies lie around the *origin*. Then OC-SVM optimizes a hyperplane that separates the region containing target data from the origin with maximum margin, which is similar to a binary SVM classifier [150]. In summary, one-class classification algorithms (e.g., SVDD [158]) can be formulated from a supervised binary classification task under conditions of particular assumptions about distribution of the negative class.

One-class and unsupervised learning models

Unsupervised learning is another type of machine learning which learns patterns from data with no labels or with minimal human supervision [30]. Clustering and dimensionality reduction are two popular unsupervised learning tasks [34, 49]. In the view of *unsupervised* learning, one-class classification is to make a description of the target class and to determine which (new) test samples resemble the target training set [157].

As introduced above, due to the absence of fully labelled anomalous data, we have to made a specific assumption about the distribution of anomalous data. Actually, one-class classification can be solved under *unsupervised* learning framework, i.e., to learn a model that can well describe the target data in an unsupervised manner, such that the model can detect anomalies through the deviations measured by reconstruction errors [157, 66, 139]. Therefore, some unsupervised algorithms that are

[1] If we denote $y = +1$ as positive, $y = -1$ as negative in a binary classification framework.

reconstruction-based methods, such as principal component analysis (PCA) [122, 52] which was mentioned before, can be used for the task of one-class classification.

	One-Class Classification	Other Tasks (e.g., clustering)
Task setting & objective	Data only from "one-class" are available in training phase. Learn model on target training data, and distinguish target samples from negative ones during testing phase.	Data without human labels are available during whole phases. Learn model by modeling the common characteristics of observing data.
Data	$T = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, training set with *prior* assumption that the majority of T are positive. $T^{te} = \{\boldsymbol{x}_1^{te}, ..., \boldsymbol{x}_m^{te}\}$, testing set.	$U = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, unlabeled set without prior knowledge. There is no distinction of training and testing.
Application	Anomaly detection [52], spam filtering [145]	Dimensionality reduction [49], clustering [34], warming-up [48]
Example (AutoEncoder)	(OCC) Use both encoder and decoder to evaluate the deviation of each sample: $\ell(\boldsymbol{x}, \phi_{de}(\phi_{en}(\boldsymbol{x}; \theta_{en}); \theta_{de}))$	(Dimensionality reduction) Use only encoder to get a lower dimension of feature representation: $\boldsymbol{h} = \phi_{en}(\boldsymbol{x}; \theta_{en})$
Example (K-means)	(OCC) Use prototypes to evaluate the deviation of each sample: $\min_k \|\boldsymbol{x} - u_k\|^2$, where u_k is the k-th prototype representations.	(Clustering) Use prototypes to get the cluster assignment of each sample: $A(x) = \arg\min_k \|\boldsymbol{x} - u_k\|^2$, where $A(x)$ means assignment.

Table 2.1: Reconstruction-based methods used for OCC and other tasks including clustering.

When unsupervised algorithms are used for OCC or other unsupervised tasks, the motivation is the same: the common characteristics of observing data can be learned by optimizing reconstruction-based objectives under kinds of assumptions about the data structures. More specifically, they all assert that a more compact representation, a set of prototypes or subspaces, can be used for the description of target data. For

instances, a *prototype assumption* indicates there exists a finite number of prototypical elements that describe the data structure or capture models of data distribution well [157, 139]. The most famous algorithm under prototype assumption would be K-means [94] which finds K prototypical instances describe data clusters well. For text data, [141] proposes a recent reconstruction-based method that represents the target data by a set of r *context* (prototype) vectors $C = (c_1, ..., c_r)$, is known as context vector data descriptions (CVDD).

However, there are many differences between OCC and ordinary unsupervised tasks such as clustering, as summarized in Table 2.1. The primary difference would be the objective. For example, specifically, the use of K-means for unsupervised clustering is to assign each observation to the nearest cluster center [95]. However, in OCC problem, we only use K-means to calculate the reconstruction-error which measures each sample the deviation from its closet center in testing [157, 74]. Another principal difference is the *prior* knowledge about the data learned [139]. Even the training process with reconstruction-based algorithms is unsupervised in OCC problem, we still have a prior assumption that the majority of training data are from the target class. While there exists no this kinds of assumption in tasks like dimensionality reduction [49].

To sum up, OCC tasks can be solved by reconstruction-based algorithms which are also commonly used for other unsupervised tasks, while the setting, objective, and data assumption are different.

2.2 Bootstrapping Methods

Information extraction Extracting knowledge, including semantic lexicon, has been addressed in the general area of *information extraction* (IE), and more specifically, *relation extraction*, which finds semantic relations, including semantic lexicons, in a text corpus [108, 62]. The extensive approaches under this topic, which share the common techniques with our task of extracting semantic lexicons, can be grouped into four categories: rule-based, supervised, distant-supervised, and bootstrapping methods. With a pioneering work by [44], who used manually constructed lexico-syntactic patterns to identify hyponyms, [13], [8] and [31] used the similar approach for identifying part-whole relations. However, these rule-based methods cannot be easily applied directly to other corpora [62] because each requires specific expert-designed patterns. A supervised algorithm, trained with human-labeled samples, is another straightforward

and effective approach where classifiers trained with hand-labeled corpora are used to label different knowledge [177, 156, 144]. However, it is time-consuming to obtain annotated data for a sizable corpus. Due to the above reasons, distantly supervised or weakly-supervised approaches have attracted more attention for obtaining semantic resources from a large-scale corpus. These methods attempt to build an alignment between a raw corpus and a knowledge database [107]. By using a database like *Freebase* [10], a large number of samples with noisy labels can be used for training a supervised classifier. However, the use of distant supervision is limited to tasks with available knowledge databases.

Bootstrapping Compared with the approaches described above, *bootstrapping* can be considered a more effective and adaptable algorithm as it is not required to construct complex patterns and knowledge bases. The bootstrapping methods start with a small number of seed instances or patterns and iteratively enlarge the set of labeled instances [119, 71, 131]. The framework of *Basilisk*, Bootstrapping approach to semantic lexicon induction) [159], has been used for learning new types of knowledge, including subjective nouns [135] and event-related nouns [127]. New types of patterns, even regular expressions, can be learned by using part of this bootstrapping framework [153, 37, 152]. Given a target corpus, the only difference is a new seed list for the task. In some subsequent works, negative samples from other categories have been incorporated in the unsupervised bootstrapping process [100]. For example, to learn lexicons of multiple categories simultaneously, the concept of *mutual exclusion* was incorporated to supervise candidate words or patterns to be extracted for a single category, e.g., *MEB* [19], *WMEB* [101]. [102] improved the above two methods by comparing the similarity drift among iterations. A more effective way of using negative information was presented by the work of [57], which trains contextual classifiers where each classifier decides whether a noun phrase belongs to a semantic category. Other work has changed how they train Basilisk. The idea of ensembling has also been considered [129]. Except the iterative bootstrapping work mentioned above, [60] attempted to learn an embedded frame lexicon using a predicate-argument structure, and [128] used a non-iterative process to learn from tweets, where much non-standard content exists, while keeping part of the evaluation functions in Basilisk. Different from those approaches, the proposed methods in Chapter 3 improve the bootstrapping of semantic lexicons by incorporating unsupervised neural networks into the framework of Basilisk and acquiring knowledge in a totally unsupervised manner.

2.3 Multi-modal Deep Support Vector Data Description

Metric learning The incorporation of negative supervision in Chapter 4 is related to metric learning [35]. Specifically, in our proposed mSVDD with the negative supervision framework, *pseudo*-negative samples are used to calculate the *contrastive* and *triplet* type losses. Through adding these losses to mSVDD, mSVDD can learn discriminative features that keep *target* data points close, while pushing *pseudo*-negative data apart in the feature space. This shares a similar concept with metric learning, such as ϵ-SVM, which minimizes the within-class distances by adding constraints to SVM [24]. In terms of using neural networks to learn features, our methods are more relevant to deep metric learning [51, 64]. Different from Mahalanobis metric learning [166], that finds only a linear transform, recent work on deep metric learning can learn non-linear neural feature mapping, such that the distance of a positive pair is smaller than a negative one [56, 169, 155]. Formally, deep metric learning methods usually define the distance between data points as $d_{ij} = \|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)\|$, where $f(\boldsymbol{x})$ is an embedding of a sample x learned by a neural network. Given three data points a, p, n, where a is an anchor, p is a positive of the same class as the anchor, and n is a negative of a different class, the triplet loss that the deep metric learning optimized is: $\mathcal{L} = max\{0, d(a,p) - d(a,n) + margin\}$. If we assume that the *centers* in mSVDD are anchors, data from the target class are positive, and external samples are negative, the loss of mSVDD with negative supervision can be written in similar formulation.

Others In Section 4.4 of Chapter 4 introduced later, we use these external data to complete auxiliary tasks with *contrastive* and *triplet* losses. One of the choice for *pseudo*-negative data used in our negative supervision is from a publicly available corpus, WikiText-2 [104]. This can be seen as one kind of *weakly supervision* method, which learns classification models from data that are incompletely or scarcely labeled [82, 179]. Since the discriminative information is learned from external data that is not directly related to the original training data in our negative supervision, this also follows the idea from *transfer learning* [118]. [126] proposed a much more related work, where they also incorporated the external image dataset to learn discriminative features for a one-class problem in the computer vision area. As for the difference between our work and [126], the primary purpose of [126] is to learn features for one-class classification with two stages setting: *feature learning* and *k-nearest-neighbor*

classifier using. In contrast, our purpose (Section 4.4) is to design a specific structure for mSVDD such that it can be trained in an end-to-end neural structure. Another related area is *self-supervised learning*, that constructs kinds of *contrastive learning* tasks for unsupervised representation learning [67, 114, 17]. For example, in the contrastive learning framework used in Momentum Contrast (MoCo) [42], they learned similar/dissimilar representations from data that are grouped into pairs by optimizing a so-called InfoNCE loss function [116]. The contrastive loss (Eq. (4.18)) used in our work can be viewed as a simplified InfoNCE loss function.

Chapter 3

AutoEncoder Guided Bootstrapping of Semantic Lexicon

3.1 Introduction

Acquiring large amounts of knowledge for natural language processing tasks can be costly when performed manually. Therefore, there is an increasing need to acquire knowledge, such as semantic lexicons, from raw text corpora. Such acquired knowledge, including the semantic lexicons, is beneficial for numerous application scenarios, e.g., domain-specific dialogue systems [112].

As minimal supervision algorithms, *bootstrapping* methods are efficient tools to explore knowledge from unannotated corpora with increasing text size. Their simple input strategy, a human-defined seed list, gets rid of the needs of complicated human-crafted rules, enormous amounts of annotated training data, which are both time-consuming and costly to construct. Another benefit of bootstrapping is the flexibility for different emerging tasks. Yarowsky showed the effectiveness of *bootstrapping* learning in improving the performance of the task of word sense disambiguation [174]. Then, *bootstrapping* techniques have been conveniently used in a variety of tasks, including relation extraction [12, 1] and semantic lexicon induction [85, 159]. In addition, the most prominent characteristic of this idea, the *iterative* learning process, has the potential to add more new instances as iterations progress.

In this work, we focus on the task of acquiring semantic lexicons using the *bootstrapping* technique. Specifically, we follow the framework of *Basilisk* [159], Bootstrapping approach to semantic lexicon induction, an effective and commonly used bootstrapping approach for constructing semantic lexicons from text corpora. As one

kind of bootstrapping methods, Basilisk inherits the advantages and characteristics of bootstrapping. In Basilisk, it is also not necessary to label training data, and this makes it more efficient and resources-saving. To explore semantic lexicons, all you need is to input an initial *seed list* with a limited number of initial seed words. Thus, it is simple to try the Basilisk algorithm to seek different types of lexicons from a text corpus. The process of Basilisk can be seen from Figure 3.1. Given a seed list, we can start the Basilisk process to explore an unannotated corpus. As the bootstrapping learning process continues, the list of seed words will be expanded iteratively. Basilisk adds not just new words, but also *patterns* indicating the context around the words. Thus, Basilisk iteratively learns two apparently different knowledge of patterns and semantic lexicons, i.e., two types of knowledge will be bootstrapped *alternately* during each iterative process. Another key idea would be *mutual bootstrapping*, where one type of knowledge (lexicon or pattern) is evaluated on the basis of the collective information from another type of knowledge. Therefore, it evaluates and learns two types of knowledge on patterns and lexicons in turn, and they then benefit each other mutually. Three benefits of Basilisk can be summarized as: 1) flexibility, 2) learning iteratively, and 3) mutual bootstrapping of two knowledge.

Basilisk would define a framework for bootstrapping that can be developed for a variety of applications, rather than only a way of bootstrapping. Basilisk has been modified and used in subsequent work, making it possible to bootstrap many types of knowledge. For example, new types of lexicons including subjectivity lexicons and patient polarity verbs (PPVs) can be learned with the same Basilisk algorithm [136, 167]. To induce new lexicons from Twitter, there was the work that dropped the iterative process while keeping the part of its evaluation mechanism [128]. Incorporating supervised learning would be another choice; it explores the idea of inducing a contextual classifier trained on positive instances of one category and negative instances from the others [57].

In this work, we would like to incorporate AutoEncoder to improve the Basilisk bootstrapping approach in an unsupervised manner. The key idea is to use AutoEncoder to guide the two scoring functions: *candidate scoring* and *pattern scoring*, which are used for two fundamental steps of evaluating patterns and candidate instances alternately. AutoEncoder, which is usually realized with a feed forward network with a bottleneck structure and could be used to learn a compact representation of input positive samples [115, 142, 11, 49]. As introduced in Chapter 2, AutoEncoder a reconstruction model which consists of an encoder for computing a representation

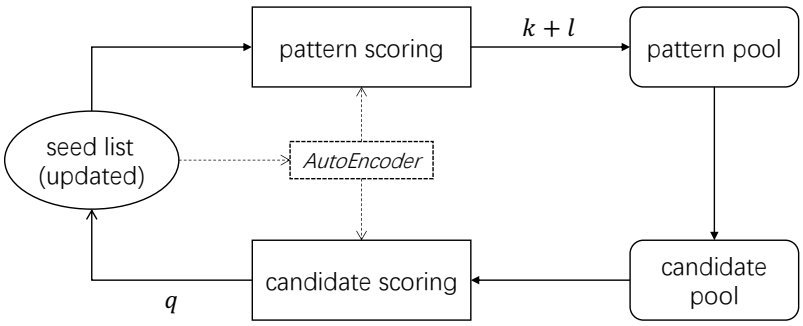


Figure 3.1: Process of Basilisk-like bootstrapping (solid line part) and the proposed AutoEncoder guided bootstrapping method (including dashed line part)

from an input, and a decoder for reconstructing the input from the representation. The parameters involved in the network can be optimized by reducing the loss of *reconstruction* which computes the difference between the reconstructed outputs and the inputs.

Our motivation is that, while we have only a small number of seed (positive) instances, we can utilize these positive samples to complete learning under one-class classification setting. In this setting, we do not rely on any negative samples, obtained through expensive expert knowledge or learning approaches [85, 19, 100]. Furthermore, the learning paradigm is unsupervised such that we can follow the principle advantages of unsupervised bootstrapping. If we train AutoEncoder only with these positive instances generated in bootstrapping iterations, the *reconstruction* score, which measures the similarity between the original input and reconstructed input, can measure how well a candidate is reconstructed by it. In addition, because of the increasing number of positive samples extracted in bootstrapping iterations, AutoEncoder can be updated in an incremental manner. In each evaluation step of bootstrapping, AutoEncoder can be used to score and identify the better candidates. Besides *candidates*, we also attempt to use AutoEncoder to get a new pattern scoring mechanism. This new metric for *pattern scoring* could benefit downstream steps, resulting in boosting the results of a bootstrapping approach based on Basilisk.

Our methods in this chapter can be summarized as follows: 1) We formulate bootstrapping as a special one-class problem. 2) We use AutoEncoder to guide the learning of bootstrapping in a totally unsupervised manner. 3) We improve candidate evaluation by incorporating AutoEncoder, which does not rely on any additional training data. 4)We also present a new function to boost the scoring of patterns such that better

candidates will be pooled in subsequent steps. The performance of candidate scoring can be improved by including our new pattern scoring function.

The rest of this chapter is organized as follows: Section 3.2 introduces the preliminary and background. Section 3.3 describes the proposed methods. The experimental setup and results are presented in Section 3.4. Finally, we summarize this chapter .

3.2 Preliminary

Problem Statement: The objective of this research is to acquire lexicons for a semantic domain from a general corpus. First, consider a text corpus $\mathcal{C}$ and a list $\mathcal{S}$ containing manually crafted seed words (e.g., "*banana,*""*spaghetti,*"...), that are intended to belong to this *food* category. This is the only human input to the entire learning process. With minimal syntactic structure information, we use only a *conjunction* pattern with the form of multiple conjunctive items (n_1, n_2, n_3) (a trigram of noun phrases or nouns). Here, candidate words are selected irrespective of order and can be at any position in (n_1, n_2, n_3). Once a word $n \in \mathcal{S}$ is in (n_1, n_2, n_3), the pattern (n_1, n_2, n_3) will be considered as a candidate pattern in the set $\mathcal{P}$. An example is (*penne*, *spaghetti* , *xyz*), where it is assumed that "*spaghetti*" and "*penne*" are in the seed list and *xyz* denotes a candidate word. If we can find a set of candidates *xyz*, we can evaluate them and add some of them to the seed list and repeat the process.

Background: Our method, much like the previous work (e.g., Basilisk [134, 159]), acquires semantic lexicons in a bootstrapping manner. We follow the general framework and iterative process of Basilisk, illustrated in Algorithm 1 and Figure 3.1.

Prior to the actual iterative process, $\mathcal{S}$, a list of seed instances with a limited number, is inputted to the system. On the basis of this seed list, corresponding contextual patterns are then generated and stored in the set $\mathcal{P}$, where the inputted seed words appear in the corpus. The process starts with *pattern scoring* as shown below:

1. **Pattern Scoring**: patterns in $\mathcal{P}$ are scored and ranked with $RlogF$:

$$RlogF(p) = F/N * log(F), \tag{3.1}$$

where $p \in \mathcal{P}$ denotes a pattern to be evaluated. $\mathcal{N}$ denotes the set of instances extracted by pattern p, and $N = |\mathcal{N}|$ is the total number of extracted instances. $\mathcal{F}$ is a set of correctly extracted words, i.e., $\mathcal{F} = \mathcal{N} \cap \mathcal{S}$, and F computes the

Algorithm 1: Algorithm of Basilisk Bootstrapping

Input: seed list $\mathcal{S}$, corpus $\mathcal{C}$
Output: seed list $\mathcal{S}$ (updated)
Parameter: k, q

1 Generate context patterns and store them in $\mathcal{P}$, score the patterns on the basis of Equation (3.1)
2 Select top-$(k+l)$ patterns to $pattern\_pool$ # l is the iteration index
3 Select all extractions with patterns in $pattern\_pool$ to $candidate\_instance\_pool$
4 Score all candidate instances in $candidate\_instance\_pool$ on the basis of Equation (3.2)
5 Add top-q candidates to seed list $\mathcal{S}$
6 $l = l + 1$ # increment the iteration index
7 Repeat *Steps* 1 *to* 6

cardinality of $\mathcal{F}$. This pattern score essentially represents the productivity of each pattern (the ability to extract candidates).

2. Pattern Selection: on the basis of the score, the top-$(k+l)$ patterns are selected and placed in the $pattern\_pool$.

3. Candidate Pool: all the instances newly extracted by the patterns in the $pattern\_pool$ are put into the $candidate\_instance\_pool$, except for those already in the seed list.

4. **Candidate Scoring**: to evaluate each candidate word c in the $candidate\_instance\_pool$, $AvglogF$ is used as the scoring metric, as defined below,

$$AvglogF(c) = \sum_{i=1}^{P} log_2(F_i + 1)/P, \tag{3.2}$$

where i is used to index pattern p_i. Assuming candidate c is extracted by P patterns, and F_i is the corresponding number of correct instances (as explained in Equation (3.1)), this function calculates the mean value of productivity for the patterns. The most important step is Step 4, where candidate instances are scored. Note that the candidates are evaluated on the basis of all patterns in $\mathcal{P}$ rather than in $pattern\_pool$.

5. Top-q candidate words are added to the seed list $\mathcal{S}$ for the next iteration.

In the work of [159], rather than using $AvglogF$ (Equation (3.2)), the authors first investigated the effect of another candidate scoring function, $AvgF$:

$$AvgF(c) = \sum_{i=1}^{P} F_i/P. \tag{3.3}$$

However, the simple *average* strategy is prone to be biased by one pattern with a large number of extractions F_i. To minimize this problem, $logarithm$ was added to constrain the effect of any single pattern. The improved version, $AvglogF$, as shown in Equation (3.2), was therefore adopted. Even so, if one candidate c is extracted by a small number of high-scored patterns (large $logF$), it is still prone to be overrated. Our preliminary experiments showed that this bias still occurred during candidate estimation. For example, totally unrelated tokens (e.g., "*kinds of*") and non-standard terms (e.g., "*g cone ,* " and "*e.g. (*") were extracted by the same one pattern with a high value of F_i. The latter error of "*g cone ,* ", which may be caused by the preprocessing tools, may have existed even if we had used other text tools. We found that both terms were sometimes ranked higher on the basis of the evaluation by $AvglogF$ or $AvgF$. It is easy to understand that, if the value of P gets smaller while F_i becomes greater, $AvglogF$ of one candidate c would obtain a higher value using Equation (3.3).

Besides the candidate scoring function of Equation (3.3), the pattern scoring function of Equation (3.1) may also cause the bias problem, that persists in the bootstrapping iteration. In Step 1 of evaluating patterns, when the value of F is large, the score of p will also be large using Equation (3.1). According to Steps 2 and 3 in Algorithm 1, high-scored pattern p is more likely to be placed into *pattern_pool* such that extractions from this pattern are put into *candidate_instance_pool*. Then, in Step 4 of evaluating words in *candidate_instance_pool*, the bias problem using Equation (3.3) will arise again.

Another issue caused by the bias problem would be the limited coverage of lexicons. There are several possible reasons: 1) the number of extraction candidates for each pattern is limited, 2) the pattern pool has a limited increase of l in each iteration, and 3) the candidate pool is updated with few high-scored patterns. These limitations may restrict the ability of bootstrapping to explore a broad range of lexicons.

The intuitive approach for alleviating the bias problem is to set a threshold for the number of patterns, i.e., to discard candidate words recalled by less than a specific number of patterns. Although several thresholds were tried in our preliminary

Method	Example extractions
AvglogF	chili pepper, gherkin, roast onion, ..., red pepper, black pepper, ...
NoisyOR	corn, butter, potato, fish, sugar, ...

Table 3.1: Example extractions obtained by *AvglogF* and *NoisyOR*.

experiments, there was no significant improvement when compared with $AvglogF$. Furthermore, setting a good threshold value also has a problem. All in all, we can say that this intuitive method "treats the symptoms rather than curing the disease." The above two types of candidates ("*g cone*" and "*e.g. (*") can essentially be seen as the *noise* in pattern extraction. We can alleviate this bias problem by evaluating candidate words on the basis of collective evidence over a wide range of patterns rather than few high-scored patterns.

We thus turn our attention to the related bootstrapping work. *NoisyOR* is an effective technique used in *Snowball* [1] and in *NOMEN* [85], as introduced in [164, 62]. In contrast to $AvglogF$, which prefers clues from a small number of high-quality patterns, *NoisyOR* considers the evidence on the quantity of patterns.

Here, we attempt to use *NoisyOR* for candidate scoring with the following considerations: 1) the confidence of each pattern, $conf$, is represented in a probability form, and 2) *all* related patterns are considered.

$$NoisyOR(c) = 1 - \prod_{i=1}^{P}(1 - conf(p_i)), \tag{3.4}$$

where p_i is a pattern, and P is the number of patterns from which candidate c is extracted. The confidence of each pattern, $conf$, can be described as $conf(p_i) = F_i/N_i$, similar to Equation (3.1). Given *NoisyOR*, the acceptance of candidate c is based on the backing of as many patterns as possible. This metric prevents the ranking and selection from being easily biased to the lexicon selected by high-confidence patterns, as in the cases with $AvglogF$. In this work, we utilize $NoisyOR$ as an alternative method for evaluating candidates, besides the baseline scoring function $AvglogF$ in Step 4 of Algorithm 1.

***AvglogF vs. NoisyOR*:** The differences in these two candidate scoring mechanisms lie in the tendencies in their *evaluation* and *results*. In the evaluation, *AvglogF* relies on the minority of patterns with higher F values, while *NoisyOR* depends on the majority of patterns with not lower scores. Consequently, the results of exploration also

differ as in the following (See examples in Table 3.1): 1) *AvglogF* mines more deeply, which results in more specific words related to one specific pattern, for example, "*red pepper*" and "*chili pepper*" are connected to "*pepper*", and 2) *NoisyOR* explores more broadly, producing more general words that are extracted by more patterns. These different features motivated us to combine the characteristics of both evaluation methods. We have tried to ensemble these two functions, but did not obtain the expected significant improvement.

3.3 Proposed Methods

3.3.1 Candidate Scoring with AutoEncoder

The above-mentioned candidate scoring methods, $NoisyOR$ and the baseline $AvglogF$, along with other bootstrapping systems, like *WMEB* [101] and *Espresso* [119], evaluate instances by using the information from pattern confidence or co-occurrence between patterns and instances. These patterns can be seen as an *interviewer* who judges candidates, and the above scoring mechanisms seek support from the interviewers in the evaluation step. Another type of knowledge, instances acquired in previous iterations of bootstrapping, can be considered as *peers* or *predecessors*. The previously extracted *peer* words can also be directly used to evaluate the current patterns.

The characteristics of the acquired lexicon in the updating seed list are as follows:

1. Both the initial set of instances and the acquired instances in the seed list can be seen as *positive* samples belonging to the target semantic category.

2. The vocabulary in the seed list is expanded over time.

3. Seed lists from the different initial seed lists will be updated and acquired independently of each other.

To utilize this resource in the bootstrapping, one option would be to train a supervised classifier or regression model with the acquired lexicon. However, this requires "appropriate" *negative* labeled samples, which are hard to obtain in different bootstrapping iterations with a varying initial seed list. One trick is to input a *negative* list as well as a *positive* list [1, 100, 57]. While general negative instances (e.g., "*the*" and "*a*") cannot cover and reflect differences among patterns, too specific instances are only adaptive to a few patterns.

Therefore, our motivation is to evaluate candidate instances only with positive seed instances. On the basis of the above motivation, we utilize a neural model called *AutoEncoder* [59, 49], which can be trained in an unsupervised manner. AutoEncoder is an unsupervised neural network that can complete training by relying only on input positive samples themselves. The architecture of AutoEncoder is based on reconstructing the input *data* at the output end through the pair of an encoder and decoder. The difference between inputs and outputs, represented by the *reconstruction* loss, can be used to measure the quality of the reconstruction. Given AutoEncoder and a seed list, we can take into account the first characteristic of a seed list in which its instances are all positive. With trained AutoEncoder, we then could examine whether a candidate new word is faithful to the original seed list, such that the evaluation step can be completed in a totally unsupervised manner.

The second characteristic of the lexicon is that its size is enlarged by iterations. AutoEncoder can not only boost training in an incremental mode but also acquire adaptability to different iterations. For the different initial seed list, we can also train independent models, such that a learned model can fit its own initial seed list. Thus, AutoEncoder can match our objectives and the three characteristics of the seed list.

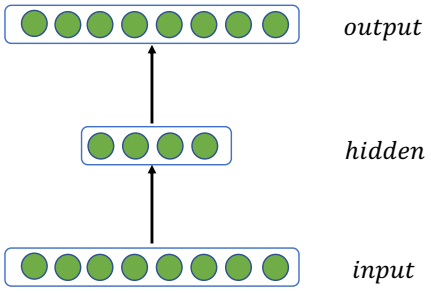


Figure 3.2: Architecture of AutoEncoder, represented as a bottleneck network: the encoder transforms *inputs* into *hidden* states in a low-dimensional space and the decoder reconstructs the inputs from the hidden states.

The architecture of AutoEncoder is shown in Figure 3.2. We first use feature vectors (learned from word embeddings) of instances as the *input* at the bottom. Specifically, an input instance c is transformed to one feature representation by averaging the embeddings of its words. Given an input instance c and its reconstruction output c', we use $r(c)$ and $r(c')$ to denote their feature representations, respectively. The input feature vector $r(c)$ is compressed into a *hidden* state of a smaller number of dimensions. Then, the *output* layer tries to rebuild the input signal from the hidden state.

The network can be optimized by reducing the loss of reconstruction that computes the difference between reconstructed outputs and inputs. After being trained with only positive instances, AutoEncoder can output positive instances with smaller reconstruction loss, which enables distinct features to be acquired from positive instances such that it is hard to reconstruct negative instances. The candidate scoring function with AutoEncoder can be defined as follows:

$$AE(c) = 1 - norm(reconstruct(r(c), r(c'))), \tag{3.5}$$

where output $r(c')$ represents the reconstructed input, and the reconstruction loss is measured by $reconstruct(,)$, for which we chose the mean squared error. We use the $sigmoid$ function as the $norm$ to output a probability value $AE(c)$. We can rank and select the better candidates with larger $AE(c)$ values using Equation (3.5). This value can also be used for the pattern scoring step, that we introduce in Section 3.3.2 below.

3.3.2 Pattern Scoring with AutoEncoder

As shown in Figure 3.1 and Algorithm 1, all candidates come from the candidate pool acquired on the basis of the pattern pool, which is determined during the pattern scoring step. Therefore, if low-quality patterns are selected, credible results are hard to obtain even when there is a better candidate assessment metric. We hypothesize the following:

Hypothesis: *Better pattern evaluation results in better candidate selection, and so the pattern scoring function contributes to improving the performance in candidate selection.*

Therefore, our motivation is to modify the pattern scoring function for better pattern evaluation. Reviewing the first step of Algorithm 1, the score for each pattern p is computed by Equation (3.1). Here, we rewrite it as follows:

$$RlogF(p) = F/N * log(F) = R * logF \quad (Rewriting), \tag{3.6}$$

where $R = F/N$ denotes the "reliability," which computes the correct probability of extracting pattern p, while $logF = log(F)$ measures the potential "productivity." The characteristics of $RlogF$ are:

1. "Reliability": $R = F/N$ assumes that all extracted instances other than previous seed instances are *false positives*.

2. "Productivity": $logF = log(F)$ is not a probability.

3. "Combination": $RlogF$ combines R and $logF$ by a "multiplication."

First, we attempt to incorporate *AutoEncoder* (See Section 3.3.1) to guide the calculation of $R = F/N$. As we know, $\mathcal{N}$ consists of the numbers of correctly extracted seed instances $\mathcal{F}$ and other instances whose correctness is unknown, i.e., $\mathcal{N} = \mathcal{F} \cup Unk$ (Unk is unknown instances extracted by this pattern.). It is ill-advised to regard all the unidentified ones as negative when calculating $R = F/N$, which is the ratio between the number of correctly extracted seed instances F and N. Therefore, for each unknown candidate c in Unk, $1 - AE(c)$ is incorporated to score the probability of the instance being a "*false positive*." The formula of R could be replaced by $autoR$ to evaluate the "reliability" of a pattern:

$$autoR = F/(F + \sum_{c \in Unk} (1 - AE(c))). \tag{3.7}$$

Second, we replace $log(F)$ with a new measure, $L = F/|\mathcal{S}|$, where $|\mathcal{S}|$ denotes the size of the updated seed list $\mathcal{S}$. Now, the "productivity" is represented as L, which is a probability that agrees with the "reliability." Finally, we combine the previous two metrics, $autoR$ and L. Rather than "multiplying" the two metrics, as $RlogF$, a more balanced combination operation called *harmonic mean* can be used:

$$autoRL(p) = 2 * autoR * L/(autoR + L), \tag{3.8}$$

where $autoR$ and L are defined as above. Similar to the F1 measure, the *harmonic mean* is a metric for averaging two measures.

3.3.3 Summarizing Proposed Methods

We summarize our proposed methods in accordance with the motivations presented above. Our methods, as shown in Figure 3.1, utilize the basic framework of Basilisk-like bootstrapping and modify its two key steps.

Candidate Scoring

To evaluate candidate instances, we modify $AvglogF$ (Equation (3.2)) or another baseline $NoisyOR$ (Equation (3.4)), adapted in the other bootstrapping systems, to

AutoEncoder (AE) in the candidate scoring step:

$$AE(c) = 1 - norm(reconstruct(r(c), r(c'))).$$

Pattern Scoring

As described in Section 3.3.2, we substitute $RlogF$ with $autoRL$ and combine the two new probabilistic metrics, $autoR$ and L, that represent "reliability" and "productivity," respectively:

$$autoRL(p) = 2 * autoR * L/(autoR + L).$$

The above formula is used as the new pattern scoring function in Step 1 of bootstrapping.

3.4 Experiments

In this section, we first introduce the data, pre-processing, and evaluation metrics. We then show the experimental results and their analysis. We also perform ablation studies to understand the behavior of each component in our methods.

3.4.1 Data and Preprocessing

The objective of this work is to acquire knowledge in an unsupervised bootstrapping manner from an unannotated text corpus. Generally, there are three kinds of corpora that can be used: 1) A domain specific corpus, such as a biomedical corpus [61] or medical text data [3]. This kind of dataset contains authoritative source of domain information, though it is expensive to obtain it. 2) Social media text data, such as Twitter, have a lot of non-standard language content and are hard to process [128]. 3) A corpus of a general domain (e.g., Wikipedia) contains diverse knowledge and standard language texts. Here, we prefer to acquire semantic lexicons from the general corpus of Wikipedia, rather than the other two corpora, based on the following considerations: 1) This general corpus has a stable text form and diverse coverage. 2) We would like to acquire lexicons for a specific domain from a general corpus and explore other domains in the future. As for the choice of the *food* domain, we thought food is a good topic for a chat dialogue systems because eating food is a daily activity, and there are a variety of foods [72]. Furthermore, there are no comprehensive

standard database for foods unlike movies. So if we try to build a dialogue system in the food domain, we need to create a list of foods. In addition, since the food domain is large [111], we believe a bootstrapping method that works for the food domain will also work for other domains. Since we used a *conjunction* pattern, the dataset was preprocessed by a spaCy2 pipeline, including a tokenizer and a noun phrase chunker [54].

3.4.2 Evaluation Metrics

We evaluated the system performance in terms of Precision, Recall, and F1-measure (F1). Precision is defined as the ratio of correct instances in all the extracted instances. Each extracted instance was manually evaluated by inspecting whether it was correct. For Recall, we calculated two different values with a small and large list of correct instances. The small list with 68 instances consisted of more single-token instances. The large list with 457 instances consisted of more noun phrases. Recall values were the ratio of the number of instances in the lists that can be acquired in the bootstrapping. Therefore, two recall metrics were recorded: Recall (small) and Recall (large). As F1 considers both the precision and the recall, we have two corresponding F1s: F1 (small) and F1 (large).

3.4.3 Experimental Settings

All experiments were performed using ten initial seed instances. In each iteration, top-$(20 + l)$ patterns were added to the pattern pool. The top-5 best candidate instances were then added to the expanded seed list for the next iteration. For the inputs for the AutoEncoder model, we used Glove embeddings (300 dimensions) trained on Common Crawl [123].[1] We transformed each instance to an input vector by averaging the embedding vectors of its words. Each input to AutoEncoder was compressed to a hidden layer of 100 dimensions. Sigmoid was selected as the decoder activation to reconstruct inputs. The mean squared error (MSE) loss was optimized by the Adam algorithm [69]. For each training phase, AutoEncoder was trained with the early stopping strategy with the maximum number of epochs (128). Because bootstrapping approaches can be easily influenced by different input seed instances [101, 133], we evaluated the performance with five different initial seed lists for all compared bootstrapping systems for 20 iterations. For comparison, all the results for the methods

[1] http://nlp.stanford.edu/projects/glove/

1	2	3	4	5
blueberry	apple	pineapple	banana	pancetta
papaya	lasagna	noodle	melon	croquette
barbecue	mutton	steak	spaghetti	white bread
salt pork	bell pepper	avocado	penne	brownie
tofu	egg	wonton	ice cream	cinnamon roll
pad thai	pear	pizza	biscuit	ramen
peking duck	rice	spring roll	beef	asparagus
zucchini	tomato	pumpkin	pork	saury
fry egg	coffee	fry shrimp	garlic	grape
pumpkin pie	donut	hamburger	onion	orange

Table 3.2: Five seed lists

Method	F1 (small)	F1 (large)	Precision	Recall (small)	Recall (large)
$AvglogF$	0.119	0.147	0.840	0.065	0.082
$NoisyOR$	0.202	0.165	0.870	0.115	0.091
AE	**0.254**	**0.280**	**0.878**	**0.150**	**0.172**

Table 3.3: Results for candidate scoring. All metrics are averaged over five independent initial seed lists. Best scores are in bold.

were the average over the five seed lists. All seeds were collected by employees in the organization to which the second author belongs. Table 3.2 lists the seed terms we used.

3.4.4 Experimental Results and Analysis

Candidate Scoring

First, we compared our AutoEncoder-guided scoring function with the two baseline methods: $AvglogF$ and $NoisyOR$. As shown in Table 3.3, $NoisyOR$ outperformed $AvglogF$ in all five metrics. AE, with AutoEncoder, outperforms both the previous baseline systems. As for F1, $AvglogF$ and AE obtained higher scores for F1 (large) than for F1 (small). In contrast, $NoisyOR$ achieved higher scores for F1 (small) than for F1 (large). This may be due to its preference for single-token instances of which the small list consists. $NoisyOR$ evaluates candidates by collecting evidence from a larger range of patterns, and so extracts more general single-token instances. In Figure 3.3, we can track five metrics as bootstrapping progresses. We see that our proposed approach AE significantly outperforms other methods in both F1 metrics over different iterations. This figure also display a continuous phenomenon of bootstrapping

learning, which is the continuing decrease of precision and the increasing trend of recall. In summary, our AutoEncoder-based candidate scoring function improved the overall performance compared with the two baseline methods.

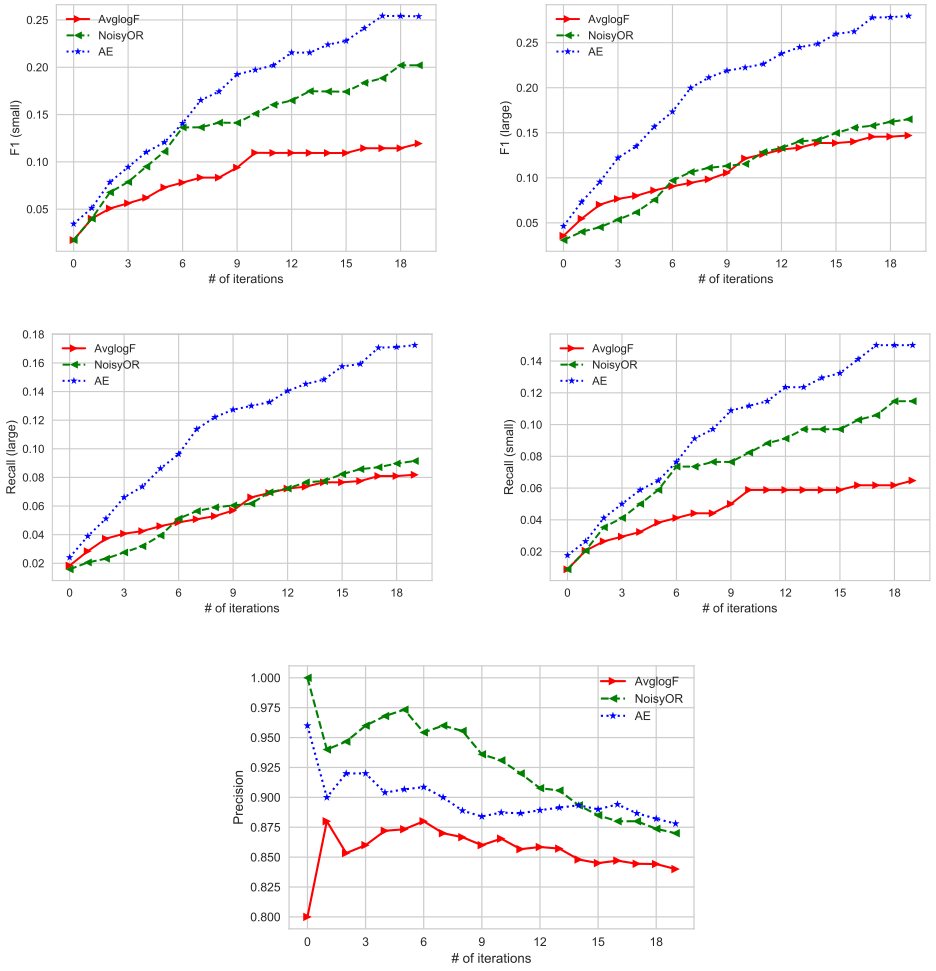


Figure 3.3: Metrics after each bootstrapping iteration. # denotes number.

Pattern Scoring

This subsection investigates the performance of our new pattern scoring function $autoRL$, to validate **Hypothesis** and our motivation in Section 3.3.2. In the experiments, we used the same baseline candidate scoring functions ($AvglogF$ and $NoisyOR$) and modified the pattern evaluation function.

As shown in Table 3.4, $autoRL$ improved the performance, especially for F1 (small), which increased to 0.232. The same scale of improvement (from 0.165 to 0.253) was also achieved for F1 (large) for the method with $NoisyOR$, at the cost of

Method	F1 (small)	F1 (large)	Precision	Recall (small)	Recall (large)
$AvglogF$	0.119	0.147	**0.840**	0.065	0.082
$autoRL + AvglogF$	**0.232**	**0.213**	0.774	**0.138**	**0.124**
$NoisyOR$	0.202	0.165	**0.870**	0.115	0.091
$autoRL + NoisyOR$	**0.259**	**0.253**	0.838	**0.153**	**0.149**

Table 3.4: Results for pattern scoring. $autoRL + AvglogF$ denotes that $autoRL$ is used for pattern evaluation, and candidate scoring still uses $AvglogF$. $autoRL + NoisyOR$ represents $autoRL$ plus $NoisyOR$. Best scores are in bold.

a slight precision drop. Precision was affected, especially for the $AvglogF$ method. In part, this is because of the bias problem, which shows that candidate instances are more likely to be extracted by few high $RlogF$ patterns with large number of extractions $\mathcal{F}$. When we changed $RlogF$ to $autoRL$, more patterns could contribute to the candidate pool. Instead of having to do evaluation in a narrow range, this change brings to the candidate scoring function $AvglogF$ a larger range of candidate words. Thus, this is beneficial to improving recall scores. However, the inclination of $autoRL$ to patterns with fewer extractions caused a decrease of precision in $AvglogF$. For $NoisyOR$, which considers pattern supports as much as possible to alleviate the influence from few patterns, these changes had a larger impact on precision than $AvglogF$. To summarize, Table 3.4 demonstrates that the overall performance of both the baseline methods in candidate scoring could be enhanced by using $autoRL$ in pattern scoring.

Ablation Study: Impact of each part in pattern scoring function

To assess the contribution of each part in our new pattern scoring method, we performed an ablation study. Table 3.5 summarizes the results in F1 scores. In Basilisk's original formula $RlogF(p) = R * logF$, R represents "reliability," and $logF$ measures "productivity." They are combined by the "multiply" operation. Therefore, our $autoRL$ (Equation (3.8)) modifies three parts of $RlogF$. More specifically, we can replace R with $autoR$ for "reliability" and $L = F/|\mathcal{S}|$ for "productivity." Finally, harmonic mean, instead of "multiply," can be used to combine "reliability" and "productivity." In this ablation study, we kept the basic *Basilisk* framework and investigated the contribution of the changes in "productivity," "reliability," and their combination. Because R and $autoR$ are in the form of probability while $logF$ is not, we do not show its score in the *Harmonic* cell.

In Table 3.5, the numbers in the last two columns denote F1-scores for the small

Index	Reliability	Productivity	F1 (small)		F1 (large)	
			Multiply	Harmonic	Multiply	Harmonic
0	R	$logF$	0.119	-	0.147	-
1	$autoR$	$logF$	0.176	-	0.192	-
2	R	L	0.149	0.205	0.147	0.182
3	$autoR$	L	0.151	**0.232**	0.199	**0.213**

Table 3.5: Impact of each part in pattern scoring function. "-" indicates that we did not obtain any result. Bold numbers indicate the best F1 scores. We use the index column for clarity.

and large lists for different combination operations including "multiply" and "harmonic mean," respectively. Index 0 denotes the initial pattern scoring function $RlogF$, while indexes $1-3$ denote three ablation studies. The first adjustment is $autoR$, which is a new reliability metric based on AutoEncoder. It significantly outperformed R, multiplied by $logF$ (index 0). Specifically, F1 (small) increased from 0.119 to 0.176, and F1 (large) increased from 0.147 to 0.192. By changing "productivity" to L, $R * L$ also achieved a better F1 (small) score than $RlogF$, as shown in the third row (index 2). However, it could not boost $logF$ in F1 (large), which remained at 0.147. As for index 3, where both parts were modified, $autoR$ multiplying L obtained a comparable F1 (small) result (0.151) to R multiplying L (0.149) and a significantly higher F1 (large) result (0.199). The last four columns for F1 show that replacing "multiply" with "harmonic mean" also improved performance. The results for "harmonic" were around $0.056 - 0.08$ in F1 (small) and $0.014 - 0.035$ in F1 (large) higher than those for "multiply." Therefore, in the combination operation, harmonic mean is a better choice than multiplying "reliability" and "productivity." Comparing the two adjustment methods, the improvement of $autoR$ over R demonstrates the benefit of AutoEncoder in improving the representation of "reliability." Overall, our proposed pattern scoring method enhanced the performance on the basis of the contribution of the multiple different parts.

Analysis: Impact of the size of the training data for AutoEncoder

We performed an analysis to investigate the effect of the training data size for AutoEncoder. The results are reported in Table 3.6. Here, 100 additional human-crafted positive instances were added to the training set with each initial seed list. We report the influences on F1 (small) and F1 (large) on the basis of the two types of training data: the original data (10 seed instances) and the augmented seed list with extra ex-

Step	Method	Basic_Inst		Aug_Inst	
		F1 (small)	F1 (large)	F1 (small)	F1 (large)
Pattern Scoring	$AvglogF$	0.232	0.213	**0.252**	**0.235**
	$NoisyOR$	**0.259**	**0.253**	0.224	0.221
Candidate Scoring	AE	**0.254**	**0.280**	0.226	0.259

Table 3.6: Impact of the training data size. "Step" indicates in which step we used augmented training data. "Aug_Inst" denotes the augmented data. "Basic_Inst" denotes the basic seed list. Bold numbers denote the **better** scores in each row comparison ("Basic_Inst" vs "Aug_Inst").

amples (110 instances). The first column, "Step," denotes in which step the augmented training data was used among the two steps in Basilisk: pattern scoring and candidate scoring. In "Pattern Scoring," we tested two baseline methods for candidate scoring: $AvglogF$ and $NoisyOR$.

The performance was improved with the augmented data for $AvglogF$. Table 3.6 shows that F1 (small) increased from 0.232 to 0.252 and F1 (large) increased from 0.213 to 0.235 when the augmented data was used. In contrast to $AvglogF$, it is more beneficial to train AutoEncoder only with Basic_Inst than with Aug_Inst in using with $NoisyOR$. The performances for $NoisyOR + AE$ degraded when more training instances were used, as shown in the second row from the bottom. This degradation makes us consider whether adding much more training examples ($110 > 10$) will offset the influence of different initial seeds and the meaning of increasing new instances in bootstrapping [102]. Random selection of augmented data might cause a biased set of training instances, that resulted in a performance degradation. Additionally, in the original setting, the five seed lists and the corresponding AutoEncoder were trained independently. Here, we directly add Aug_Inst for *all* seed lists to train. This might dismiss the adaptability of AutoEncoder to each independent initial seed list, as shown in the third characteristic of a seed list (See Section 3.3.1).

Case Study

Here, we present successful and erroneous example outputs for a case study in Table 3.7. In each column, seven good and three bad example words extracted by one of the approaches are listed from top to bottom. As mentioned in Section 3.2, the results of $AvglogF$ and $NoisyOR$ were quite different. $AvglogF$ tended to choose related noun phrases from few patterns, while $NoisyOR$ preferred to explore more general words in the candidate pool, which can be observed from the columns of $AvglogF$

Step	Candidate Scoring			Pattern Scoring	
Methods	$AvglogF$	$NoisyOR$	AE	$autoRL + AvglogF$	$autoRL + NoisyOR$
good cases	green pepper	lychee	food	honey mustard sauce	milk
	sweet melon	rum	fish	toasted rice	seafood
	chinese cabbage	cashew	potato	melon soda	chocolate
	musk melon	strawberry	tomato sauce	fry chicken	wine
	watermelon	breadfruit	instant noodle	mint ice cream	salt
	chili pepper	cheese	chicken masala	mince beef	cheese
	kesar mango	pomegranate	roast onion	green pepper	sugar cane
bad cases	production Ŏ13	indigo	peach tree	ingredient list	wood
	marigold petal	gold	a variety	woolen product	silk
	five kind	flower	other plant	other field crop	tobacco

Table 3.7: Sample results of different methods.

and $NoisyOR$. Even though $NoisyOR$ achieved higher accuracy than $AvglogF$, its extractions have two disadvantages: 1) extractions (e.g., "*cashew*") did not have much semantic relatedness with its seed list, and 2) it could not extract more specific words, (e.g., "*chinese cabbage*," "*chili pepper*"), as $AvglogF$ did. If we had used the method AE, which was introduced in Section 3.3.1, the results could have been more balanced by taking both general (e.g., "*food*," "*fish*") and specific words (e.g., "*instant noodle*," "*tomato sauce*") into account.

Next, when $autoRL$ was incorporated into the step of pattern scoring, both of $AvglogF$ and $NoisyOR$ have benefited from the diverse candidate pool. From the column of $autoRL + AvglogF$, it can be observed that $AvglogF$ obtained from a large range of words including stable diet ("*toasted rice*"), drinks ("*melon soda*"), and meat ("*mince beef*," "*fry chicken*"), rather than being restricted to the small vocabulary of "*melon*" and "*pepper*." The last column shows a similar effect for $NoisyOR$. Even though $autoRL + NoisyOR$ still extracted as many general words as the column for

$NoisyOR$, the new pattern scoring step $autoRL$ enabled $NoisyOR$ to choose from a large vocabulary which was related to the food domain. For example, it contains drink-related tokens ("*milk*," "*wine*"), seasonings ("*salt*") and specific tokens ("*sugar cane*").

Lastly, we summarize the erroneous sample outputs in the bottom of Table 3.7. The errors caused by pre-processing tools may appear in various methods, e.g., "*five kind*," "*a variety*", "*production 0̆13*". More important errors would be erroneous tokens extracted by different methods. As for $AvglogF$, its erroneous examples (e.g., "*e.g. (*", "*marigold petal*") were caused by the bias problem and its *average* strategy, as mentioned in Section 3.2. For $autoRL + AvglogF$, similar errors to $AvglogF$ may still exist, e.g.,"*woolen product*", which was also extracted by few high-scored patterns when using the *average* strategy of $AvglogF$. As for AE, an example of "*peach tree*" has semantic relatedness with several fruits (e.g., "*orange*," "*blueberry*") obtained in the previous bootstrapping iterations, while it can be more likely to be in the domain of *plants* rather than *food*. "*Wood*" and "*gold*" are similar erroneous examples in the methods of $NoisyOR$ and $autoRL + NoisyOR$, where both words were easily extracted by more patterns when $NoisyOR$ was used.

3.5 Summary of This Chapter

In this chapter, we presented methods for improving the bootstrapping of a semantic lexicon that uses AutoEncoder to better evaluate candidate instances. The experimental results, including ablation studies, validated the effectiveness of the proposed methods. By training AutoEncoder with the updated seed lists, we could provide a better candidate scoring function. Additionally, a more balanced pattern evaluation function, guided by AutoEncoder, also improved the overall performance. This verified the hypothesis that a better candidate pool would be beneficial.

Chapter 4

One-class Text Classification with Multi-modal Deep Support Vector Data Description

In this chapter, we present a generic *boundary*-based one-class classification framework for text data, multi-modal deep support vector data description, called mSVDD.

4.1 Introduction

As have introduced in Chapter 2, one-class support vector machines (OC-SVM) [149] and support vector data description (SVDD) [158] are *boundary*-based one-class methods [157]. [140] introduced deep support vector data description (deep SVDD), a fully unsupervised method for deep one-class classification for image data. Deep SVDD learns to extract the common factors of target training samples with a neural network to minimize the radius of a hypersphere that encloses the network representations of the data. The learned hypersphere, with a center $\boldsymbol{c}$ and a neural feature transformer $\phi(\boldsymbol{x})$, can be an end-to-end feature learning and one-class classification model. Another piece of work would be one-class neural networks (OC-NN) [14], which extends OC-SVM to an end-to-end neural architecture. Instead of a hypersphere in deep SVDD, OC-NN builds a hyperplane on features learned by a feed-forward neural network.

Target data samples may have distinctive distributions that are located in different regions. Therefore, uni-modal deep SVDD with one hypersphere may not be enough

to describe the target samples. In this work, we extend deep SVDD to multiple modes, where each mode describes the target samples from a distinctive aspect. Given our multi-modal deep SVDD, mSVDD in short, we can create an ensemble set of hyper-spheres with different centers to build a better one-class model. [29] proposed deep multi-sphere SVDD (DMSVDD), a similar but different work from ours. Another recent work, context vector data description (CVDD), proposed by [141], fully uses word embedding knowledge and a neural network structure to process one-class text classification problems. We will also discuss their relationships compare them in the experiments.

In one-class classification, only samples from the target class are available for training, while the model needs to discriminate between the target class and other classes in testing. Due to the unavailability of training samples from negative classes, it is hard for the one-class models to learn effective discrimination information, especially for mSVDD with a multi-layer neural structure. In this study, we also propose an architecture for improving the discrimination ability of mSVDD by incorporating *negative supervision*. Specifically, we define two kinds of losses, *contrastive* and *triplet*, for joint training with the objective function of mSVDD, which is expected to enhance the discriminative power of mSVDD.

In summary, the main contributions of this work are as follows. 1) We propose a general one-class neural learning framework, called mSVDD, to extend the uni-modal deep SVDD to end-to-end multi-modal. 2) We also prove that four one-class models, deep SVDD, CVDD, DMSVDD, and OC-NN, are all special cases of the mSVDD model. 3) We propose two approaches for effectively incorporating negative supervision information to improve the performance of the proposed mSVDD.

The remainder of this chapter is structured as follows: Section 4.2 introduces notations and baseline models. Section 4.3 describes the proposed mSVDD. Section 4.4 describes the combinatorial use of mSVDD and negative supervision. We then analyze evaluation results by comparing mSVDD and the baseline models in Section 4.5. Finally, we conclude this chapter.

4.2 Preliminary

We have introduced the details of SVDD [158] and OC-SVM [149] in Chapter 2. Now, we describe their extensions, deep SVDD [140] and OC-NN [14].

4.2.1 Deep Support Vector Data Description

Deep support vector data description (DSVDD) [140] is an end-to-end deep neural model that not only optimizes the SVDD objective loss but also learns a neural feature transformation. Given target training samples $T = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, deep SVDD first transforms instance $\boldsymbol{x}$ into a data point of the output feature space with ϕ, which is a multi-layer neural network of $L \in \mathbb{N}$ layers with parameters $\mathcal{W} = \{\boldsymbol{W}^1, ..., \boldsymbol{W}^L\}$. Deep SVDD defines two kinds of loss functions:

Soft-boundary deep SVDD :

$$\frac{1}{\nu n}\sum_i max(0, \|\phi(\boldsymbol{x}_i; \mathcal{W}) - \boldsymbol{c}\|_2^2 - R^2) + R^2 + \frac{\lambda}{2}\sum_l \|\boldsymbol{W}^l\|_F^2 \tag{4.1}$$

The first penalty term is for samples lying outside the sphere, i.e., when the distance of $\boldsymbol{x}_i$ to the center, $\|\phi(\boldsymbol{x}_i; \mathcal{W}) - \boldsymbol{c}\|_2^2$, is greater than radius R after the transformation by network ϕ. The above loss also regularizes the radius and neural weight parameters in the second term. As with SVDD, parameter $\nu \in (0, 1]$ adjusts the tradeoff between the radius of the hypersphere and the points outside the hypersphere.

[149] proved that, in single-class classification, ν is the upper bound of the fraction of anomalies, and the lower bound of the fraction of training samples being anomalies or on the optimal boundary. [140] proved that this ν-property still holds for uni-modal soft-boundary deep SVDD.

Another simplified objective that minimizes the mean distance of all positive training samples to the center, the *one-class* form, can be defined as follows:

One-class deep SVDD (simplified form):

$$\frac{1}{n}\sum_i \|\phi(\boldsymbol{x}_i; \mathcal{W}) - \boldsymbol{c}\|_2^2 + \frac{\lambda}{2}\sum_l \|\boldsymbol{W}^l\|_F^2 \tag{4.2}$$

Here, we can rewrite both the above in a unified form:

$$\mathcal{L}_{DSVDD} = C\sum_i [\|\phi(\boldsymbol{x}_i; \mathcal{W}) - \boldsymbol{c}\|_2^2 - \beta]_+ + \beta + \frac{\lambda}{2}\sum_l \|\boldsymbol{W}^l\|_F^2, \tag{4.3}$$

where $[\cdot]_+ = max\{0, \cdot\}$, $\beta \in \{0, R^2\}$ and regularization parameter $C \in \{\frac{1}{n}, \frac{1}{\nu n}\}$ correspond to the two types of forms.

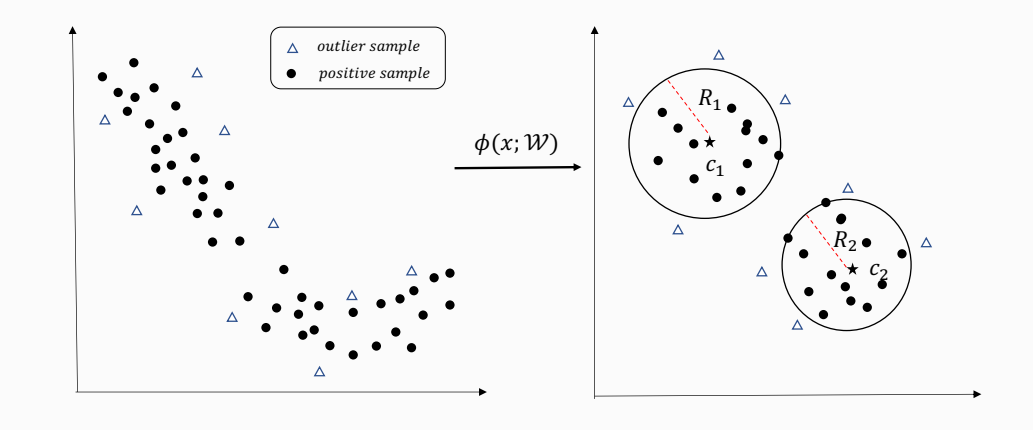


Figure 4.1: mSVDD with two modes. ϕ is a neural network. Fivestars denote centers, and black points denote positive target samples, while triangles denote negative outliers that need to be rejected by hyperspheres.

4.2.2 One-class Neural Networks

[14] proposed one-class neural networks (OC-NN), which extends OC-SVM to a neural architecture. As with deep SVDD, OC-NN also uses an end-to-end neural network to perform feature learning and to optimize a OC-SVM equivalent loss objective function. Given a training set $T = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ and an L-layer neural network ϕ with parameter $\mathcal{W}$, the objective of OC-NN is:

$$\frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{\nu n}\sum_i max(0, \rho - \boldsymbol{w}^T\phi(\boldsymbol{x}_i;\mathcal{W})) - \rho + \frac{\lambda}{2}\sum_l \|\boldsymbol{W}^l\|_F^2, \tag{4.4}$$

where the key idea is to use ϕ to learn the feature transformation for each instance $\boldsymbol{x}$. Similar to soft-boundary deep SVDD (Eq. (4.1)), OC-NN also uses ν to regularize the loss.

We will discuss the relationship between deep SVDD and OC-NN in Section 4.3.3.

4.3 Multi-modal Deep SVDD

In this section, we present our mSVDD, a method for deep one-class classification. Unlike a uni-modal model with a hypersphere, mSVDD uses a set of hyperspheres to describe target class data and to reject samples from negative classes. Figure 4.1 shows the general idea of mSVDD with two modes. Consider that we have m modes, each of which is described by a hypersphere $\boldsymbol{M}_j$ with center $\boldsymbol{c}_j$ and radius R_j; mSVDD uses each $\boldsymbol{M}_j$ to describe a distinctive aspect of the target class and then ensemble

them. This ensembled deep mSVDD model can provide better descriptions for the target data.

As with deep SVDD, given target training samples $T = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, mSVDD first transforms instance $\boldsymbol{x}$ into a data point of the output feature space with ϕ, where ϕ is a deep neural network of $L \in \mathbb{N}$ layers with parameters $\mathcal{W} = \{\boldsymbol{W}^1, ..., \boldsymbol{W}^L\}$. In contrast to deep SVDD, mSVDD uses m hyperspheres to include almost all of the target data with the minimum radii, i.e., $\frac{1}{m}\sum_{j=1}^{m} R_j^2$. The objective of kernel SVDD and deep SVDD is to minimize the volume of a data-enclosing hypersphere in feature space that is represented by center $\boldsymbol{c}$ and radius R. Therefore, to minimize the volume, deep SVDD punishes points lying outside the sphere, i.e., if the distance of $\boldsymbol{x}$ to the center $\boldsymbol{c}$, $\|\phi(\boldsymbol{x}; \mathcal{W}) - \boldsymbol{c}\|_2^2$, is greater than radius R. Since we have a set of hyperspheres $M = \{\boldsymbol{M}_1, ..., \boldsymbol{M}_m\}$, one choice would be to punish $\boldsymbol{x}$ with respect to each hypersphere by adding $\sum_j \max\left(0, \|\phi\left(\boldsymbol{x}; \mathcal{W}\right) - \boldsymbol{c}_j\|_2^2 - R_j^2\right)$ to the loss function. However, the above penalty term does not take into account the different influences from multiple hyperspheres together. Therefore, we would like to take another ensembled constraint by incorporating attention mechanism. Attention mechanism has been adapted to various tasks, allowing models to learn the weights of focusing on different contexts [93]. Thus, it is naturally to incorporate this mechanism to assign a weight for each sample to each center. Given non-negative *attention weight* α_{ij} for $\boldsymbol{x}_i$ to each $\boldsymbol{M}_j$, the penalty term can be computed as the weighted average over m constraints. Now, only one ensembled constraint is required, i.e., *the sum of radii is greater than the sum distance to the center*. Formally, we can define our mSVDD objective as follows:

$$\mathcal{L}_{\text{soft-mSVDD}} = \frac{1}{\nu n}\sum_i \max(0, \sum_j \alpha_{ij}(\|\phi\left(\boldsymbol{x}_i; \mathcal{W}\right) - \boldsymbol{c}_j\|_2^2 - R_j^2)) + \frac{1}{m}\sum_j R_j^2 + \frac{\lambda}{2}\sum_l \|\boldsymbol{W}^l\|_F^2, \quad (4.5)$$

where $\frac{1}{m}\sum_{j=1}^{m} R_j^2$ is the regularization term for radii from all m hyperspheres to get a closer boundary around the target data. This form can be seen as mSVDD with weighted soft-boundary constraints, which we call *soft-boundary mSVDD*.

Although the ν-property, mentioned in Section 4.2.1, does not hold true for our multi-modal case as it is in general, it is still true when the attention weight α_{ij} is constant for different hyperspheres. This will give us an intuition on the role of ν.

Proposition 1. *The ν-property holds if we set equal attention weight to each hypersphere:*

i. ν is an upper bound for the fraction of outlier samples.

ii. ν is a lower bound for the fraction of training samples being rejected or on the optimal boundary.

Proof. **Ad (i).** For each training instance $\boldsymbol{x}_i$, its loss function is defined as *hinge-loss*: $l(f(\boldsymbol{x}_i)) = \max\{0, \sum_j \alpha_{ij} \left(\|\phi(\boldsymbol{x}_i; \mathcal{W}) - \boldsymbol{c}_j\|^2 - R_j^2\right)\}$, where f is the model with parameters. Let us define $d_i = \sum_j \alpha_{ij} d(\boldsymbol{x}_i, \boldsymbol{c}_j)$. Assume $\alpha_{ij} = 1/m$, we have $d_i = \frac{1}{m}\sum_j d(\boldsymbol{x}_i, \boldsymbol{c}_j)$. We also define $R_s = \frac{1}{m}\sum_j R_j^2$. And W.L.O.G, we also assume $d_1 \leqslant ... \leqslant d_n$ which means d_n is n-th farthest sum distance. The number of outliers is given by $n_{out} = |\{i | d_i > R_s\}|$. Rewrite the objective of soft-boundary deep mSVDD (Eq. (4.5)) as:

$$\mathcal{L}_{\text{soft-mSVDD}} = R_s - \frac{n_{out}}{\nu n} R_s = (1 - \frac{n_{out}}{\nu n}) R_s \tag{4.6}$$

Since the objective of mSVDD is to get a minimum R_s, therefore $1 - \frac{n_{out}}{\nu n}$ should be positive, Thus, $n_{out} \leqslant \nu n$ must hold in the training. It implies that at most νn outliers should be rejected.

Ad (ii). The optimal R_s^* has to hold the inequality $n_{out} \leqslant \nu n$. If $R_s^* >= d_n$, then n_{out} takes the minimum value of 0 which means the boundary includes all the samples. Since n_{out} is increased as long as R_s decreased. If n_{out} take the maximum value of νn under condition **(i)**, we can have the minimal $R_s^* = d_{i*}$, where $i* = n - n_{out}$ means d_{i*} is $(n - n_{out})$-th farthest distance. We define $\{\boldsymbol{x}_i | d_i \geqslant R_s^*\}$ is the set of training samples being rejected ($d_i > R_s^*$) or on the optimal boundary ($d_i = R_s^*$). Then we have inequality: $|\{\boldsymbol{x}_i | d_i \geqslant R_s^*\}| = |\{\boldsymbol{x}_i | d_i > R_s^*\} \cup \{\boldsymbol{x}_i | d_i = R_s^*\}| \geqslant n_{out} + 1 \geqslant \nu n$. This implies that at least νn samples being rejected or just on the optimal boundary. □

Proposition 1 and its proof refer to works [140, 16, 150].

4.3.1 One-class mSVDD (Simplified Form)

As in deep SVDD, we also have the simplified form and called: *one-class*[1] *mSVDD*. If we assume that the majority of the training data is not anomalous, then the radius

[1]The term *"one-class"* is used following [140]. Note that most of the models discussed in this theis are one-class models.

can be ignored and we can define the simplified mSVDD as follows:

$$\mathcal{L}_{\text{oc-mSVDD}} = \frac{1}{n}\sum_{i}\sum_{j}\alpha_{ij}\|\phi\left(\boldsymbol{x}_i;\mathcal{W}\right)-\boldsymbol{c}_j\|_2^2+\frac{\lambda}{2}\sum_{l}\|\boldsymbol{W}^l\|_F^2, \tag{4.7}$$

where the attention weight α_{ij} will be kept, while the penalty of radius R is deleted.

4.3.2 Unified Form of mSVDD

We can write the two variants of mSVDD (i.e., *soft-boundary* mSVDD and simplified *one-class* mSVDD) in a unified form:

$$\mathcal{L}_{\text{mSVDD}} = C\sum_{i}[\sum_{j}\alpha_{ij}\left(\|\phi\left(\boldsymbol{x}_i;\mathcal{W}\right)-\boldsymbol{c}_j\|_2^2-\beta_j\right)]_+ + \frac{1}{m}\sum_{j}\beta_j+\frac{\lambda}{2}\sum_{l}\|\boldsymbol{W}^l\|_F^2, \tag{4.8}$$

where $\beta_j \in \{0, R_j^2\}$. $\beta_j = 0$ corresponds to simplified one-class mSVDD, and $\beta_j = R_j^2$ corresponds to soft-boundary mSVDD. For $\boldsymbol{x}_i$ to the j-th hypersphere, attention weight α_{ij} should be inversely proportional to its distance to center $\boldsymbol{c}_j$. Thus, we define:

$$\alpha_{ij} = \frac{\exp(d(\boldsymbol{x}_i,\boldsymbol{c}_j)/\delta))}{\sum_{k=1}^{m}\exp(d(\boldsymbol{x}_i,\boldsymbol{c}_k)/\delta)} = \frac{\exp(\|\phi\left(\boldsymbol{x}_i;\mathcal{W}\right)-\boldsymbol{c}_j\|_2^2/\delta)}{\sum_{k=1}^{m}\exp(\|\phi\left(\boldsymbol{x}_i;\mathcal{W}\right)-\boldsymbol{c}_k\|_2^2/\delta)} \tag{4.9}$$

where $\delta < 0$ is a temperature hyperparameter.

4.3.3 Relationship between mSVDD and Other Models

In this subsection, we would like to discuss the relationships between mSVDD and other related models. First, it will help us to understand the differences, and more importantly the common characteristics of deep SVDD-related approaches. Second, these discussions would give us a unified view on the development of deep SVDD-related approaches in a limited range. Further, it also provides possible future directions. For example, if one model obtained improvements from certain method, e.g., negative supervision introduced later, this may motivate other deep SVDD-related models.

Relationship between mSVDD and Uni-modal Deep SVDD

Their relationship is obvious and can be summarized by the following proposition.

Proposition 2. *Deep SVDD is a special case of the unified form of mSVDD with one hypersphere used.*

Proof. Obviously, mSVDD becomes uni-modal [140] if we use only one hypersphere, i.e., $m = 1$. □

Relationship between mSVDD and CVDD

CVDD [141] is a one-class model for text data. In CVDD, each training sample x_i (i.e., a text) is represented by r self-attention feature vectors $S_i = (\boldsymbol{s}_{i1}, ..., \boldsymbol{s}_{ir})$ [86]. CVDD uses a group of r *context vectors* $C = (\boldsymbol{c}_1, ..., \boldsymbol{c}_r)$ to describe the target one-class data, where $\boldsymbol{c}_k \in \mathbb{R}^p$. CVDD tries to reduce the one-to-one *reconstruction* distance between feature vectors S_i and context vectors C. The loss can be defined as:

$$\mathcal{L}_{CVDD} = \frac{1}{n} \sum_i \sum_k \sigma_{ik} d(\boldsymbol{c}_k, \boldsymbol{s}_{ik}), \tag{4.10}$$

where $d(\boldsymbol{c}_k, \boldsymbol{s}_{ik})$ computes the distance, and σ_{ik} denotes the attention weight. The following proposition implies the close connection between two to learn one-class text problem.

Proposition 3. *CVDD is a very special case of one-class mSVDD when mSVDD is applied to text-based tasks under certain conditions.*

Proof. W.L.O.G., rewrite the loss function of one-class mSVDD in a simplified form for each sample as follows:

$$\begin{aligned} \mathcal{L}_{mSVDD} &= \frac{1}{n} \sum_i \sum_j \alpha_{ij} \|\phi_j(\boldsymbol{x}_i; \mathcal{W}) - \boldsymbol{c}_j\|^2 \\ &= \frac{1}{n} \sum_i \sum_j \sigma_{ij} d(\boldsymbol{x}_i, \boldsymbol{c}_j) \approx \mathcal{L}_{CVDD}, \end{aligned}$$

where we drop the regularization terms for weights of ϕ and radii, and set $m = r$, $\sigma_{ij} = \alpha_{ij}$, $d(\boldsymbol{x}_i, \boldsymbol{c}_j) = \|\phi_j(\boldsymbol{x}_i; \mathcal{W}) - \boldsymbol{c}_j\|^2$. $\phi(\boldsymbol{x}_i; \mathcal{W})$ has to be a self-attention neural model, $\phi_j(\boldsymbol{x}_i; \mathcal{W})$ is the $j-$th feature vector of sample $\boldsymbol{x}_i$, and $\boldsymbol{c}_j$ is the $j-$th context vector of target samples. Now the loss functions of CVDD and one-class mSVDD are almost the same. □

As stated before, CVDD uses a different multi-head structure for text feature learning, while the common point would be both adopt the simplified form of mSVDD loss.

In the experiments, we will show that the proposed negative supervision for mSVDD (Section 4.4) can also be used for CVDD due to the similarity of both.

Relationship between mSVDD and DMSVDD

DMSVDD [29] also uses multi-hyperspheres to extend SVDD. The loss function of DMSVDD is as follows:

$$\mathcal{L}_{DMSVDD} = \frac{1}{\nu n}\sum_{i}[\|\phi(\boldsymbol{x}_i;\mathcal{W}) - \boldsymbol{c}_{i^*}\|_2^2 - R_{i^*}^2]_+ + \frac{1}{K}\sum_{k} R_k^2 + \frac{\lambda}{2}\sum_{l}\|\boldsymbol{W}^l\|_F^2, \tag{4.11}$$

where K is the number of hyperspheres[2], $\boldsymbol{c}_{i^*}$ is the *nearest center* of sample $\boldsymbol{x}_i$ and R_{i^*} is its radius.

Proposition 4. *DMSVDD can be seen as a hard-version of soft-boundary mSVDD if we set the attention weight in some way.*

Proof. In the calculation of the attention weight for mSVDD with Eq. (4.9) $\alpha_{ij} = \frac{\exp(d(\boldsymbol{x}_i,\boldsymbol{c}_j/\delta))}{\sum_{k=1}^{m}\exp(d(\boldsymbol{x}_i,\boldsymbol{c}_k)/\delta)}$, the temperature parameter δ can influence the assignment of center c_k. If we set $\delta \to 0^-$, the above formula acts as the *argmin* operation.[3] In this case, $\alpha_{ii*} = 1$ if $i* = \arg\min_{k=1,\ldots,K} d(\boldsymbol{x}_i, \boldsymbol{c}_k)$, and 0 otherwise. Now, we can get the form of DMSVDD from soft-boundary mSVDD (Eq. (4.5)) through the adjustment of the attention weight. Therefore, we can prove that DMSVDD is also a special case of mSVDD. □

The above relation illustrates key difference between them: DMSVDD puts value on one hypersphere with the largest weight.

Relationship between mSVDD and OC-NN

As mentioned, OC-NN and deep SVDD [14, 140] extend one-class SVM and SVDD [149, 158] to their neural network structures, respectively. [157] has discussed and proved the equivalence of OC-SVM and SVDD. Therefore, in this subsection, we discuss the relationships among the three neural models: OC-NN, deep SVDD and mSVDD. Since deep SVDD is the uni-modal version of mSVDD (See Proposition 2), we focus

[2]In DMSVDD, K changes dynamically. However, we ignore this difference and focus on the comparison of the models.

[3]δ approaches 0 from the negative side.

on the discussion of the relationship between mSVDD and OC-NN. We provide their relationship in the following proposition.

Proposition 5. *i.) OC-NN can be extended to the multi-modal version. ii.) Soft-boundary mSVDD can be seen as multi-modal OC-NN under certain conditions.*

Proof. **Ad (i).** First, we show that OC-NN can be extended to its multi-modal version. OC-NN uses one hyperplane to separate the target data from the origin. Actually we can also use a set of hyperplanes to separate target class data in the feature space, which is a similar idea to mSVDD. We give the objective function of multi-modal OC-NN as follows:

$$\mathcal{L}_{mOCNN} = \frac{1}{\nu n}\sum_i \max(0, \sum_j \alpha_{ij}(\rho_j - \boldsymbol{w}_j^T\phi(\boldsymbol{x}_i;\mathcal{W}))) + \frac{1}{m}\sum_j \|\boldsymbol{w}_j\|_2^2 - \frac{1}{m}\sum_j \rho_j + \frac{\lambda}{2}\sum_l \|\boldsymbol{W}^l\|_F^2, \quad (4.12)$$

where $\boldsymbol{w}_j$ and ρ_j denote the norm perpendicular and the margin of the j-th hyperplane in the neural feature space, respectively. For one data sample $\boldsymbol{x}_i$, the ensembled constraint term, $\max(0, \sum_j \alpha_{ij}(\rho_j - \boldsymbol{w}_j^T\phi(\boldsymbol{x}_i;\mathcal{W}))$, denotes that *the sum of margins should be less than the sum of x_i's distances to the origin*. The definition of weight α is the same as in mSVDD (Eq. (4.9)).

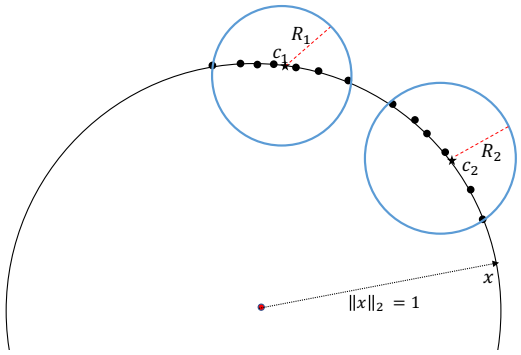
Ad (ii). Then, we prove the equivalence of soft-boundary mSVDD and multi-modal OC-NN. For clarity, we use $\boldsymbol{f}_i = \phi(\boldsymbol{x}_i;\mathcal{W})$ to denote the feature vector of $\boldsymbol{x}_i$. And we also assume the feature vector $\boldsymbol{f}$ is normalized ($\hat{\boldsymbol{f}} = \boldsymbol{f}/\sqrt{\sum |f_i|^2}$), so does $\boldsymbol{c}$. As shown in Fig. 4.2, in this case, all the data representations and centers are mapped onto an unit hypersphere. In this case, a simplified formulation for the loss of mSVDD (Eq. (4.5)) with normalized $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{c}}$ would be:

$$\mathcal{L}_{mSVDD} = \frac{1}{\nu n}\sum_i \max(0, \sum_j \alpha_{ij}(\|\hat{\boldsymbol{f}}_i - \hat{\boldsymbol{c}}_j\|_2^2 - R_j^2)) + \frac{1}{m}\sum_j R_j^2, \quad (4.13)$$

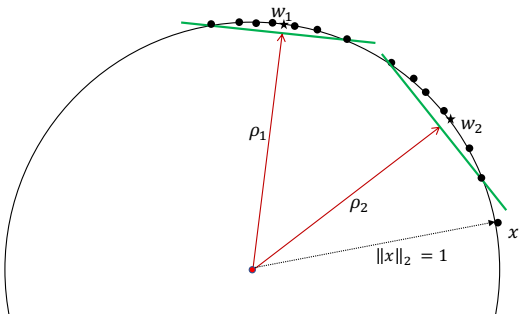
We further transform Eq. (4.13) by using the relation $\|\hat{\boldsymbol{f}}_i - \hat{\boldsymbol{c}}_j\|_2^2 = \|\hat{\boldsymbol{f}}_i\|_2^2 - 2\hat{\boldsymbol{f}}_i^T\hat{\boldsymbol{c}}_j + \|\hat{\boldsymbol{c}}_j\|_2^2 = 2 - 2\hat{\boldsymbol{f}}_i^T\hat{\boldsymbol{c}}_j$,

$$
\begin{aligned}
\mathcal{L}_{mSVDD} &= \frac{1}{\nu n}\sum_i \max(0, \sum_j \alpha_{ij}(2 - 2\hat{\boldsymbol{f}}_i{}^T\hat{\boldsymbol{c}}_j - R_j^2)) + \frac{1}{m}\sum_j R_j^2 \\
&= \frac{1}{\nu n}\sum_i \max(0, \sum_j \alpha_{ij} 2(\underbrace{1 - \frac{1}{2}R_j^2}_{\text{redefine:}\rho'_j = 1-\frac{1}{2}R_j^2} - \hat{\boldsymbol{c}}_j{}^T\hat{\boldsymbol{f}}_i)) + \frac{1}{m}\sum_j \underbrace{R_j^2}_{R_j^2 = 2-2\rho'_j} \\
&= \frac{2}{\nu n}\sum_i \max(0, \sum_j \alpha_{ij}(\rho'_j - \underbrace{\hat{\boldsymbol{c}}_j{}^T}_{\text{redefine:}\hat{\boldsymbol{w}}_j = \hat{\boldsymbol{c}}_j}\hat{\boldsymbol{f}}_i)) + \frac{2}{m}\sum_j (1 - \rho'_j) \\
&= 2\left(\frac{1}{\nu n}\sum_i \max(0, \sum_j \alpha_{ij}(\rho'_j - \hat{\boldsymbol{w}}_j{}^T\hat{\boldsymbol{f}}_i)) + \frac{1}{m}\sum_j \|\hat{\boldsymbol{w}}_j\|_2^2 - \frac{1}{m}\sum_j \rho'_j\right) \\
&= 2\mathcal{L}_{mOCNN}, \qquad (4.14)
\end{aligned}
$$

where we redefine $\rho'_j = 1 - \frac{1}{2}R_j^2, \hat{\boldsymbol{w}}_j = \hat{\boldsymbol{c}}_j, \|\hat{\boldsymbol{w}}_j\|_2^2 = 1, \quad \forall j \in \{1, ..., m\}$. The above formulation indicates that the loss mSVDD is equal to the loss of multi-modal OC-NN, $\mathcal{L}_{mOCNN}$ (Eq. (4.12)), up to a factor 2. Therefore, we can prove the equivalence of soft-boundary mSVDD and multi-modal OC-NN under the condition of normalization. Certainly, the identity can hold when we use uni-modal deep SVDD and OC-NN. □



(a) Multi-modal Deep SVDD with two hyperspheres represented by $(\boldsymbol{c}_1, R_1)$ and $(\boldsymbol{c}_2, R_2)$.



(b) Multi-modal OC-NN with two hyperplanes represented by $(\boldsymbol{w}_1, \rho_1)$ and $(\boldsymbol{w}_2, \rho_2)$.

Figure 4.2: Multi-modal Deep SVDD and OC-NN with two modes, where the data representations are mapped to unit norm hypersphere. Black points denote normalized data samples.

As for the geometric interpretation, Fig. 4.2 illustrates how the two models show when data representations are normalized and mapped onto unit hypersphere. The two

subfigures in Fig. 4.2 show multi-modal OC-NN and SVDD with two modes, respectively. As shown in Fig. 4.2, each data sample can be regarded as a point on the unit hypersphere, and maximizing the margins of hyperplanes from the origin in OC-NN is equal to minimizing the radii of hyperspheres in SVDD. We can also get the relationship from $\rho'_j = 1 - \frac{1}{2}R_j^2$ which shows that the margin ρ'_j is inversely proportional to the radius R_j. During the (maximizing/minimizing) optimization process in OC-NN and mSVDD, these two neural models can also perform the feature learning such that data representations are rotated to their "center" representations (i.e., $\boldsymbol{w}$ and $\boldsymbol{c}$) on the unit hypersphere [90, 22, 175].

4.3.4 Summarizing mSVDD

We summarize the proposed mSVDD in accordance with the discussions presented above. The proposed multi-modal deep SVDD (mSVDD) learns a compact description of one-class data with multiple hyperspheres. mSVDD is also a generic framework that includes deep SVDD, CVDD, DMSVDD and OC-NN if the corresponding conditions are met.

4.4 Multi-modal Deep SVDD with Negative Supervision

In this section, we incorporate negative supervision into the training of mSVDD. The SVDD-related models are usually trained with only positive samples from the target one-class, while, if negative samples are available, the models can be extended to train with them to improve the description [157][4]. Note that these samples are *not* necessarily required to be from "*real*" negative class. For example, in our experiment, external data can be seen as one of the choices for the incorporated *pseudo*-negative samples.

Given a set of extended training samples $T' = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_{n'}, y_{n'})\}$, where the first n samples are labeled $y_i = 1$, denoting positive, whereas the others are labeled $y_i = 0$, which denotes negative samples that should be rejected by mSVDD. The proposed mSVDD is represented with m hyperspheres and is formulated as $M = \{\boldsymbol{M}_1(\boldsymbol{c}_1, R_1), ..., \boldsymbol{M}_m(\boldsymbol{c}_m, R_m)\}$. It is required that the positive samples should be

[4] Section 2.2

inside the m hyperspheres, while the negative samples should lie outside. Given training samples composed of positive and a negative samples, we can first get their corresponding distances to each center $\boldsymbol{c}_j$. The goal of optimization should be to pull the positive samples closer the center and to push the negative ones away. Formally, we define the distance between one sample $\boldsymbol{x}_i$ and one center $\boldsymbol{c}_j$ as $d_{ij} = d(\boldsymbol{x}_i, \boldsymbol{c}_j) = \|\phi(\boldsymbol{x}_i; \mathcal{W}) - \boldsymbol{c}_j\|_2^2$. There are usually two types of losses to obtain the discriminative loss.

Contrastive type: The contrastive-type loss directly optimizes the distance by encouraging the distance between a positive sample and a center to be smaller, while it forces the larger distance to a negative sample:

$$\mathcal{L}_{Con\_d}^{(ij)} = y_i[d_{ij} - R_j^2]_+ + (1 - y_i)\,[R_j^2 - d_{ij}]_+, \tag{4.15}$$

where R_j^2 can be seen as a *margin* (or *threshold*) with a function that prevents too much effort from being wasted in enlarging/reducing distances [38].

Triplet type: The triplet-type loss is defined for a pair of positive sample $\boldsymbol{x}_i$ and negative sample $\boldsymbol{x}_{i'}$. If we consider center $\boldsymbol{c}_j$ as an *anchor* representative of target data, the triplet loss punishes only when d_{ij}, the distance from $\boldsymbol{x}_i$ to $\boldsymbol{c}_j$, is greater than $d_{i'j}$, the distance from $\boldsymbol{x}_{i'}$ to $\boldsymbol{c}_j$, with a margin $\tau > 0$:

$$\mathcal{L}_{Tri\_d}^{(ii'j)} = [d_{ij} - d_{i'j} + \tau]_+ . \tag{4.16}$$

For clarity, Eqs. (4.15) and (4.16) show only the two types of losses for one hypersphere. Multi-modal version can be obtained by sum operation over $j \in \{1, ..., m\}$.

The triplet loss forces only positive samples to be closer to the center than negative samples, and the contrastive loss requires only keeping the distances for negative samples above the radius. These two types of objectives are easy to achieve, especially when we assume that negative samples are "*not real*." This may result in failing to make a full use of negative supervision. Therefore, we will reformulate both $\mathcal{L}_{Tri\_d}$ and $\mathcal{L}_{Con\_d}$.

4.4.1 Reformulating Contrastive and Triplet Losses

Normalization layer In neural models with the contrastive or triplet loss, it is a common strategy to normalize the feature representations of samples for training stability [151, 165]. Therefore, we apply the normalization to the input vectors:

$\hat{\boldsymbol{x}} = \boldsymbol{x}/\sqrt{\sum |x_i|^2 + \epsilon}$, where $\epsilon > 0$ is a value avoiding division by zero.

Reformulation Given a center $\boldsymbol{c}_j$ and positive and negative samples, we can use the probability form in the optimization objective, rather than the two *non-probabilistic* ones: $\mathcal{L}_{Con\_d}$ and $\mathcal{L}_{Tri\_d}$. We introduce $p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{c}_j)$, which is the probability that a hypersphere with center $\boldsymbol{c}_j$ *accepts* the sample $\boldsymbol{x}_i$, and define it as follows:

$$p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{c}_j) = \sigma(s\hat{\boldsymbol{f}}_i^{\,T}\hat{\boldsymbol{c}}_j), \tag{4.17}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$, $\boldsymbol{f}_i = \phi(\boldsymbol{x}_i; \mathcal{W})$ denotes the feature output vector of $\boldsymbol{x}_i$, and s is a scale hyper-parameter for preventing failed convergence [170] after the normalization. For each sample $\boldsymbol{x}_i$, $\boldsymbol{c}_j$ acts as a *pseudo*-weight vector for the classification of the j-th hypersphere of mSVDD. Thus, given $p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{c}_j)$, the probability of a sample being accepted by hypersphere $\boldsymbol{M}_j$, we can reformulate the two discriminative losses with the probability.

***Contrastive type loss*:**

$$\begin{aligned}\mathcal{L}_{Con}^{(ij)} &= -y_i \log p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{c}_j) - (1 - y_i) \log p(y_i = 0|\boldsymbol{x}_i, \boldsymbol{c}_j) \\ &= -y_i \log \sigma(s\hat{\boldsymbol{f}}_i^{\,T}\hat{\boldsymbol{c}}_j) - (1 - y_i) \log \sigma(-s\hat{\boldsymbol{f}}_i^{\,T}\hat{\boldsymbol{c}}_j),\end{aligned} \tag{4.18}$$

This loss maximizes the likelihood of training positive samples being accepted or negative rejected.

***Triplet type loss*:**

$$\begin{aligned}\mathcal{L}_{Tri}^{(ii'j)} &= [\log p(y_{i'} = 1|\boldsymbol{x}_{i'}, \boldsymbol{c}_j) - \log p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{c}_j) + \tau]_+ \\ &= \left[\log \sigma(s\hat{\boldsymbol{f}}_{i'}^{\,T}\hat{\boldsymbol{c}}_j) - \log \sigma(s\hat{\boldsymbol{f}}_i^{\,T}\hat{\boldsymbol{c}}_j) + \tau\right]_+\end{aligned} \tag{4.19}$$

The loss will punish when the log probability of a negative sample is greater than a positive sample with a margin τ.

4.4.2 Reformulating Contrastive and Triplet Losses for Multiple Modes

While Eqs. (4.17), (4.18), and (4.19) show the uni-modal case, for the multi-modal one, we have to consider m different centers $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m\}$ in the calculation of the two reformulated discriminative losses. Therefore, we propose two strategies as follows:

$$p(y_i = 1|\boldsymbol{x}_i) = \begin{cases} \max\limits_{1 \leqslant j \leqslant m} p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{c}_j) & \textit{Max} \\ \frac{1}{m} \sum_j p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{c}_j) & \textit{Mean}, \end{cases} \tag{4.20}$$

where *Max* references only $\boldsymbol{M}_j$ with the max logit output, while *Mean* takes account of all hyperspheres equally. Then, we can obtain the corresponding *Contrastive* and *Triplet* losses by substituting Eqs. (4.18) and (4.19) with the probability term (Eq. (4.20)).

4.4.3 Training Loss

The final training loss for the mSVDD with negative supervision can be formulated as:

$$\mathcal{L} = \mathcal{L}_{mSVDD} + \gamma \mathcal{L}_{Con|Tri}, \tag{4.21}$$

where γ adjusts between the mSVDD loss and the discrimination with negative supervision. In the training process, $\mathcal{L}_{Con|Tri}$ will sum the loss from one batch samples with Eqs. (4.18) and (4.19).

4.4.4 Relationship between mSVDD and the Use of Negative Supervision

mSVDD and negative supervision are not two independent sub-architectures. Negative supervision, including *contrastive* and *triplet* losses, are specially equipped to mSVDD. Specifically, these two components are closely connected by the center of the hypersphere, $\boldsymbol{c}_j$. Both mSVDD (Eq. (4.8)) and negative supervision (Eq. (4.18) or (4.19)) contain $\boldsymbol{c}_j$. Since there is no *real* negative data, external data are used as *pseudo* negative samples to complete negative supervision. The use of negative supervision can improve the discrimination ability of mSVDD. In training, negative supervision loss forces mSVDD to reject *unseen* samples since real negative data in

testing are also unseen in training. This improves inter-class discrepancy, compared with intra-class loss mSVDD optimized. However, in testing, the decision function will be the same as mSVDD trained only with positive samples.

4.5 Experiments

4.5.1 Datasets and Implementation Details

Datasets Experiments were conducted on two datasets: *20 Newsgroups*[5] and *Reuters*[6], which have been commonly used in other one-class text classification work [99, 141]. We also conducted experiments on another dataset *TREC*[7] [84]. We follow the same one-class classification setting in [141], i.e., one of the classes in each dataset is considered as the target class, we call it as *target* or *positive*, while the remaining classes are considered *negative*. Note that we labelled positive samples as $y = 1$, while negative ones $y = 0$. We used the same pre-processing steps as the ones used in earlier work [141], including lowercasing, removing stopwords, and tokenization. We used the external data for negative supervision in the absence of "*real*" labeled negative instances. We followed the similar logic for choosing our external data as the one in the field of pretrained word vectors, in which one general corpus, such as Wikipedia articles, is often adopted as the training dataset [105]. So we also chose one publicly available corpus WikiText-2 [104], extracted from Wikipedia articles, as our external data. In the training, data loader loads one batch of negative samples, i.e., sentences from WikiText-2, which are labeled with 0.

Encoder For encoding the text input, i.e., $\phi(\boldsymbol{x}, W)$, we used a Bidirectional LSTM with attention [50, 173], with the number of hidden units being 150. For the pretrained word embeddings, we experimented with GloVe Vectors [123] and set the dimension to 300. In our experiments, we did not adopt the widely used BERT model [23], as [141] showed that BERT model did not improve the performance.

Settings As for the optimization of parameters, Adam [69] with a base learning rate of 0.001 was used for 50 epochs. The batch sizes were set to 32, 32, and 64 for *Reuters*, *TREC*, and *Newsgroups*, respectively. For the initialization of mSVDD

[5]http://qwone.com/json/20Newsgroups
[6]http://daviddlewis.com/resources/testcollections/reuters21578/
[7]https://cogcomp.seas.upenn.edu/Data/QA/QC/

model, we employed two operation steps. In the absence of negative samples, mSVDD was first pre-trained on target samples by using an AutoEncoder with two objectives: 1) warm-up and 2) reducing the *reconstruction* error for the target samples, such that the model can be more robust to noise or anomalous inputs [58, 49]. An AutoEncoder feed-forward network with a 0.5 compression rate, which consists of an encoder and a decoder, was put on the back of the BiLSTM feature network. Then, the weights of the m hyperspheres in mSVDD were initialized by running k-means clustering on the features learned before [92]. As for the regularization term of mSVDD, $\boldsymbol{c}_j$ was regularized [113], and a weight decay with 0.95 was applied for the parameters. As for the number of hyperspheres, different settings, $1, 3, 5, 10$, were tested. For the hyperparameters, we set parameter $s = 1.2$ for scale, $\nu = 0.1$, $\delta = -0.9$ for the attention weight, $\tau = 0.1$ for the triplet loss, $\epsilon = 1\mathrm{e}{-6}$ for norm, and $\gamma = 1$ for the training loss. The results were averaged over 10 runs with different random seeds.

Evaluation metrics The performance was measured by the *area under the receiver operating characteristics* (ROC) curve (AUCs), a commonly used metric for one-class text classification [99, 141].

4.5.2 Results

Comparison between multi-modal and unit-modal SVDD

Table 4.1 shows the performance of mSVDD with different choices of m, i.e., the number of hyperspheres. Here, mSVDD(1) represents uni-modal deep SVDD (DSVDD) [140]. The results show that: 1) mSVDD outperformed the uni-modal DSVDD under most target classes over three datasets. Generally, multi-modal mSVDD provided better performances by obtaining best scores in more times than the uni-modal one. As for the *one-class* version, compared to mSVDD(3), mSVDD with more hyperspheres (5) setting performed better in *Reuters* and *TREC*, while there was no improvement found on *Newsgroup*. mSVDD(3) also shows improvements on four target classes on *TREC* datset. Similar results can also be observed for *soft-boundary* version. On the three target classes, *rec*, *sci* and *misc*, *soft-boundary* mSVDD(3) shows consistent performances. It shows also that the *soft-boundary* mSVDD(5) achieved the best scores on five out of size cases in *TREC* dataset. The results show the effects of incorporating more hyperspheres to better describe the target data.

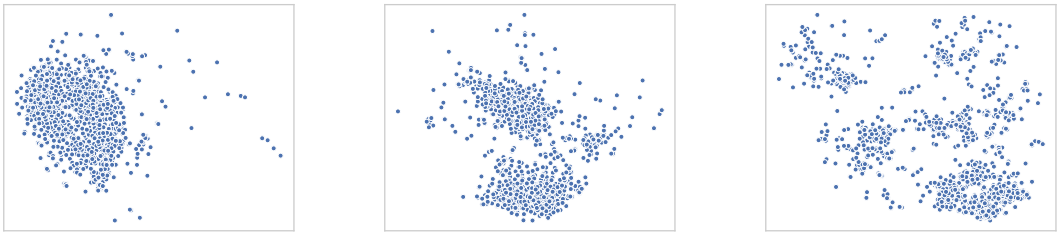
Target Class	*One-v* mSVDD (m) 1	3	5	10	*Soft-v* mSVDD(m) 1	3	5	10
Reuters								
earn	95.6	95.5	**96.0**	95.9	95.9	95.9	**96.2**	96.1
acq	89.4	90.0†	89.3	**90.1**	89.0	89.1	89.1	**89.2**
crude	**92.7**	92.5	92.5	92.4	**92.8**	91.5	92.5	92.4
trade	98.4	98.3	**98.8**†	98.6†	98.3	**98.9**†	98.7	98.8
money	86.3	85.1	86.4	**87.1**	86.2	86.2	86.4	**86.8**
interest	97.2	97.2	97.2	**97.3**	**97.3**	96.9	96.6	96.8
ship	92.5	**93.8**†	**93.8**†	92.6	91.7	91.6	**92.3**†	91.7
20 News								
comp	85.3	86.2	86.1	**86.7**	84.9	86.0	85.9	**86.5**
rec	77.1	**77.7**†	77.6	77.6	76.2	**77.0**†	76.9	76.8
sci	66.5	**67.3**†	67.1	66.9	66.3	**67.3**†	66.7	67.0
misc	75.2	**76.0**†	75.5	75.5	75.0	**76.2**	**76.2**	**76.2**
pol	**79.2**	78.5	78.7	78.5	**79.1**	78.7	78.4	78.4
rel	**83.6**	83.1	83.1	83.1	**82.5**	**82.5**	82.3	82.2
TREC								
ENTY	71.8	72.4	**72.5**†	72.3	70.9	71.8	**72.0**	71.8
HUM	75.3	**76.4**†	**76.4**	**76.4**	75.7	76.6	**76.8**†	76.7
DESC	**52.9**	**52.9**	52.6	52.4	54.5	**55.3**†	**55.3**†	54.9
NUM	73.7	74.5†	**74.7**	74.6	73.4	74.4	**74.5**	74.4
LOC	79.0	**79.7**	**79.7**	79.6	78.7	79.5	**79.6**†	79.4
ABBR	**97.6**	96.6	96.3	94.7	**97.3**	96.9	96.4	96.7

Table 4.1: Results of mSVDD with different settings of m. Numbers in brackets denote the number of hyperspheres in mSVDD. *One-v* and *Soft-v* denote the two versions of mSVDD, *One-class* and *Soft-boundary*, respectively. AUCs in % on the *Reuters* (upper part), *20 Newsgroup* (middle part) and *TREC* (lower part) datasets. Best scores in each *row* are presented in **bold** within each *One-v* and *Soft-v* group. Significant improvements over uni-modal have been marked with a dagger $^\dagger(p < 0.05)$.

2)The performance of mSVDD did not improve linearly along with m. For examples, except for *comp* in *Newsgroup* dataset, mSVDD(10) with *one-class* form won't achieve best scores. While mSVDD(3 or 5) perform the best in most target classes over three datasets in the *one-class* loss setting. This indicates that the incorporation of more descriptions is not necessary sometimes. Fig. 4.3 shows an example with t-SNE [161] visualization of feature representations learned by DSVDD, mSVDD, and CVDD. For the data inputs for target *rec*, mSVDD maps data to multiple clusters in the feature space which matches the assumption that there are multiple sub-groups on *rec*. While DSVDD maps them into one cluster, and CVDD outputs feature representations separated by five heads. As for the performance, both forms of mSVDD obtained better scores than CVDD or DSVDD on *rec*.

As for uni-modal has good results in some cases such as *ABBR*, *pol*, and *rel*. We can explain this from the following aspects. As for the model, mSVDD with more centers means that it has more parameters and a complex model structure, which is hard to be optimized, especially on the data with a small training size (e.g., *pol* or

rel.) As for the data, some data might have simple data distributions without the need for more modes. Another aspect would be the attention weights of multiple hyperspheres. [29] showed that focusing on some "*good*" hyperspheres would be beneficial rather than over all hyperspheres. In the calculation of attentions, we did not adjust δ so as to have a large weight for one specific hypersphere. This may cause limited improvements. We will compare mSVDD with DMSVDD later.



(a) DSVDD (b) mSVDD (c) CVDD

Figure 4.3: t-SNE visualization of feature representations learned by DSVDD, mSVDD and CVDD over *rec* on *Newsgroup* dataset

Comparison between mSVDD and other models

Table 4.2 shows the results of the comparison between mSVDD and other models. From the discussion in the last subsection, we used $m = 3$ for mSVDD in this subsection. As for the three baseline methods, kernel OC-SVM, SVDD, and CVDD, we adopted their *best* scores on *Reuters* and *20 Newsgroup* from [141]. As for *TREC*, we conducted experiments with the same setting with [141] and reported the best results for the above three approaches. For the method of DMSVDD, we reported the results in the setting of the initial number of spheres $K_{init} = 10$. We also ran experiments with one-class neural networks (OC-NN) [14]. Since OC-NN[8] was originally designed and applied for computer vision tasks, we re-implemented it to match our experiments with PyTorch [121].

From the results in Table 4.2, we have the following observations. In general, CVDD performed better than other neural models on *Reuters* dataset, while mSVDD achieved better performances than other models on other two datasets, *Newsgroup* and *TREC*. As for the two traditional kernel methods, they performed well on *Reuters*, especially for *earn and acq*. As regard the comparison between DMSVDD and mSVDD, DMSVDD puts value on one hypersphere and performed slightly better over mSVDD(3)

[8]https://github.com/raghavchalapathy/oc-nn

in some cases (e.g., *earn, acq* and *comp*). This indicates one inspiration that discarding "*bad*" hyperspheres is sometimes necessary. Regarding OC-NN, another uni-modal method, mSVDD performed better than it in most target classes, while OC-NN obtained high scores in the cases of *ABBR* and *pol*. Therefore, OC-NN and its multi-modal version deserve to be explored and discussed in-depth in the future. Even mSVDD did not obtain the best AUC scores on *Reuters*, it still attained the second best in three cases. And the performances on the most cases for other two datasets showed the advantage of the proposed mSVDD over other related baseline models.

	Reuters target class							*20 Newsgroup* target class						*TREC* target class					
Model	*earn*	*acq*	*crude*	*trade*	*money*	*interest*	*ship*	*comp*	*rec*	*sci*	*misc*	*pol*	*rel*	*ENTY*	*HUM*	*DESC*	*NUM*	*LOC*	*ABBR*
OC-SVM/SVDD	91.1	**93.1**	92.4	99.0	**88.6**	97.4	93.1	82.0	75.6	64.1	63.1	75.5	79.2	66.6	71.5	52.0	67.7	74.3	93.1
OC-NN	95.7	90.0	91.8	98.2	85.0	97.3	93.2	84.4	76.3	66.1	75.3	78.6	82.1	71.1	75.9	54.0	73.8	79.0	**97.2**
CVDD	94.0	91.5	**95.5**	**99.2**	82.8	**97.7**	**97.6**	70.9	53.3	56.8	75.1	65.3	76.3	60.4	62.0	**76.1**	65.4	64.8	82.0
DMSVDD	**96.0**	89.8	92.1	98.8	87.1	97.2	93.0	**86.3**	77.1	66.8	75.3	78.5	82.0	70.6	75.2	53.6	73.0	78.2	96.0
mSVDD_*One*	95.5	90.0	92.5	98.3	85.1	97.2	93.8	86.2	**77.7**	**67.3**	76.0	78.5	**83.1**	**72.4**	76.4	52.9	**74.5**	**79.7**	96.6
mSVDD_*Soft*	95.9	89.1	91.5	98.9	86.2	96.9	91.6	86.0	77.0	**67.3**	**76.2**	**78.7**	82.5	71.8	**76.6**	55.3	74.4	79.5	96.9

Table 4.2: Results of mSVDD and other models. Average AUCs in % on the *Reuters* (left part), *20 Newsgroup* (middle part), and *TREC* (right part). _*One* and _*Soft* mean *One-class* and *Soft-boundary* forms, respectively. Best scores in each *column* are presented in **bold**, while the second best are underlined.

Results of mSVDD with negative supervision

	Reuters target class							*20 Newsgroup* target class						*TREC* target class					
Model	*earn*	*acq*	*crude*	*trade*	*money*	*interest*	*ship*	*comp*	*rec*	*sci*	*misc*	*pol*	*rel*	*ENTY*	*HUM*	*DESC*	*NUM*	*LOC*	*ABBR*
mSVDD_*One*	95.5+	90.0	92.5+	98.3+	85.1+	97.2+	**93.8**	86.2+	77.7+	67.3+	76.0+	78.5+	83.1+	72.4	76.4+	52.9+	74.5+	79.7+	96.6+
+Triple+Max	96.9	89.4	93.8	**99.6**	89.0	**98.4**	92.7	88.3	78.6	67.5	**77.6**	79.9	83.8	**74.7**	78.4	60.6	77.1	**82.4**	97.3
+Triple+Mean	97.1	89.9	**93.9**	99.5	89.3	98.3	92.3	87.9	77.5	67.7	75.5	79.1	83.9	74.6	78.4	61.7	**77.2**	82.3	96.7
+Con+Max	**97.2**	90.8	92.9	98.8	**91.3**	97.8	92.3	**89.4**	**79.1**	**68.3**	76.4	**80.7**	84.2	71.1	**78.5**	65.3	75.9	82.1	**98.5**
+Con+Mean	96.6	**91.0**	92.8	98.6	90.2	98.0	91.9	89.2	78.9	**68.3**	76.6	79.9	**84.4**	71.5	77.8	**69.7**	75.7	81.8	98.4
mSVDD_*Soft*	95.9+	89.1	91.5+	98.9+	86.2+	96.9+	91.6	86.0+	77.0+	67.3+	76.2+	78.7+	82.5+	71.8	76.6+	55.3+	74.4+	79.5+	96.9
+Triple+Max	97.1	89.6	92.8	**99.4**	89.3	97.4	92.6	87.8	78.5	68.8	76.1	79.9	82.9	72.7	78.1	63.0	76.5	**81.8**	96.8
+Triple+Mean	**97.2**	**90.1**	92.4	99.3	91.0	98.0	**92.7**	87.5	**78.7**	68.5	76.6	79.7	83.0	**72.8**	78.3	62.7	**76.6**	**81.8**	96.5
+Con+Max	97.0	88.2	**93.1**	99.2	91.2	**98.4**	91.6	**88.6**	78.3	**69.0**	**78.3**	80.5	83.2	70.4	**78.4**	63.4	76.2	81.6	97.9
+Con+Mean	**97.2**	88.1	93.0	98.9	**91.3**	**98.4**	90.4	88.3	**78.7**	68.4	78.1	**80.8**	**83.4**	70.1	78.1	**69.6**	75.4	81.3	**98.4**

Table 4.3: Results of mSVDD with negative supervision. Average AUCs in % on the *Reuters* (left part), *20 Newsgroup* (middle part), and *TREC* (right part). +Triple+Max, which denotes mSVDD with *Triplet* loss with *Max* probability strategy, followed by three other negative supervision methods. In the rows of mSVDD_*Soft* and mSVDD_*One*, '+' following numbers means that there were improvements with negative supervision (three of four methods.) Best scores in each *column* are presented in **bold** within each _*One* and _*Soft* group.

Table 4.3 shows the performance of mSVDD trained with negative supervision. As mentioned, we still used $m = 3$ in this subsection. To perform negative supervision for mSVDD, we evaluated four approaches where different losses and their reformulated probability forms were selected.

For *Reuters*, the results indicate that mSVDD benefited from the joint training of the discrimination losses, except for *acq* and *ship* where negative supervision did not show consistent improvements (three of four methods). We can see more obvious improvements for *20 Newsgroup*. All four negative supervision methods improved mSVDD markedly for all target classes of *20 Newsgroup*. For example, mSVDD with negative supervision gained 2-3 points for *comp*. For different losses for negative supervision, the *contrastive* type loss, which has larger punishment over negative data, performs better than the *triplet* type loss, which uses a relatively small margin. In addition, the performance of *Con+Max* was greater than the *Con+Mean* strategy to reformulate the probability. We hypothesize that focusing on one of the hyperspheres is effective when we used mSVDD with the *contrastive* loss.

As for the third *TREC* dataset, mSVDD with negative supervision achieved the best results on all target classes. The greatest gain came from *DESC* when the *one-class* form mSVDD trained with *contrastive* type loss with *mean* probability strategy. As for the probability strategy on *TREC*, *max* strategy would be beneficial to the *one-class* form, while *max* for *soft-boundary* mSVDD. In summary, Table 4.3 validated that mSVDD can be further improved by incorporating the proposed negative supervision methods.

Results of CVDD with negative supervision

Table 4.4 shows the results of CVDD with the proposed negative supervision for mSVDD. As mentioned in Section 4.3.3, CVDD can be seen as a special case of mSVDD. Therefore, the proposed negative supervision approach to mSVDD can be also applied to CVDD theoretically. To highlight the usefulness of the negative supervision, we conducted experiments to use the *triplet* loss with *Max* probability for CVDD.

As for the implementation, since CVDD uses a different multi-head structure, we also used a different form to incorporate *Triplet+Max* to CVDD. Specifically, CVDD uses a group of r *context vectors* $C = (\boldsymbol{c}_1, ..., \boldsymbol{c}_r)$ to describe the target one-class data, where $\boldsymbol{c}_k \in \mathbb{R}^p$. Given one context vector $\boldsymbol{c}_k, \forall k \in \{1, ..., r\}$ and a pair of training positive and negative samples, $(x_i, x_{i'})$, we can get the reformulated probability form. First, CVDD maps a training sample x_i to r heads of feature vectors $S_i = (\boldsymbol{s}_{i1}, ..., \boldsymbol{s}_{ir})$. Then, we denote $p(y_i = 1|\boldsymbol{s}_{ik}, \boldsymbol{c}_k)$ as the probability that k-th $\boldsymbol{s}_{ik}$ reconstructs k-th context vector $\boldsymbol{c}_k$ well.

Model (r)	*Reuters* target class						
	earn	*acq*	*crude*	*trade*	*money*	*interest*	*ship*
CVDD (3)	94.0	90.2	89.6	98.3	82.5	92.3	97.6
+Triple+Max	**96.1**	90.2	**97.3**	98.3	**84.2**	**92.4**	91.8
CVDD (5)	92.8	88.7	92.5	98.2	76.7	91.7	96.9
+Triple+Max	**94.0**	**94.4**	**96.7**	**98.7**	**84.0**	**97.3**	92.5
CVDD (10)	91.8	91.5	95.5	99.2	82.8	97.7	95.6
+Triple+Max	**93.0**	91.2	**97.4**	**99.6**	**85.7**	**98.7**	94.2

Model (r)	*20 Newsgroup* target class					
	comp	*rec*	*sci*	*misc*	*pol*	*rel*
CVDD (3)	70.9	50.8	56.7	75.1	62.9	76.3
+Triple+Max	**74.5**	**64.2**	**61.0**	75.1	62.2	72.5
CVDD (5)	66.4	52.8	56.8	70.2	65.3	72.9
+Triple+Max	**73.2**	**64.5**	**58.4**	**76.2**	63.6	**76.1**
CVDD (10)	63.3	53.3	55.7	68.6	65.1	70.7
+Triple+Max	**78.3**	**69.7**	**60.5**	**73.3**	**67.5**	**79.1**

Model(r)	*TREC* target class					
	ENTY	*HUM*	*DESC*	*NUM*	*LOC*	*ABBR*
CVDD (3)	58.6	55.4	76.1	60.5	62.1	74.2
+Triple+Max	**68.3**	**70.7**	**81.1**	**74.1**	**71.6**	**86.7**
CVDD (5)	60.4	62.0	54.2	65.4	63.8	82.0
+Triple+Max	**65.6**	59.7	**82.4**	**73.4**	**67.7**	**88.7**
CVDD (10)	60.0	56.1	69.8	65.2	59.7	80.7
+Triple+Max	**72.0**	**69.8**	**80.4**	**69.9**	**81.0**	**88.1**

Table 4.4: CVDD with the proposed negative supervision. Average AUCs in % on the *Reuters* (upper part), *20 Newsgroup* (middle part), and *TREC*(lower part) datasets. Number r in brackets denotes the number of heads in CVDD. **Bold** means the better AUCs score.

$$p(y_i = 1|\boldsymbol{s}_{ik}, \boldsymbol{c}_k) = \sigma(\hat{\boldsymbol{s}_{ik}}^T \hat{\boldsymbol{c}_k}) \tag{4.22}$$

And with *triplet* and *Max* probability strategy, we can define the negative supervision loss as:

$$\begin{aligned} \mathcal{L}_{Tri}^{(ii')} &= [\log p(y_{i'} = 1|\boldsymbol{x}_{i'}) - \log p(y_i = 1|\boldsymbol{x}_i) + \tau]_+ \\ &= [\log \max_{k=1,\dots,r} p(y_{i'} = 1|\boldsymbol{s}_{i'k}, \boldsymbol{c}_k) - \log \max_{k=1,\dots,r} p(y_i = 1|\boldsymbol{s}_{ik}, \boldsymbol{c}_k) + \tau]_+ \end{aligned} \tag{4.23}$$

where τ is a margin. Then, $\mathcal{L}_{Tri}^{(ii')}$ can then be added to Eq. (4.10) to obtain the training

loss.

Overall, we can see that the proposed negative supervision enhanced CVDD in most cases on the three datasets. The overall performance mainly shows the following: 1) The improvement by the negative supervision to CVDD is consistent with mSVDD due to the similarity between the two. 2) The generality of the negative supervision can be shown, as *Triple+Max* was successfully applied to the different multi-head structure.

Regarding different target-classes, *ship* with the smaller training data size may cause worse performance, so does *real* with CVDD(3), which are similar phenomena with mSVDD. While for *TREC*, almost all the target classes can be improved by using negative supervision. In addition, the negative supervision also prevented over-fitting for CVDD. For example, CVDD(3) with the minimal parameters achieved the best score for *comp* when varying "*r*" among 3, 5 and 10. In contrast, when the negative supervision was used, CVDD(10) with the maximal parameters attained the best and also performed better for all six target classes of the *20 Newsgroup* dataset.

Negative supervision using word replacement with TF-IDF

In the former parts, we have introduced that mSVDD can be further improved by incorporating negative supervision. And negative supervision was trained on *pseudo*-negative samples that were randomly sampled sentences from a certain publicly available corpus, WikiText-2 [104]. While this sentence-level sampling strategy can provide a wealth of out-of-domain samples with diverse semantics, it is not easy to control the generation of samples and the adaptability of diverse target classes settings over different datasets.

Thus, we provide another method for *pseudo*-negative construction without the use of out-of-domain corpus. This idea was motivated by the recent remarkable data augmentation techniques, especially by unsupervised data augmentation method [172]. In general, we propose a word-level data augmentation method to construct *pseudo*-negative samples from the only given the positive target training dataset.

Our method is very simple. First, we assume that : 1) The *pseudo*-negative samples should have more common words to target samples than samples from external corpus in the one-class classification setting. 2) While *pseudo*-negative samples constructed should *not* contain words that are more informative to the instances from the target class. Therefore, we proposed a method that replaces *informative* words with high

TF-IDF scores while keeping those with low values. Now, for a given target training dataset, this approach could construct *pseudo*-negative samples on the basis of itself, rather than any other external corpus.

Formally, for a word w in one target training text sequence, we compute its TF-IDF value as TFIDF(w)= TF(w)*IDF(w), where TF(w) is the *term frequency* score for word w in this sequence, and IDF(w) denotes the *inverse document frequency* calculated on the only target training dataset. Specifically, we sample words to be replaced with weights according to the TF-IDF value, i.e., sampling with weights $p(w) \propto$ TFIDF(w)/Z, where Z is the sum of TF-IDF scores in one sequence. When a word is replaced, we randomly sample one word from the whole vocabulary for the replacement. Once completed the replacement for one input sequence, we can get one corresponding *pseudo*-negative sample. In the training, for one batch of positive training samples, we randomly sampled its *pseudo*-negative sample from the generated augmented set. Then, same as before, the constructed word-level *pseudo*-negative samples were used for negative supervision.

Table 4.5 shows the results of mSVDD with negative supervision using word replacement with TF-IDF. In general, it shows that the proposed negative supervision still works with the *pseudo*-negative samples generated by the method of TFIDF replacement. We can see that mSVDD obtained improvements in most cases over the three datasets, as the same with the use of external dataset shown in Table 4.3. Besides, for *acq* and *ship* on *Reuters*, mSVDD still did not benefit from negative supervision. And *ABBR* also did not show consistent improvements for four approaches. While differently, there were more obvious variations for four negative supervision approaches, especially on the the former two datasets. *Con+Mean* strategy performed better than other three methods on ten target classes, except for *ship* where this strategy degraded the performance of mSVDD. As for *TREC* dataset, *max* strategy would be the best choice for the *triplet* loss type which beats other combinations on *HUM* and *LOC*. While *mean* strategy would be still beneficial to *constrastive* type loss, as with other two datasets.

Affects of parameter γ

Parameter γ in Eq. (4.21) controls the trade-off between the intra-class loss mSVDD optimized and the discrimination loss negative supervision learned. In the former experiments on negative supervision using TFIDF word-level replacement method,

	Reuters target class							*20 Newsgroup* target class						*TREC* target class					
Model	*earn*	*acq*	*crude*	*trade*	*money*	*interest*	*ship*	*comp*	*rec*	*sci*	*misc*	*pol*	*rel*	*ENTY*	*HUM*	*DESC*	*NUM*	*LOC*	*ABBR*
mSVDD_*One*	95.5+	**90.0**	92.5+	98.3+	85.1+	97.2+	**93.8**	86.2+	77.7+	67.3+	76.0+	78.5+	83.1+	72.4	76.4+	52.9+	74.5+	79.7+	96.6
+Triple+Max	97.7	89.1	92.8	99.3	90.3	98.2	91.7	88.0	78.9	68.9	76.5	81.5	83.3	73.4	**78.6**	58.6	78.6	**82.3**	96.6
+Triple+Mean	97.6	89.1	93.1	99.3	90.3	98.2	91.5	88.2	79.1	69.1	76.5	81.4	83.3	**73.5**	78.5	58.1	78.2	82.1	96.5
+Con+Max	97.6	89.3	93.4	99.4	90.7	97.9	91.5	88.1	79.2	69.1	75.7	81.7	83.4	71.9	76.9	62.8	79.3	81.9	95.1
+Con+Mean	**97.9**	89.8	**94.1**	**99.5**	**91.6**	**98.4**	85.1	**89.8**	**80.7**	**70.2**	**84.1**	**82.7**	**85.8**	71.3	76.5	**64.3**	**79.7**	81.5	**97.1**
mSVDD_*Soft*	95.9+	**89.1**	91.5+	98.9+	86.2+	96.9+	91.6	86.0+	77.0+	67.3+	76.2+	78.7+	82.5+	71.8	76.6	55.3+	74.4+	79.5+	**96.9**
+Triple+Max	97.9	88.6	92.5	99.3	90.1	98.0	**91.8**	87.6	78.4	68.6	77.5	81.0	82.7	**73.3**	**79.0**	65.1	78.1	**82.1**	96.8
+Triple+Mean	97.7	88.5	92.3	99.3	90.3	97.9	91.2	87.7	78.5	68.6	77.3	81.0	82.6	72.9	78.7	65.6	78.0	82.0	**96.9**
+Con+Max	97.9	88.9	92.4	99.3	90.5	97.4	91.7	87.8	78.8	69.0	76.4	81.4	82.8	71.7	76.4	63.9	77.3	80.9	96.8
+Con+Mean	**98.0**	88.7	**93.3**	**99.5**	**91.5**	**98.2**	89.9	**88.7**	**79.7**	**69.5**	**80.5**	**81.8**	**84.1**	70.8	74.5	**67.2**	**78.2**	81.0	96.3

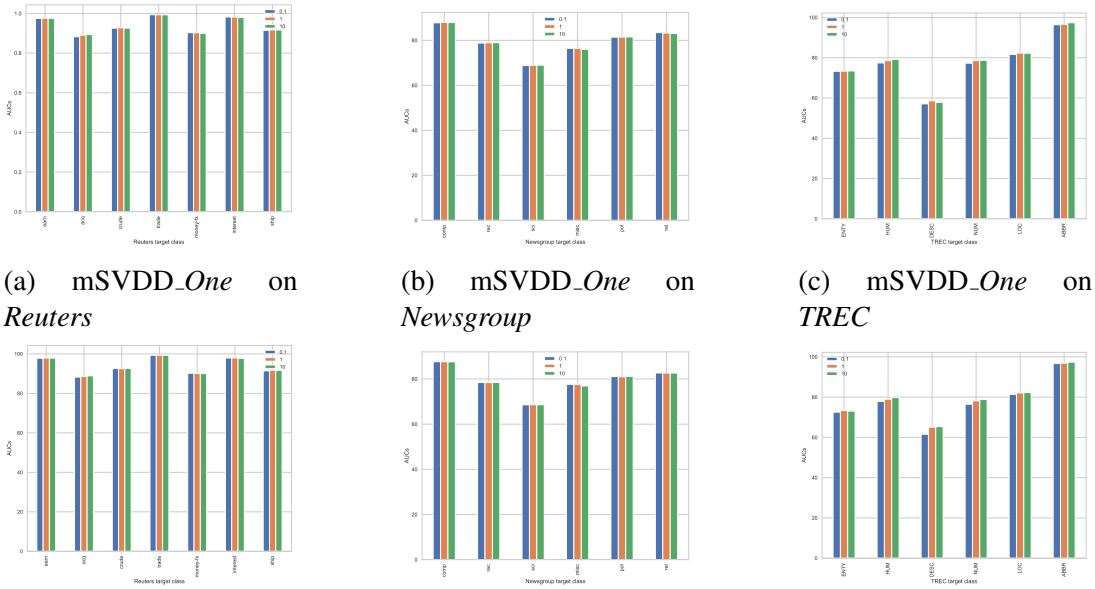
Table 4.5: mSVDD with negative supervision using TFIDF replacement method. Settings are the same with Table 4.3.

we fixed $\gamma = 1$. Here, we would like to conduct experiments with varying γ settings to verify how the trade-off affects the overall performance. There are four negative supervision approaches that can be jointly trained with mSVDD. As shown in Table 4.5, other three choices outperformed *Triple+Max* in many cases. Thus we chose *Triple+Max* for our sensitivity analysis experiments here. We expected *Triple+Max* can obtain improvements through the adjustment of γ.

Fig. 4.4 illustrates the relationship of γ and the evaluation of mSVDD+*Triple+Max* on different target classes over three datasets. As for the two versions of mSVDD on *Reuters*, Figs. 4.4a and 4.4d show that a larger γ (10 or 1) would be better setting on the former three target classes, especially on *acq*. While on *trade* or *money-fx*, mSVDD achieved better performances with smaller value of $\gamma = 0.1$. Figs. 4.4b and 4.4e indicate that when γ is set to 1 or 10, mSVDD can become slighter better for most target classes of *Newsgroup* dataset. While for *rel* with smaller data size, mSVDD became worse by focusing more on negative supervision. We can observe a clear relationship between the increases of γ and the improvements of mSVDD from Figs. 4.4c and 4.4f. These two subfigures show that mSVDD gained steady improvements over most target classes on *TREC* for both mSVDD forms. Besides, the verification performance of the negative supervision remains stable across different settings of γ over a wide range of target class settings.

4.6 Summary of This Chapter

In this chapter, we proposed mSVDD, a new generic one-class text classification framework that uses multi-modal deep SVDD. Rather than the uni-modal deep SVDD, mSVDD can enhance the description ability to the target one-class data with multiple hyperspheres. We also proved that this generic framework can include four variants, deep SVDD, DMSVDD, OC-NN, and CVDD under certain conditions. In addition,



(a) mSVDD_*One* on *Reuters*

(b) mSVDD_*One* on *Newsgroup*

(c) mSVDD_*One* on *TREC*

(d) mSVDD_*Soft* on *Reuters*

(e) mSVDD_*Soft* on *Newsgroup*

(f) mSVDD_*Soft* on *TREC*

Figure 4.4: Results of mSVDD using the method of *Triple+Max* with parameter γ varying from 0.1, 1 to 10.

in the absence of "*real*" negative training data, we also proposed approaches for effectively adding negative supervision to further improve the performance of mSVDD. The experiments validated that the proposed mSVDD provides better performance compared to uni-modal SVDD. The experiments also showed the further improvements in most cases when negative supervision was used for mSVDD and CVDD.

Chapter 5

Conclusion and Future Work

5.1 Summary of This Thesis

In this study, we have studied the problem of one-class text classification with neural networks in two works. In the first work, we focused on mutual bootstrapping, which is a commonly used technique for many natural language processing tasks, including acquiring semantic lexicon from corpora. On the basis of Basilisk, an effective and commonly used bootstrapping approach, we formulated bootstrapping as a special one-class problem. This answered the issue, *"could we formulate Basilisk as one-class problem ?"* Chapter 1 mentioned. In the setting of one-class classification, we can improve the bootstrapping framework with reconstruction-based methods. To improve the bootstrapping processes, we incorporated AutoEncoder, which is a reconstruction model that uses neural networks as the encoder and the decoder. More specifically, we used AutoEncoder to modify its two key scoring functions: candidate scoring and pattern scoring. The experimental results demonstrated that our proposed method for guiding the bootstrapping of a semantic lexicon with AutoEncoder can boost the overall performance. And the ablation study validated that the proposed pattern scoring method enhanced the performance on the basis of the contribution of its multiple different parts. Furthermore, the case study with examples also showed that the proposed methods can provide better output.

Second, we focused on a family of several boundary-based methods. We started with extending the uni-modal deep support vector data description to a multiple modal one, mSVDD we called, that enables us to build a much better description for target one-class data. Then, the close relationships between mSVDD and four models were discussed. These discussions helped us to understand the differences, and more im-

portantly, the common characteristics of deep SVDD-related approaches. Formally, we proved that mSVDD is also a generic framework that can include all four variants: DSVDD [140], DMSVDD [29], CVDD [141], and OC-NN [14] under certain conditions. We then introduced a mechanism for incorporating negative supervision in the absence of real negative data, which is beneficial to the mSVDD model. The improvement of CVDD with negative supervision confirmed the effect of negative supervision and the close relationship between CVDD and mSVDD. And the proposed negative supervision showed stable improvements with *pseudo*-negative samples generated by word-replacement method or extracted from a external corpus. In summary, the proposed methods in Chapter 4 answered the three research issues related to the second work mentioned in Chapter 1.

5.2 Retrospective Discussion on Methods

In this thesis, we have tried two types of methods, reconstruction-based and boundary-based, to one-class classification tasks. We now provide a retrospective discussion on these methods.

First, we discuss the relationship between the two types of methods. Reconstruction-based approaches usually make assumptions about data structure of target class. *Manifold* assumption, target data come from relatively low-dimensional manifolds (or subspaces), is made by AutoEncoder [115, 142, 98] and principal component analysis (PCA) [122, 52]. Another *prototype* assumption is adopted by methods like context vector data descriptions (CVDD) [141] and clustering algorithms [157, 74]. On the other hand, boundary-based methods directly learn a decision boundary that describes the target data [149, 158] without the assumptions above. Even with the difference in data assumption, both types of methods can be explained as one-class classification methods in a unified form [125, 139].

$$\min_{\theta} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(f(\phi(\boldsymbol{x}_i);\theta), y_i) + \mathcal{R}(\theta, f, \phi) \tag{5.1}$$

where ϕ denotes feature map function, f the model, θ the parameters and $\mathcal{R}$ the regularization.

In the view of unified form, more specifically, for boundary method SVDD [158] (cf. Eq. (2.4), Section 2.1.2), the model can be represented by $f = \|\phi(\boldsymbol{x}_i) - c\|^2 - R^2$, θ contains c and R, and it uses kernel to learn feature map and hinge loss $max(0, \cdot)$

calculates the classification risk. In this case, most of the n training samples are labelled with $y = 1$. In the case of unsupervised AutoEncoder (cf. Eq. (2.1), Section 2.1.1), the model becomes $f = \|\boldsymbol{x}_i - \phi_{de}(\phi_{en}(\boldsymbol{x}_i; \theta_{en}); \theta_{de})\|^2$, θ contains both the parameters from the encoder and decoder, and ϕ is a neural networks with bottleneck structure consists of encoder ϕ_{en} and decoder ϕ_{de}. Here we do not need the label y_i.

Second, we discuss the applicability of methods used for the tasks in this study. Chapter 3 processed the task of acquiring semantic lexicons under a specific bootstrapping framework. We did a preliminary attempt that this task can be solved under one-class problem setting and improved by AutoEncoder, one of reconstruction-based approaches. Instead, in Chapter 4, we focused on a more general one-class classification problem for text data and tried a family of boundary-based approaches.

Technically speaking, AutoEncoder can also be applied to the task in Chapter 4. That's exactly what we have tried to do before the forming of the idea (mSVDD) in the second work. We tried to directly apply kinds of AutoEncoder structures (e.g., denoising AutoEncoder [163], or generative adversarial networks (GAN) based AutoEncoder [124]) to perform one-class classification on *Newsgroup* and *Reuters* datasets. While there is no improvement over the two more popular boundary-based baseline methods in Chapter 4: SVDD and OC-SVM [99, 158]. As shown in Eq. (5.1), besides the model f and regularization item $R(f, \theta, \phi)$(e.g., rule the various forms of AutoEncoder), feature map ϕ is another important component to influence the performance of OCC methods [139]. Therefore, to improve the performance of AutoEncoder, we also incorporated BERT [23], one of the most remarkable pre-trained models, as the choice of ϕ. Contrary to expectations, the incorporation of BERT to AutoEncoder did not bring any anticipated result on two datasets with different settings. So we did not chose AutoEncoder as the baselines and did not use BERT for encoding text in Chapter 4, so did [141].

AutoEnocer and its variants have been successfully applied to many NLP tasks with word-level data (e.g., bootstrapping of the semantic lexicon in Chapter 3) [87, 178] and computer vision tasks [124]. While the unanticipated performance of AutoEncoder may suggest that AutoEncoder could not well-reconstruct the target data and distinguish anomalies for document-level text data. As for reconstruction-based methods, context vector data descriptions (CVDD) [141], which reduces the one-to-one reconstruction distance between r multi-head feature vectors and r prototypical context vectors, can be seen as a recent reconstruction-based approach for one-class text classification. We compared CVDD with other related models in Chapter 4.

And CVDD showed comparable performance compared with other models on three datasets including *TREC*. As for the poor performance of BERT, we also tested BERT with other models. And we will discuss the results and provide some possible explanations in the next section.

In Chapter 3, we formulated Basilisk as special one-class problem and solved it with AutoEncoder. There are still many unanswered questions such as the performance of a more wide range of OCC methods. That would be an extensive work to explore and compare them in future. Besides Basilisk, there exist many different bootstrapping frameworks [133], further research should be undertaken to investigate how to choose appropriate approaches for kinds of tasks.

5.3 Future Work

Finally, we present some directions or open questions for future work.

Regarding the topic of AutoEncoder guided bootstrapping of the semantic lexicon in Chapter 3: We formulated Basilisk, one of *bootstrapping* frameworks, as a special one-class problem. And we proposed methods to incorporate AutoEncoder to improve its evaluation steps. Since most of *bootstrapping* methods follow the similar iterative learning process and start with a small number of seed instances or patterns. It is natural to think that whether the proposed methods can be applied to *bootstrapping* algorithms such as Snowball [1] and WMEB [101]. In recent years, there is an increasing need to acquire kinds of knowledge from row text corpora. Such acquired knowledge, including the semantic lexicons Chapter 3 focused on, is beneficial for numerous application scenarios. In the future, we would like to observe the effects of our methods to learn other types of knowledge such as entity relationships in COVID-19 clinical articles [4, 20]. Since there are a lot of algorithms can be applied for one-class classification tasks [157, 65, 125, 139], so the choice of different methods used for different bootstrapping frameworks and their performance would be a future diction of this study.

Regarding the topic of one-class text classification with multi-modal deep support vector data description in Chapter 4: Pre-trained models, e.g., BERT [23], have shown strong performances on many natural language processing tasks [91, 5]. It deserves to try pre-trained model for one-class tasks. However, for the general

one-class text classification problem, AutoEncoder did not obtain any improvement by using BERT as the choice of feature map, as mentioned in last section. We also tested more models with BERT. However, the results were also not satisfactory, even worse than the average of GloVe [123] word embedding sometimes, so we won't put these unsatisfactory results in the thesis. mSVDD and other related models, including DSVDD [140], OC-NN [14] and DMSVDD [29], also did not get improvement with BERT, as well as CVDD [141] also mentioned. A possible explanation for this might be the *anisotropy* problem which leads to the difficulty of capturing semantic similarity for text data with BERT model [154]. One-class classification algorithms need to make a description of the target class and to determine which (new) test samples resemble the target training [157]. Therefore, OCC models also have to capture semantic similarity for determination. To address the problem, normalizing flow [83] and whitening[154] would be two recent methods to tackle it. The unsatisfactory performances leave an open question for in-depth investigation on the application of pre-trained models for one-class text classification.

Not just BERT, there are many other pre-trained models (e.g., Albert [77]), we would like to explore how to incorporate other pre-trained models for the task of one-class classification for future work. Incorporating some specific negative supervision methods or contrastive learning frameworks to pre-trained models for one-class text classification may be promising [114, 28]

To complete the negative supervision, we proposed a word-level data augmentation method that can construct *pseudo*-negative samples from only given positive target training dataset. The proposed method is just a primary attempt to generate *pseudo*-negative text data for the application of one-class text classification. As for future works, how to generate diverse and appropriate *pseudo*-negative text data using generative adversarial networks (GAN) [33] or other data augmentation techniques deserve to be explored [39, 89].

References

[1] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM Conference on Digital libraries*, pages 85–94. ACM, 2000.

[2] Hany Alashwal, Safaai Deris, and Razib M Othman. One-class support vector machines for protein-protein interactions prediction. *International Journal of Biological and Medical Sciences*, 1(2), 2006.

[3] Eiji Aramaki. *Iryo Gengo Shori (Medical Language Processing)*. Corona Publishing Company, 2017. in Japanese.

[4] Sayantan Basu, Sinchani Chakraborty, Atif Hassan, Sana Siddique, and Ashish Anand. ERLKG: Entity representation learning and knowledge graph based association analysis of COVID-19 through mining of unstructured biomedical corpora. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 127–137, Online, November 2020. Association for Computational Linguistics.

[5] Enkhbold Bataa and Joshua Wu. An investigation of transfer learning-based sentiment analysis in japanese. *arXiv preprint arXiv:1905.09642*, 2019.

[6] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[8] Matthew Berland and Eugene Charniak. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 57–64, 1999.

[9] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[10] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of data*, pages 1247–1250. ACM, 2008.

[11] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988.

[12] Sergey Brin. Extracting patterns and relations from the world wide web. In *International Workshop on The World Wide Web and Databases*, pages 172–183. Springer, 1998.

[13] Sharon A Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 120–126, 1999.

[14] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *CoRR*, abs/1802.06360, 2018.

[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, 2010.

[16] Pai-Hsuen Chen, Chih-Jen Lin, and Bernhard Schölkopf. A tutorial on ν-support vector machines. *Applied Stochastic Models in Business and Industry*, 21(2):111–136, 2005.

[17] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[18] Xiaoran Chen and Ender Konukoglu. Unsupervised detection of lesions in brain mri using constrained adversarial auto-encoders. *arXiv preprint arXiv:1806.04972*, 2018.

[19] James R Curran, Tara Murphy, and Bernhard Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, volume 6, pages 172–180. Australia: Pacific Association for Computation Linguistics., 2007.

[20] Debasmita Das, Yatin Katyal, Janu Verma, Shashank Dubey, AakashDeep Singh, Kushagra Agarwal, Sourojit Bhaduri, and RajeshKumar Ranjan. Information retrieval and extraction on COVID-19 clinical articles using graph community detection and Bio-BERT embeddings. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online, July 2020. Association for Computational Linguistics.

[21] Santanu Das, Bryan L Matthews, Ashok N Srivastava, and Nikunj C Oza. Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 47–56, 2010.

[22] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[24] Huyen Do, Alexandros Kalousis, Jun Wang, and Adam Woznica. A metric learning perspective of svm: on the relation of lmnn and svm. In *Artificial Intelligence and Statistics*, pages 308–317, 2012.

[25] ST Dumais, J Platt, D Heckerman, and M Sahami. Inductive learning algorithms and representations for text categorization. 1998. In *Proceedings of*

CIKM-98, 7th ACM International Conference on Information and Knowledge Management (Bethesda, MD, 1998), pages 148–155, 1998.

[26] Ran El-Yaniv and Mordechai Nisenson. Optimal single-class classification strategies. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 377. MIT Press, 2007.

[27] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

[28] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[29] Zahra Ghafoori and Christopher Leckie. Deep multi-sphere support vector data description. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 109–117. SIAM, 2020.

[30] Zoubin Ghahramani. Unsupervised learning. In *Summer School on Machine Learning*, pages 72–112. Springer, 2003.

[31] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1):83–135, 2006.

[32] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[34] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *A review of machine learning techniques for processing multimedia content*, 1:9–16, 2004.

[35] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? metric learning approaches for face identification. In *2009 IEEE 12th international conference on computer vision*, pages 498–505. IEEE, 2009.

[36] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2013.

[37] Sonal Gupta and Christopher D Manning. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 98–108, 2014.

[38] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[39] Md Akmal Haidar and Mehdi Rezagholizadeh. Textkd-gan: Text generation using knowledge distillation and generative adversarial networks. In *Canadian Conference on Artificial Intelligence*, pages 107–118. Springer, 2019.

[40] Pei-Yi Hao and Yen-Hsiu Lin. A new multi-class support vector machine with multi-sphere in the feature space. In Hiroshi G. Okuno and Moonis Ali, editors, *New Trends in Applied Artificial Intelligence*, pages 756–765, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[41] Wolfgang Härdle. *Applied nonparametric regression*. Number 19. Cambridge university press, 1990.

[42] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[44] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

[45] Simon Heron. Technologies for spam detection. *Network Security*, 2009(1):11–15, 2009.

[46] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[47] Geoffrey E Hinton. Connectionist learning procedures. In *Machine learning*, pages 555–610. Elsevier, 1990.

[48] Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.

[49] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

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

[51] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, pages 84–92. Springer, 2015.

[52] Heiko Hoffmann. Kernel pca for novelty detection. *Pattern recognition*, 40(3):863–874, 2007.

[53] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[54] Matthew Honnibal and Ines Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 2017.

[55] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[56] Junlin Hu, Jiwen Lu, Junsong Yuan, and Yap-Peng Tan. Large margin multi-metric learning for face and kinship verification in the wild. In *Asian conference on computer vision*, pages 252–267. Springer, 2014.

[57] Ruihong Huang and Ellen Riloff. Inducing domain-specific semantic class taggers from (almost) nothing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 275–285. Association for Computational Linguistics, 2010.

[58] Robert A Jacobs. Methods for combining experts' probability assessments. *Neural computation*, 7(5):867–888, 1995.

[59] Nathalie Japkowicz, Catherine Myers, Mark Gluck, et al. A novelty detection approach to classification. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 1, pages 518–523, 1995.

[60] Sujay Kumar Jauhar and Eduard Hovy. Embedded semantic lexicon induction with joint global and local optimization. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 209–219, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[61] Kim Jin-Dong, Nédellec Claire, Bossy Robert, and Deléger Louise, editors. *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, Hong Kong, China, November 2019. Association for Computational Linguistics.

[62] Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009.

[63] Piotr Juszczak. *Learning to recognise: A study on one-class classification and active learning.* PhD thesis, Technische Universiteit Delft, 2006.

[64] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[65] Shehroz S Khan and Michael G Madden. A survey of recent trends in one class classification. In *Irish conference on artificial intelligence and cognitive science*, pages 188–197. Springer, 2009.

[66] Shehroz S Khan and Michael G Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.

[67] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

[68] JooSeuk Kim and Clayton D Scott. Robust kernel density estimation. *The Journal of Machine Learning Research*, 13(1):2529–2565, 2012.

[69] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[70] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*, 2015.

[71] Tetsuo Kiso, Masashi Shimbo, Mamoru Komachi, and Yuji Matsumoto. Hits-based seed selection and stop list construction for bootstrapping. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 30–36. Association for Computational Linguistics, 2011.

[72] Takahiro Kobori, Mikio Nakano, and Tomoaki Nakamura. Small talk improves user impressions of interview dialogue systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 370–380, Los Angeles, September 2016. Association for Computational Linguistics.

[73] M Kraaijveld and Robert PW Duin. Generalization capabilities of minimal kernel-based networks. In *International Joint Conference on Neural Networks.*, volume 1, pages 843–848, 1991.

[74] Bartosz Krawczyk, Michal Wozniak, and Boguslaw Cyganek. Clustering-based ensembles for one-class classification. *Inf. Sci.*, 264:182–195, 2014.

[75] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[76] Raksha Kumaraswamy, Anurag Wazalwar, Tushar Khot, Jude Shavlik, and Sriraam Natarajan. Anomaly detection in text: The value of domain knowledge. In *The Twenty-Eighth International Flairs Conference*, 2015.

[77] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[78] Trung Le, Dat Tran, and Wanli Ma. Fuzzy multi-sphere support vector data description. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 570–581. Springer, 2013.

[79] Trung Le, Dat Tran, Wanli Ma, and Dharmendra Sharma. A theoretical framework for multi-sphere support vector data description. In *International Conference on Neural Information Processing*, pages 132–142. Springer, 2010.

[80] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[81] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[82] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 2013.

[83] Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, November 2020.

[84] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[85] Winston Lin, Roman Yangarber, and Ralph Grishman. Bootstrapped learning of semantic classes from positive and negative examples. In *Proceedings of ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data*, volume 1, page 21, 2003.

[86] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[87] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. Autoencoder for words. *Neurocomputing*, 139:84–96, 2014.

[88] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[89] Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. Data boost: Text data augmentation through reinforcement learning guided conditional generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9031–9041, Online, November 2020. Association for Computational Linguistics.

[90] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[91] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*, 2019.

[92] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[93] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[94] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[95] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[96] Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[97] Alireza Makhzani and Brendan Frey. Winner-take-all autoencoders. *arXiv preprint arXiv:1409.2752*, 2014.

[98] Larry Manevitz and Malik Yousef. One-class document classification via neural networks. *Neurocomputing*, 70(7-9):1466–1481, 2007.

[99] Larry M Manevitz and Malik Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.

[100] Tara McIntosh. Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 356–365, 2010.

[101] Tara McIntosh and James R Curran. Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition. In *Proceedings of the Australasian Language Technology Association Workshop 2008*, pages 97–105, 2008.

[102] Tara McIntosh and James R Curran. Reducing semantic drift with bagging and distributional similarity. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 396–404, 2009.

[103] Aditya Krishna Menon and Robert C Williamson. A loss framework for calibrated anomaly detection. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 1494–1504, 2018.

[104] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[105] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[106] TC Minter. Single-class classification. In *LARS Symposia*, page 54, 1975.

[107] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint*

Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, pages 1003–1011. Association for Computational Linguistics, 2009.

[108] Marie-Francine Moens. *Information extraction: algorithms and prospects in a retrieval context*, volume 21. Springer Science & Business Media, 2006.

[109] Bahram Mohammadi, Mahmood Fathy, and Mohammad Sabokrou. Image/video deep anomaly detection: A survey. *arXiv preprint arXiv:2103.01739*, 2021.

[110] Mary M Moya, Mark W Koch, and Larry D Hostetler. One-class classifier networks for target recognition applications. *NASA STI/Recon Technical Report N*, 93, 1993.

[111] Mikio Nakano and Kazunori Komatani. A framework for building closed-domain chat dialogue systems. *arXiv preprint arXiv:1910.13826*, 2019.

[112] Mikio Nakano, Kazunori Komatani, Kotaro Funakoshi, and Yukiko Nakano. *Taiwa Sisutemu (Dialogue Systems)*. Corona Publishing Company, 2015. in Japanese.

[113] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78, 2004.

[114] Sora Ohashi, Junya Takayama, Tomoyuki Kajiwara, Chenhui Chu, and Yuki Arase. Text classification with negative supervision. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 351–357, 2020.

[115] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

[116] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[117] Mahesh Pal and Giles M Foody. Feature selection for classification of hyperspectral data by svm. *IEEE Transactions on Geoscience and Remote Sensing*, 48(5):2297–2307, 2010.

[118] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[119] Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 113–120. Association for Computational Linguistics, 2006.

[120] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[121] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[122] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[123] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[124] Pramuditha Perera, Ramesh Nallapati, and Bing Xiang. Ocgan: One-class novelty detection using gans with constrained latent representations. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2893–2901, 2019.

[125] Pramuditha Perera, Poojan Oza, and Vishal M Patel. One-class classification: A survey. *arXiv preprint arXiv:2101.03064*, 2021.

[126] Pramuditha Perera and Vishal M Patel. Learning deep features for one-class classification. *IEEE Transactions on Image Processing*, 28(11):5450–5463, 2019.

[127] William Phillips and Ellen Riloff. Exploiting role-identifying nouns and expressions for information extraction. In *Proceedings of International Conference on Recent Advances in Natural Language Processing*, pages 468–473, 2007.

[128] Ashequl Qadir, Pablo Mendes, Daniel Gruhl, and Neal Lewis. Semantic lexicon induction from twitter with pattern relatedness and flexible term length. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2432–2439. AAAI Press, 2015.

[129] Ashequl Qadir and Ellen Riloff. Ensemble-based semantic lexicon induction for semantic tagging. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 199–208, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics.

[130] Asma Rabaoui, Manuel Davy, Stéphane Rossignol, and Noureddine Ellouze. Using one-class svms and wavelets for audio surveillance. *IEEE Transactions on information forensics and security*, 3(4):763–775, 2008.

[131] Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 41–47. Association for Computational Linguistics, 2002.

[132] Ellen Riloff and Rosie Jones. A retrospective on mutual bootstrapping. *AI Magazine*, 39(1):51–61, 2018.

[133] Ellen Riloff and Rosie Jones. A retrospective on mutual bootstrapping. *AI Magazine*, 39(1):51–61, 2018.

[134] Ellen Riloff, Rosie Jones, et al. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pages 474–479, 1999.

[135] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 704–714, 2013.

[136] Ellen Riloff, Janyce Wiebe, and Theresa Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 25–32. Association for Computational Linguistics, 2003.

[137] Stephen J Roberts. Novelty detection using extreme value statistics. *IEE Proceedings-Vision, Image and Signal Processing*, 146(3):124–129, 1999.

[138] Erik Rodner, Esther-Sabrina Wacker, Michael Kemmler, and Joachim Denzler. One-class classification for anomaly detection in wire ropes with gaussian processes in a few lines of code. *training*, 1:1–5, 2011.

[139] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 2021.

[140] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International Conference on Machine Learning*, pages 4393–4402, 2018.

[141] Lukas Ruff, Yury Zemlyanskiy, Robert Vandermeulen, Thomas Schnake, and Marius Kloft. Self-attentive, multi-context one-class classification for unsupervised anomaly detection on text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4061–4071, 2019.

[142] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[143] Saburou Saitoh. *Theory of Reproducing Kernels*, pages 135–150. Springer US, Boston, MA, 2003.

[144] Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580*, 2015.

[145] Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Borja Sanz, and Pablo G Bringas. Spam filtering through anomaly detection. In *International Conference on E-Business and Telecommunications*, pages 203–216. Springer, 2011.

[146] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

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

[148] B. Schölkopf and AJ. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, December 2002. Parts of this book, including an introduction to kernel methods, can be downloaded ¡a href=”http://www.learning-with-kernels.org/sections/”¿here¡/a¿.

[149] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[150] Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

[151] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[152] Stanley Simoes, P Deepak, Munu Sairamesh, Deepak Khemani, and Sameep Mehta. Content and context: Two-pronged bootstrapped learning for regex-formatted entity extraction. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[153] Mark Stevenson and Mark Greenwood. A semantic approach to IE pattern induction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 379–386, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[154] Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint arXiv:2103.15316*, 2021.

[155] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle loss: A unified perspective of pair similarity optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[156] Mihai Surdeanu and Massimiliano Ciaramita. Robust information extraction with perceptrons. In *Proceedings of the NIST 2007 Automatic Content Extraction Workshop (ACE07)*, 2007.

[157] David Martinus Johannes Tax. *One-class classification: Concept learning in the absence of counter-examples.* PhD thesis, Technische Universiteit Delft, 2001.

[158] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[159] Michael Thelen and Ellen Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 214–221. Association for Computational Linguistics, 2002.

[160] Alexandre B Tsybakov et al. On nonparametric estimation of density level sets. *The Annals of Statistics*, 25(3):948–969, 1997.

[161] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[162] Régis Vert, Jean-Philippe Vert, and Bernhard Schölkopf. Consistency and convergence rates of one-class svms and related algorithms. *Journal of Machine Learning Research*, 7(5), 2006.

[163] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[164] Jiří Vomlel. Noisy-or classifier. *International Journal of Intelligent Systems*, 21(3):381–398, 2006.

[165] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1041–1049, 2017.

[166] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(2), 2009.

[167] Janyce Wiebe and Ellen Riloff. Creating subjective and objective sentence classifiers from unannotated texts. In *International conference on intelligent text processing and computational linguistics*, pages 486–497. Springer, 2005.

[168] Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. What's strange about recent events (wsare): an algorithm for the early detection of disease outbreaks. *The Journal of Machine Learning Research*, 6:1961–1998, 2005.

[169] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017.

[170] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.

[171] Yanshan Xiao, Bo Liu, Longbing Cao, Xindong Wu, Chengqi Zhang, Zhifeng Hao, Fengzhao Yang, and Jie Cao. Multi-sphere support vector data description for outliers detection on multi-distribution data. In *2009 IEEE international conference on data mining workshops*, pages 82–87. IEEE, 2009.

[172] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

[173] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural

image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[174] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 189–196. Association for Computational Linguistics, 1995.

[175] Sho Yokoi, Ryo Takahashi, Reina Akama, Jun Suzuki, and Kentaro Inui. Word rotator ' s distance. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2944–2960, 2020.

[176] Hwanjo Yu, Jiawei Han, and KC-C Chang. Pebl: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):70–81, 2004.

[177] Guodong Zhou, Min Zhang, DongHong Ji, and Qiaoming Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 728–736, 2007.

[178] HuiWei Zhou, Long Chen, Fulin Shi, and Degen Huang. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 430–440, Beijing, China, July 2015. Association for Computational Linguistics.

[179] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.

[180] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

Acknowledgments

First of all, I would like to express my sincere appreciation and utmost gratitude to my supervisor Prof. Manabu Okumura for his kind support and guidance during my doctoral study. Thanks to him for giving me an opportunity to join his laboratory. From the beginning, Okumura-sensei guided my study, and provided the insightful advice and valuable comments on my research. Especially, he continued to support me and to help me when I face difficulties. He also patiently helped me to check and modify all papers and responses in person. I could not finish this study without the support of him. He also supported me and other overseas students in many aspects of life in Japan. It is my great honor to be a student of him.

Besides Okumura-sensei, I would also like to gratefully thank the remaining committee members: Prof. Itsuo Kumazawa, Prof. Minoru Nakayama, Assoc. Prof. Takahiro Shinozaki, and Assoc. Prof. Kotaro Funakoshi. Thanks so much for their kind suggestions and very valuable comments on my research and dissertation.

I truly thank Funakoshi-sensei for his guidance and kind help, no matter at Honda Research Institute (HRI) or lab. I always remember the helpful conversation on the subway to Nagatachō. His valuable comments help me to better understand the research and to improve the value of the thesis.

I thank all the lab members, especially Asst. Prof. Hidetaka Kamigaito who helped me to modify paper and to solve many questions in research and phd study. I also thank Dr. Tatsuya Ishigaki who advised me in various aspects including research, phd study, and life in Japan. I thank the help of Tatsuya Aoki for writing response. Thank Mrs. Iiyama for her long time of help and care. I thank many students including Kobayashi-san, Fujita-san, and Kokurin-san, to help on programming and research. Thanks to Sun-san, Zhai-san, Ishigaki-san, Hayashi-san, Watanabe-san, Matsunagi-san, Takimoto-san, Tommy-san, Hitomi-san, Surya-san, and many other labmates who helped me fitting into this lab. I thank Dr. Kikuchi, Dr. Miura, Dr. Makino, and Dr. Lya for their useful suggestions on my research. I would like to thank all the participants for active discussions during lab seminars to refine the study.

I really thank Prof. Hiroya Takamura for his guidance and help for years. Takamura-sensei guided me and provided me valuable comments and suggestions about my research, especially for the second part of work. His strict comments while kind advice

help me raise the level of papers. And I appreciate his rigorous attitude on writing, math, and algorithm in research. I admire his soccer skills and miss the time of playing soccer together in Suzukakedai.

I would like to thank Dr. Mikio Nakano for his support and help in the cooperation with him at HRI. Thanks for his valuable comments and advice to refine our research. I also thank Assoc. Prof. Ryohei Sasano for his help and useful advice when he was at this lab. I thank Prof. Ichiro Kobayashi and Assoc. Prof. Kanako Komiya for their useful comments on different occasions.

I also thank the recommendation of my previous supervisor Prof. Wenping Ma. I would like to thank the support from China Scholarship Council. I would also express my gratitude to my previous supervisor Prof. Jinglu Hu who helps me to continue study and encouraged me when I encounter difficulties. I appreciate the help and useful discussions from Yukun, Jingyi, Dongyuan, and other colleges. I wish they have a successful doctoral/master career. I wish to thank my friends and relatives who help me and care about me.

Finally, I would like to thank my parents and my sister for their understanding and support. All to them.

List of Publications

Journal Papers

- Chenlong Hu, Mikio Nakano, and Manabu Okumura. Autoencoder Guided Bootstrapping of Semantic Lexicon. *Journal of Natural Language Processing*, 27(3):627–652, 2020.
- Chenlong Hu, Yukun Feng, Hidetaka Kamigaito, Hiroya Takamura, and Manabu Okumura. One-class Text Classification with Multi-modal Deep Support Vector Data Description. *Journal of Natural Language Processing*, 28(4), 2021.

Conference Papers (Peer Reviewed)

- Jingyi You, Chenlong Hu, Hidetaka Kamigaito, Kotaro Funakoshi and Manabu Okumura. Robust Dynamic Clustering for Temporal Networks. In *Proceedings of 30th ACM International Conference on Information and Knowledge Management*, 2021.
- Jingyi You, Chenlong Hu, Hidetaka Kamigaito, Hiroya Takamura and Manabu Okumura. Abstractive Document Summarization with Word Embedding Reconstruction. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, 2021.
- Yukun Feng, Chenlong Hu, Hidetaka Kamigaito, Hiroya Takamura and Manabu Okumura. Improving Character-Aware Neural Language Model by Warming Up Character Encoder under Skip-gram Architecture. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, 2021.
- Chenlong Hu, Yukun Feng, Hidetaka Kamigaito, Hiroya Takamura, and Manabu Okumura. One-class Text Classification with Multi-modal Deep Support Vector Data Description. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 3378–3390, Online. Association for Computational Linguistics. 2021

- Yukun Feng, Chenlong Hu, Hidetaka Kamigaito, Hiroya Takamura, and Manabu Okumura. A Simple and Effective Usage of Word Clusters for CBOW model. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 80–86, Suzhou, China, December 2020. Association for Computational Linguistics.

- Chenlong Hu, Mikio Nakano, and Manabu Okumura. Autoencoder Guided Bootstrapping of Semantic Lexicon. In *Proceedings of Pacific Rim International Conference on Artificial Intelligence*, pages 220–233. Springer, 2019.