Large-scale Document Labeling using Supervised Sequence Embedding

A Thesis

Submitted to the Faculty

of

Drexel University

by

Dmitriy Bespalov

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy in Computer Science

November 2012

Dedications

This dissertation is dedicated to my beloved grandmother.

Acknowledgments

First of all, I would like to thank my advisor, Ali Shokoufandeh, for all of his patient mentoring and encouragement during my entire education at Drexel. Ali has helped me at every step of this long journey that I would not have been able to accomplish without his guidance. I would also like to thank researchers at Machine Learning Department in NEC Labs America, Bing Bai and Yanjun Qi, for all of their help with this research during my internships at NEC Labs and thereafter. I would not have been able to complete this research without the computing resources that were generously made available to me at NEC Labs. I am also grateful to other members of my committee, Dario Salvucci and Spiros Mancoridis, for their insightful suggestions and comments that improved this work. Certainly, a very special thanks must go to William C. Regli who introduced me to academic research during my undergraduate education at Drexel.

I am truly thankful to my family and friends for their support throughout this process. I am also grateful to Walt Mankowski and Tom Plick for the time each of them took to carefully proofread my manuscript. I would also like to thank other friends that I have made at Drexel, including Kamelia Aryafar, Chris Cera, Trip Denton, Cheuk Yiu Ip, Ehsan Khosroshahi, Yusuf Osmanlioglu and Evan Sultanik. Finally, this research was made possible thanks to the financial support of grants from various United States government agencies.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Abstract

Large-scale Document Labeling using Supervised Sequence Embedding
Dmitriy Bespalov
Ali Shokoufandeh, Ph.D.

A critical component in computational treatment of an automated document labeling is the choice of an appropriate representation. Proper representation captures specific phenomena of interest in data while transforming it to a format appropriate for a classifier. For a text document, a popular choice is the bag-of-words (BoW) representation that encodes presence of unique words with non-zero weights such as TF-IDF. Extending this model to long, overlapping phrases (n-grams) results in exponential explosion in the dimensionality of the representation. In this work, we develop a model that encodes long phrases in a low-dimensional latent space with a cumulative function of individual words in each phrase. In contrast to BoW, the parameter space of the proposed model grows linearly with the length of the phrase. The proposed model requires only vector additions and multiplications with scalars to compute the latent representation of phrases, which makes it applicable to large-scale text labeling problems. Several sentiment classification and binary topic categorization problems will be used to empirically evaluate the proposed representation. The same model can also encode relative spatial distribution of elements in higher-dimensional sequences. In order to verify this claim, the proposed model will be evaluated on a large-scale image classification dataset, where images are transformed into two-dimensional sequences of quantized image descriptors.

Chapter 1: Introduction

Abstract models for naturally occurring phenomena are at the heart of developing computational methods for solving real-word problems. These models often involve sub-sampling or processing of sensory data that facilitates the procedure to carry out the required computational task. The term *abstract* implies that the model describes properties that are attributed to an entire family of sensory data, rather than specific examples among this family. It is common to refer to an abstract model as a *computational model*, or model for short. Abstract models are usually tailored to particular applications where each model is carefully constructed to capture a known phenomena, often specific to the modality of the signal. More often than not, models can only capture certain aspects of the phenomena, but the theoretical relation to the entire domain of the signal is weak or non-existent. Every model is encoded with a series of symbolically represented abstractions that eventually reduce each model to a form executable by computer. "Abstraction is a quintessential activity of computer science – the intellectual tool that allows computer scientists to express their understanding of a problem, manage complexity, and select the level of detail and degree of generality they need at the moment" (p. 65 [5]).

Before we continue with our presentation, a short overview of the terminology is in order. This dissertation will consider computational problems that process input *sensory data* and carry out prescribed tasks, where the input data is said to be provided in a form of a natural *signal*, such as text, images, audio or other multimedia data. A *computational task*, which will also be referred to as a *computational problem*, describes specific formally defined problem that describes specific goals that a computer *system* or *method* has to perform in order to *compute the task*. Given a specific problem, a *task instance* or a *sample* will refer to a single example of the input sensory data for this task. In the spirit of abstract models, a computational method can be described as a composition of several abstract models, where a set of *parameters* controls computational procedure performed by each model. Furthermore, two varieties of computational models can be identified in many computational methods – *representation* and *task model*. As such, this dissertation takes on the following view of any system that computes a task (please refer to Figure 1.1 for an illustration). Represen-

tation transforms a task sample from a *signal domain* into a *representation domain* in a way that facilitates the computations, performed by the task model, to produce the *task output* for the sample. The performance of computational methods can be expressed in terms of their efficiency and effectiveness on a task. The *efficiency* of a model is expressed in terms of its computational complexity, while the notion of the *effectiveness* quantifies the quality of its performance. The effectiveness measures considered in this dissertation will be formally defined in Section 2.4.

Adaptive Models For brevity and clarity, the so-called adaptive property of computational models will be recognized in this dissertation. Parameters of an *adaptive model* can be updated using a *machine learning* procedure in order to obtain preferred behavior from this model. The learning procedure requires a collection of *training samples*, which amounts to a set of exemplars that encode sought behavior of the model. On the other hand, *non-adaptive* models do not admit a mechanism to bias their parameters towards the preferred behavior. The adaptive property is intentially introduced to describe a family of computational models that *train* (assign or learn) their parameters using supervised, unsupervised, or semi-supervised machine learning approaches. *Supervised learning* is a procedure that trains parameters of a model using a collection of samples, where each instance is assigned with a preferred output. The preferred outputs are often referred to as the *ground-truth labels*, and a sample paired with the corresponding label is referred to as a *supervised sample or instance*. Supervised learning updates the parameters of the model to minimize the discrepancy between the desired output, specified by the ground-truth labels, and the actual output of the model [6]. In *unsupervised learning*, the ground-truth labels are not available for the training samples. The so-called auto-encoders, which will be described in Section 2.2, are often used to initialize the parameters of a model in an unsupervised learning setting. *Semi-supervised learning* refers to a setting when a combination of supervised and unsupervised training is used to learn the parameters of a model. The adaptive property of computational models is analogous to parametric regression analysis that estimates a function, defined by a set of parameters, from observations for independent and dependent variables. The process of estimating the parameters (e.g., ordinary least squares method) that analytically define the sought function (e.g., a polynomial of specified degree) is akin to a supervised learning procedure, where observations are comprised of input and output exemplar pairs that encode preferred behavior for the model.

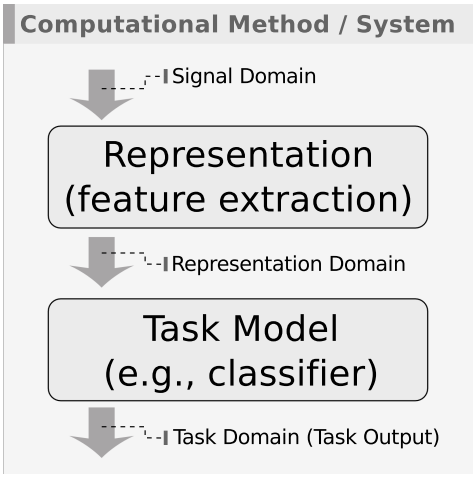


Figure 1.1: Illustration of a typical computational system.

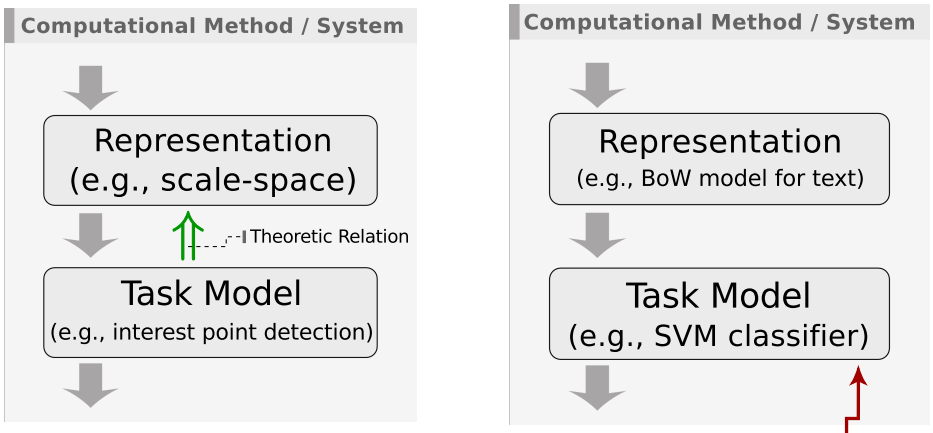
It is common to distinguish representations from task models in general. Indeed, a task model describes computations required to carry out the task. A representation, on the other hand, deals with data organization that enables the task model to tackle the computational problem in the representation domain. For example, the graph data structure is a popular representation in computational geometry or network optimization problems. Search trees, which are used in relational databases, are another example of representations. When dealing with sensory data, a representation dictates how the information from the input signal is processed, organized, stored and retrieved. In general, the goal of representations is to convert sensory data into a form that retains information relevant to a prescribed phenomena and at the same time abates the computations induced by the task model.

Consider the design of a band-pass filter for a discretized audio signal. Here, the intensity values of a continuous audio signal, sampled at fixed intervals (e.g. 44 kHz), constitute the domain of the signal. It is unclear how to implement the task model for the band-pass filter in the signal domain. Thus, a model for representation is required to transform a task sample into the appropriate representation domain. Discrete-time Fourier transform represents an audio signal as a function of time, as well as its frequency. Consequently, a model for the band-pass filter can now be devised in the time-frequency representation domain. For each time window (interval), the frequency coefficients outside of the band spectrum are set to zero, and the remaining coefficients are transformed back to the signal domain and returned as the task output.

Meaningful representations offer potential benefits when dealing with computational problems that handle sensory data. The process of converting a signal domain into a representation domain is often referred to as *feature extraction*, which is a popular term in the image processing and pattern recognition communities. This dissertation will consider the notion of feature extraction as a generalized concept which is applicable to a variety of representations that deal with sensory data. In Section 1.1, several models for feature extraction will be discussed and their common properties will be highlighted. Specifically, the notion of top-down and bottom-up representations will be introduced. A *bottom-up* representation is a direct consequence of a theoretical framework, developed to tackle a specific task (see Figure 1.2a). Conversely, the representation is said to be *top-down* when no theoretic relation between feature extraction and a task model can be established. Top-down feature extraction, which is often formulated under certain assumptions about the sensory data, converts the input signal to match the input format of the task model. In the analogy with regression analysis, bottom-up representation would be equivalent to the case when the regression function is known or derived as an implication of a theoretical framework. And a top-down feature extraction is similar to regression analysis when the function is unknown, so a variety of analytical forms of the sought regression function would have to be considered on a case-by-case basis.

This dissertation will focus on automatic *document labeling* tasks that are usually tackled as supervised *classification* problems. In general, document labeling automatically assigns one or more labels to a given text, where the labels are drawn from a finite set of category labels. Three examples of these tasks will be discussed in Section 1.3. As will be shown in Section 1.1, most existing representations for document labeling are non-adaptive. Thus, the only adaptive component in a document labeling system is its task model, which is often referred to as *classifier*. Popular classification models used in a variety of document labeling problems will be discussed in Section 1.2. The main hypothesis of this thesis states that replacing a non-adaptive representation with a novel adaptive model will result in more effective document labeling. The main motivation for our hypothesis is that the adaptive feature extraction will learn more meaningful representations of sensory data, which in turn will improve labeling performance. In Section 1.2 a further motivation of our hypothesis will be provided.

The rest of this chapter is organized as follows. In Section 1.1 a representation for text, called the bag-of-



(a) System with a bottom-up representation

(b) System with an adaptive task model and a non-adaptive top-down representation

Figure 1.2: Computational methods with **(a)** bottom-up and **(b)** top-down representations. Feature extraction in a *bottom-up* representation is a direct consequence of a theoretical framework, developed to tackle a specific task. In a *top-down* representation, no theoretic relation between feature extraction and a task model can be established. Most existing representations for document labeling are non-adaptive. Thus, the only adaptive component in a document labeling system is its task model, which is often referred to as the *classifier*.

words model, will be discussed. It is one of the most popular choices for many document labeling problems. In addition, several varieties of representations will be introduced, and their applications and common properties will be identified. Then, a discussion of the shortcomings of non-adaptive representations will be presented. This will motivate a novel adaptive model for feature extraction which is the main contribution of this dissertation. In Section 1.2 an overview of classifiers that are applicable document labeling problems will be presented. In Section 1.3 we will review labeling tasks and our experimental setup to empirically validate our thesis. Section 1.4 will concludes the chapter with a brief overview of the manuscript.

1.1 Feature Extraction

In this section several models for feature extraction will be considered. Intuitively, feature extraction transforms sensory data into a reduced set of features so the task model can compute the task efficiently. However, it is also common to treat sensory data itself as a stream or a sequence of *features*, where each feature is

encoded with a symbolic or a real value. For example, a discretized audio signal is a stream of real-valued features – i.e., intensity values. A natural language text is a sequence of enumerable symbolic features – i.e., single words or unigrams. We say that feature extraction transforms a set of features from the signal domain into a set of features in the representation domain. As such, five groups of representations can be identified, based on the way these models handle features in the signal domain: feature selection, feature detection, feature generation, feature embedding, and feature clustering.

We say that a representation performs *feature selection* in the signal domain when only a subset of original features are used in the representation domain. A representation with *feature generation* computes a new feature space induced by the input sensory data, while *feature detection* identifies certain phenomena in the sensory data and treats this information as features in the representation domain. Representation with *feature embedding* maps the sensory data into low-dimensional space so that a specific property of the signal domain is preserved. Finally, *feature clustering* performs clustering of signal samples, and centroids are mapped as features in the representation domain. Ideally, any representation "promotes" or "generates" features that are effective for the given task. In what follows, examples of representations will be presented and their approaches to handling features in the sensory data will be discussed. In addition, the adaptive and top-down/bottom-up properties of representations will also be identified.

Interest point detection in *scale-space theory* [7] is an example of representation with *feature detection* that encodes visual information in images at various scales. The motivation for generating a *scale-space representation* for images is rooted in the fact that the size of real-world objects in a captured scene depends on the scale of the observation. However, computational systems that analyze images are often required to deal with visual signals where the scale of observations is not known ahead of time. Scale-space framework deals with this phenomena by representing an image as a family of gradually smoothed images. The scale-space representation automatically selects locally appropriate scale for the observed image structures. It was shown that detecting the local maxima (and minima) of a scale-normalized Laplacian operator allows one to identify the location and size of maximally stable blob regions in the scene [7]. These stable blob regions are sometimes referred to as interest points. This property of the scale-space representation is a direct consequence of the so-called *causality principle* [8], stating that new image structures (or features) are not

created in the representation as the scale parameter is increased. It follows that the scale-space representation performs feature selection in the image domain to identify maximally stable blob regions that are invariant to translation, rotation and rescaling in the image domain [9]. In other words, scale-space theory provides us with a formal framework that defines the image representation in terms of features which are invariant to the scale of the captured scene. There are other examples of computational problems where theoretic relationship between a task model and a representation can be established. *Rate-distortion theory* gives rise to a representation with *feature generation* for lossy signal compression. The task of lossy signal compression can be viewed as a representation that maps features from the original signal domain to a new domain of "compressed" features (i.e. codes), such that the representation length of a signal in the new domain is reduced. The representation length of a signal is defined as the number of bits required to store the signal in each domain. The lossy nature of the task allows for imperfect reconstruction of the original signal from the compressed features. Rate-distortion theory yields an analytical expression that relates the compression ratio to the distortion introduced due to lossy nature of the compression [10–12]. The compression is measured in terms of the ratio between original and compressed representation lengths. The distortion is defined as an mean squared error between the original and the reconstructed signals. The task of lossy signal compression can then be formulated as an optimization where the objective is to minimize the distortion while maximizing the compression ratio. As a result, the solution to the rate-distortion optimization yields an adaptive model for feature extraction with feature generation and guaranteed performance on the lossy signal compression task [13–15]. Lossy signal compression is an adaptive model, since signal samples are required to solve rate-distortion optimization, which in turn maximizes the quality of the solution for the lossy compression task (i.e., minimize distortion and maximize rate).

In the two examples of representations we have considered so far, the scale-space and rate-distortion theories allow each task to formally define the corresponding representations. Intuitively, in these models the accuracy of the representation is influenced by the quality of the eventual outcome that solves the computational problem. Whenever it is possible to establish theoretic relation between a task model and a model for feature extraction, we refer to such representation as *bottom-up*. Unfortunately, in some cases no such connections can be established. For instance, in the so-called transform-based compression system, the repre-

sentation relies on pre-defined procedures that map the original features into the compressed domain. Fourier, discrete cosine or wavelet transforms are popular choices for these procedures [16]. It is possible to bound the distortion introduced by transform-based compression for various domains of natural signals [16]. However, the transform-based representations are non-adaptive, and feature generation in these models is not influenced by specific task instances. We refer to such representations as *top-down*. Lossy compression methods that use adaptive representations based on the rate-distortion optimization are known to result in better compression than the transform-based methods [17, 18]. However, the gain in compression ratio usually comes at a significant computational cost.

Computing scale-space representation and rate-distortion optimization are both computationally tractable procedures. Unfortunately, this is not always the case for bottom-up representations. An example of such a scenario is the signal reconstruction problem known as *sparse coding*. Intuitively, sparse coding attempts to find a small number of representative patterns (or atoms) for features in the input sensory data, such that a linear combination of the atoms reproduces the original signal. Given a vector space representation $\mathbf{y}$ of the input data and matrix $\mathbf{A}$, we are interested in finding $\mathbf{x}$ that minimizes

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|\mathbf{x}\|_0 \\ \text{s.t.} \quad & \mathbf{y} = \mathbf{A}\mathbf{x}, \end{aligned} \tag{1.1}$$

where $\|\mathbf{x}\|_0$ is the L_0 norm that denotes number of non-zero components in vector $\mathbf{x}$, and overcomplete dictionary $\mathbf{A}$ is $N \times M$ matrix where $M \gg N$. Optimization (1.1) computes a bottom-up representation for input signal $\mathbf{x}$ with respect to basis system $\mathbf{A}$. Solving (1.1) in general is known to be computationally intractable [19]. However, an approximate solution can be obtained using an L_1 relaxation that replaces L_0 in (1.1) with the L_1 norm or $\|\mathbf{x}\|_1$, which denotes the sum of the absolute values of entries in $\mathbf{x}$. Donoho and Huo [20] identified certain conditions (i.e., when "sufficiently sparse" $\mathbf{x}$ exists) that result in ideal atomic decomposition – solution to the convex optimization with L_1 relaxation is the unique and optimal solution to (1.1). Various pursuit methods for solving (1.1) have been proposed [21–24] that compute sparse representation. There exists biological evidence suggesting that most natural signals such as images [25] or audio [26]

admit sparse representations [27]. In fact, formulating a task as a sparse representation problem has been successfully applied in several applications, including blind source separation [28], image decomposition [29] and denoising [30]. When overcomplete dictionary is fixed, we say that sparse coding performs feature selection from $\mathbf{A}$. Feature generation can also be introduced to the representation, when a custom overcomplete dictionary is learned [31].

Rank-minimization (RM) optimization is a generalization of the sparse coding problem. In the optimization, the objective is to minimize the rank of the sought matrix $\mathbf{H}$ such that $\mathcal{A}(\mathbf{X}) = \mathbf{H}$, where $\mathcal{A}(\cdot)$ is a linear map applied to the input signal $\mathbf{X}$. RM optimization gives rise to yet another bottom-up representation with feature selection that computes the "simplest" possible explanation of the phenomena, described by map $\mathcal{A}(\cdot)$. This is a popular bottom-up representation in image in-painting that reconstructs damaged positions of an image by recovering intensities of the missing pixels. Using a 2D autoregressive model, one can arrange observed and missing pixels into a Hankel matrix $\mathbf{H}$. Computing the matrix $\mathbf{H}$ of minimum rank will then produce the sought image in-paining [32]. The RM optimization in general is also known to be intractable [33]. Recht *et al.* [34] showed that when the restricted isometry property holds for map $\mathcal{A}$, a solution to the non-convex RM optimization can be recovered by solving a convex optimization where the objective is set to the nuclear norm $\|\mathbf{H}\|_*$. The nuclear norm of the matrix $\mathbf{H}$ is a defined as the sum of singular values of $\mathbf{H}$.

In computational methods with bottom-up representations, the mechanism that sets the parameters of the model is a direct consequence of a theoretic correspondence between the sought outcome for the method and its representation. The task implies the procedure for feature extraction (within a formal framework) in bottom-up representations. Recognizing this property may lead to the development of computational systems that are more effective for the task. However, systems that use top-down representations, coupled with adaptive task models, may be equally effective. The parameters of the task models in such systems are usually trained using supervised learning methods, and the construction of the top-down representations are often based on a specific heuristic or an educated guess. In addition, for some top-down representations it is possible to estimate its effectiveness on a whole task or a group of tasks, which is the case in transform-based lossy compression [16]. For other models, their effectiveness can only be established via empirical evaluation

performed on individual instances of the task. Quite often, tasks that involve adaptive models are evaluated empirically. For example, in the case of a document labeling task we evaluate the performance using a fixed set of samples, where each instance is a pair of text and its corresponding (ground-truth) label. Every method for document labeling is evaluated on the set with a measure that compares predictions of the method with the ground-truth labels. The measure returns a single number that quantifies the effectiveness of the method. This allows one to qualitatively compare the performance of different systems on this labeling task. In Section 1.3 we discuss several performance measures that are considered in this dissertation.

A popular top-down representation used in methods applicable to document classification and retrieval systems is the *bag-of-words* (BoW) model. The BoW maps a free text into a high-dimensional vector space parameterized by a finite vocabulary of features, where only single words or unigrams are used as features. The coefficients for every feature are defined as a function of word frequencies in the text. Consequently, BoW treats a text as an unordered collection of features, and builds a frequency distribution of the unigrams as the primary means for labeling or retrieval problems. The non-adaptive BoW representation can be coupled with a classifier, which is an adaptive task model used in variety of labeling problems. The parameters of the classifier are usually biased using supervised instances for the labeling task. However, the parameters of the BoW model do not participate in the supervision, as the representation for text is obtained during pre-processing with coefficients assigned using one of the weighting heuristics (e.g., TF-IDF, BM25, etc). This makes the BoW representation non-adaptive.

The main contribution of this thesis is in designing adaptive representations for sentiment analysis and other labeling problems. Intuitively, *sentiment analysis* is a labeling problem which aims at predicting whether expressed opinion in a text is positive, negative, or neutral. The motivation for the proposed representation is our hypothesis that adaptive feature extraction will result in more effective methods for tackling document labeling problems, especially the problems that benefit from encoding *phrase semantics* in the representation domain. This hypothesis is due to our observation that short phrases are less ambiguous in terms of their sentiment. For example, the term *good* often appears in positive online reviews, but "not very good" is less likely to appear in positive comments. When using the BoW representation, the proximity of "not", "good" and "very" in the text is ignored. The other facet of sentiment is the degree of positivity or

negativity of long phrases. As one such example, consider the compounding effect of the phrase "... terrible terrible terrible ..." in contrast to three dispersed "terrible" in a text. As a result, modeling short phrases in the representation domain may result in more effective methods for sentiment analysis, as well as for document labeling problems in general. A popular remedy is to extend the bag-of-words model to encode phrase semantics in the representation domain by incorporating n-grams (a contiguous sequence of n words) as features. This latter model is referred to as the *bag-of-n-grams* (BoN) [35]. Augmenting the BoW model with n-grams[1] allows BoN to encode the local spatial configuration (or ordering) of unigram features in the representation.

Extending the BoW model to incorporate n-grams (for $n >= 3$) adversely effects complexity of the feature space, since the dimensionality of a BoN vector, $\|\mathcal{D}\|^n$, grows exponentially as a function of n, and each n-gram requires a parameter in the classifier model. For instance, extending an English word vocabulary $\mathcal{D}$ of size $\|\mathcal{D}\| = 10,000$ by including the bigrams ($n = 2$) and trigrams ($n = 3$) will add up to $\|\mathcal{D}\|^2 = 10^8$ and $\|\mathcal{D}\|^3 = 10^{12}$ additional features to BoN, respectively. As a result, a pre-processing procedure that performs feature selection [36] is used before forming the BoN vectors. The basic idea of the *feature selection heuristic* is to retain a small subset of n-gram features based on a scoring function (statistics) that the BoN representation can recognize. All other phrases are ignored during the construction of BoN vectors. The motivation is that scoring function is chosen so that only effective features for the given task will be retained. Popular feature selection heuristics used in classifying text include Information Gain [37], Chi-Square statistics [38], Mutual Information (MI) [39], Optimal Orthogonal Centroid feature selection [36], as well as a generalized framework recently proposed by Jing *et al.* [40]. The effectiveness of feature selection heuristics is often task-dependent and requires empirical evaluation for each classifier model considered. It is worth noting that some feature selection pre-processing can take advantage of the supervised samples in their computations. For instance, MI-based feature selection computes the scoring function based on mutual information between unique n-grams and ground-truth labels [1]. However, the supervised samples in this case are used to estimate feature-label statistics in a way that is invariant to the specific task. Furthermore, the ground-truth labels are not considered during the actual construction of BoN

[1] We will use "n-gram" and "phrase" interchangeably.

representations, so BoN with feature selection pre-processing is also a non-adaptive model (see Figure 1.4a for an illustration). To be fair, some bottom-up representations, such as supervised variants of dictionary learning for sparse coding [41–44], can benefit from supervision as well.

This dissertation develops a novel adaptive top-down representation which is an alternative to the non-adaptive BoN model. As we shall see from our empirical evaluations, the proposed model performs feature selection among unigram features that promotes effective phrases for document labeling tasks. In addition, our representation encodes unigram features in a low-dimensional latent space and constructs short phrases from latent embedding vectors of the unigrams. We refer to this representation as the *Supervised Sequence Embedding* (SSE). Supervision is used to bias parameters of SSE, hence the proposed model is adaptive. Furthermore, the total number of features that an SSE model can recognize grows linearly with the phrase length n. In contrast, the feature space in the BoN representation grows exponentially as a function of n. We believe the SSE model is a viable alternative to BoN when dealing with various document labeling problems that benefit from encoding short phrases in the representation domain. The proposed representation encodes spatial configuration of the features from the signal domain using supervision. In that, our model is similar to convolutional neural networks (CNNs) that give rise to adaptive representations as well. However, SSE is designed to handle *sparse or quantized* features in the signal domain, while CNNs assume the input signal is represented as a sequence of dense feature vectors. We discuss CNNs in more details in Chapter 2. It is also worth noting that, by construction, a free text is a sequence of symbolic features that can be encoded with sparse vectors. Indeed, a unique word (or a phrase) can be encoded with a *word selector*, a high-dimensional canonical basis vector with a single non-zero entry that corresponds to the feature id assigned to the word (or the phrase). In addition to free text, images can be encoded as two-dimensional sequences of quantized features, namely, the histograms of quantized image descriptors. As a result, the SSE model can be used for data in other modalities such as images. We defer the discussion of the application of SSE to images until Chapter 5. The experimental results that we obtain in this dissertation suggest our model may yield a more effective representation for some large-scale text and image classification tasks. We describe our experimental setup in Section 1.3.

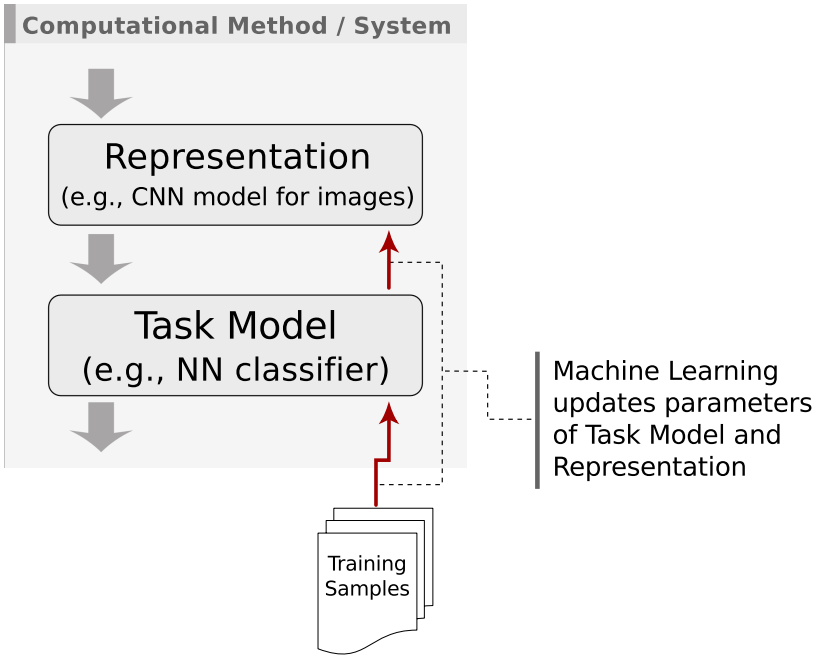
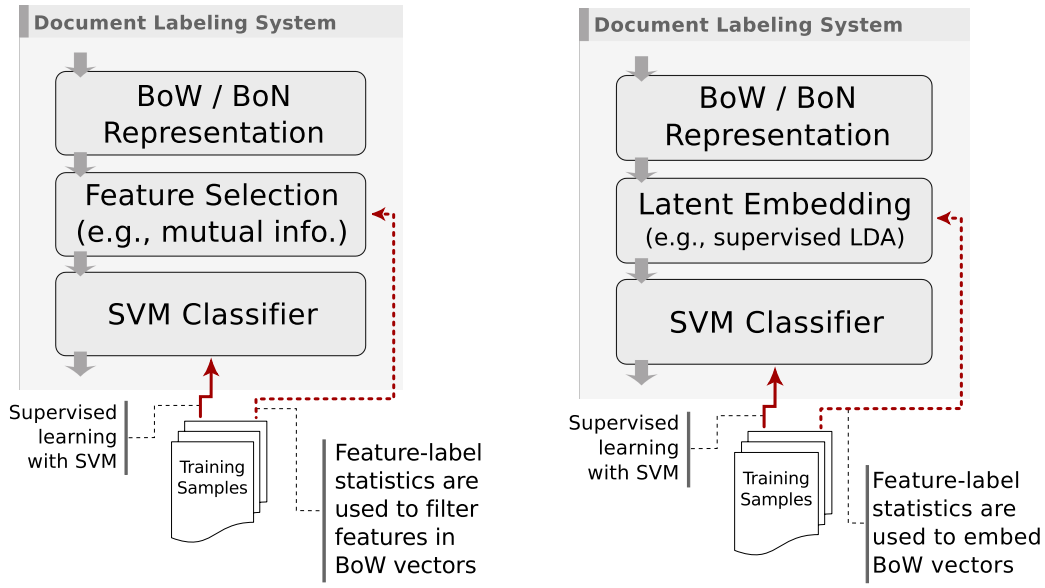


Figure 1.3: Computational method with an adaptive top-down representation. Machine learning updates both sets of parameters in the task model and the representation.

1.2 Adaptive Models for Classification

The focus of this dissertation is on document labeling or classification problems that automatically assign one or more of a finite set of labels to a document (e.g., text or image). We refer to a classification task with two labels as *binary*, while *multi-class* refers to the setting with three or more labels. In addition, *multi-label* setting refers to classification tasks where multiple labels can be assigned per document.

In Section 1.1 we discussed the BoN model, which is a popular non-adaptive top-down representation in document labeling systems. BoN maps a free text into a high-dimensional vector space, and a classifier model is used to compute the labeling task from the representation domain. However, combining a representation and a classifier task may not be sufficient to tackle the problem effectively. For example, it is not clear how to set the parameters of a sentiment classification system so that it captures the intricate behavior of semantic reasoning. Semantic reasoning, although often subjective and task-dependent, allows human readers to agree with one another when tasked with a manual sentiment classification task. Hence, auxiliary knowledge or supervised samples may be required to tackle the labeling task effectively. The supervised samples are used in conjunction with a *supervised learning* (or training) procedure to bias parameters of the system so it produces



(a) BoW model with feature selection pre-processing.

(b) Latent Embedding for BoW model (e.g., LSI [45] or supervised LDA [46]).

Figure 1.4: Document labeling with the BoW representation.

output that closely matches the ground-truth labels. We say that a system is *supervised* if it requires supervised learning to tune parameters in at least one of its models. In general, a supervised learning can be formulated as a *numerical optimization* over the parameter space in the adaptive model, where the optimization objective is a function that quantifies the agreement between the preferred and the actual behavior of the system. In addition, supervised samples often result in a set of convex constraints for the parameter space. The main premise of supervised learning is that the preferred behavior of the system will extrapolate to the unseen instances of the task which are not present among the samples that were used to train the system.

The details of the supervised models used in prior-art document labeling systems will be discussed in Section 2.3. A typical classifier utilizes features in the representation domain as the primary evidence for the labeling task. Due to computational complexity, some of these models (e.g., non-linear SVM) can benefit from a feature reduction procedure, such as feature selection heuristics that were discussed in Section 1.1. A different approach to reduce the dimensionality of the representation domain is to use *feature embedding*, also commonly referred to as *latent embedding*. For example, a latent embedding procedure can be used to map a BoW domain to a low-dimensional (dense) feature space while capturing synonymy among the features.

Latent semantic indexing (LSI) [45] is a popular embedding method based on the singular value decomposition of a feature-document co-occurrence matrix that computes both document and feature projections into a shared low-dimensional space. An important extension of LSI is probabilistic LSI (pLSI) proposed by Hofmann [47]. The pLSI model assumes each document is a mixture of a small number of latent topics and each feature is attributed to one of the document's topics. Despite its solid statistical foundation, pLSI is not a proper generative model of an unknown document [48]. To alleviate this shortcoming, Latent Dirichlet Allocation (LDA) [49] introduced the notion of a Dirichlet prior associated with per-document topic distribution. We note that pLSI and LDA are closely related, that is, pLSI has been shown to be a maximum a posteriori estimated LDA model under a uniform Dirichlet prior [48]. It has been reported that the pLSI and LDA methods do not provide significant advantage (if any) over LSI on a number of standard text classification benchmark datasets [50]. In addition all three methods (LSI, pLSI, LDA), being unsupervised, are invariant to the labels associated with a specific task. Blei and McAuliffe [46] proposed a supervised extension to a generative LDA model that includes a response variable for document labels. It is important to note that any latent embedding can be viewed as yet another abstract model. In a document labeling system, such latent embedding is arranged as an additional component between the BoN representation and a task model (see Figure 1.4b). Furthermore, the parameters of the latent embedding and task models are not biased in a unified framework. In other words, the parameters of the latent embedding model are estimated during pre-processing, and the latent embedding parameters remain fixed during the supervised training of the task model.

We propose to unify an adaptive representation (SSE) and a classifier in a single framework that we apply to several document labeling problems. The SSE model is a *feature embedding* approach that learns low-dimensional representation of text phrases. The proposed system embeds unigram features in a low-dimensional space, where construction of n-grams take place. Combining adaptive SSE with a classifier allows us to jointly train the parameters of both models and compute the labeling task more effectively. The empirical evidence that was obtained suggests our system outperforms standard BoN baseline systems on several large-scale document classification tasks.

When dealing with large-scale classification problems, the system is often required to dynamically adapt

to new samples without expensive *retraining* on the entire set of supervised samples. The so-called *online learning* provides this advantage over the models trained with *batch learning*. The latter requires complete retraining of the parameters when a new supervised sample is introduced. Support vector machines (SVM) are a popular example of a classification model trained with *batch learning*. In large-scale classification problems even first-order iterative optimization methods that adopt a batch learning model may pose a significant computational burden. An example of an online learning method is Stochastic gradient descent (SGD), which approximates the "true" gradient of the objective with a gradient at a single data point selected at random from the training set. To minimize the objective, SGD then updates the parameters with a small step in the direction of the negative gradient. It has been proven that, when the objective is quasi-convex, SGD will converge to the optimal solution when training with an infinitesimally small step size and infinitely many data points [51]. In practice, the convergence of online learning models rivals batch learning, especially when a large training dataset is available [52]. The step size (a.k.a., *learning rate*) in the SGD method is often subject to some empirical selection procedure to ensure better convergence.

We implement the proposed system for document classification as a deep feed-forward neural network which is trained online with SGD that takes advantage of backpropagation to jointly bias parameters of all components in the system. Deep architectures that received increasing attention in recent years have been used to learn complicated functions in natural language processing and computational vision [53]. Each layer in the architecture encodes features at different levels of abstraction, defined as a composition of features computed at the previous layer. Glorot *et al.* [54] utilized a deep learning model to extract the representation of each text review in an unsupervised fashion using stacked Denoising Auto-encoders. With the learned high-level feature representation, they claim to achieve state-of-the-art performance for domain adaption tasks using sentiment classification data. Socher *et al.* [55] used recursive neural networks to perform simultaneous parsing and classification of both text and image data. In addition, multi-layered neural networks have been successfully used for learning statistical language models that estimate conditional probability distribution for word sequences [56, 57].

The proposed system combines a representation in the form of latent n-gram and document embedding, as well as a classification model in a unified framework where each component is adaptive and fully-supervised.

It is our belief that the latent n-gram embedding (i.e., the SSE model) is a form of feature learning that retains phrase semantics in the representation domain. In addition SSE is closely related to convolutional neural networks (CNN). Similar to the SSE model, as will be discussed in Section 2.2, CNNs give rise to top-down adaptive representations. However, the former handles sequences of quantized features, while CNNs are designed to deal with dense features (e.g., raw pixel intensities in images) in the signal domain. We further elaborate on the similarities, as well as differences, between CNNs and SSE in Chapter 2.

In Section 2.3 an overview of related task models for document labeling will be presented. These models formulate supervised learning as an optimization problem where supervised samples induce a set of constraints for parameters of an adaptive classifier. Let Ψ_1 denote a parameter set for the classifier model. Then, supervised learning will solve the following optimization:

$$\min_{\Psi_1} \mathcal{L}(\Psi_1), \tag{1.2}$$

where $\mathcal{L}(\cdot)$ is often referred to as the *loss functional*. Furthermore, when the adaptive model for feature extraction is used, supervised learning solves the following optimization:

$$\min_{\Psi_1, \Psi_2} \mathcal{L}(\Psi_1, \Psi_2), \tag{1.3}$$

where Ψ_2 denotes a set of parameters for the adaptive representation. From one perspective, the adaptive representation can encode information in its domain (e.g., semantics of phrases), which is tailored to the specific labeling task at hand. This information would then amount to evidence that the classifier can leverage to achieve the higher labeling accuracy. In other words, higher degree of freedom in the parameter space of optimization (1.3) will enable the supervised system to achieve better *generalization*, i.e., to predict labels of unseen samples more accurately. From another perspective, when only a few supervised samples are available for training, additional parameters Ψ_2 in the optimization (1.3) may result in *overfitting*, thus causing the generalization of the system to deteriorate. Overfitting occurs in adaptive models when their underlying parameters encode noise in training samples, rather than phenomena exhibited by these exemplars. As a result, since the loss function $\mathcal{L}(\cdot)$ is not a measure of the effectiveness of the system, in the event of overfitting

the accuracy of the label predictions for unseen instances can decrease significantly. The notion of overfitting is best explained with a polynomial regression analysis example, where the number of data observations is lower than the degree of the polynomial regression function. In this case, the regression function that passes through all observation points is estimated.

It is worth noting that multi-layered CNN models are prone to overfitting as well, since CNNs usually contain a large set of parameters to train. However, as will be discussed in Section 2.2, so-called auto-encoders can be used to initialize parameters of CNNs in an unsupervised fashion so fewer supervised samples are needed to train these models. Indeed, it is common to pre-train the parameters in one CNN layer at a time while descending deeper into the hierarchy using samples without ground-truth labels. We believe that SSE can admit an auto-encoder as well, which is an immediate issue we plan to address in our future work. Unlike CNN models that rely on multiple convolutional projection layers, SSE requires a single projection to encode n-grams for document labeling. In addition, only large-scale datasets were used to empirically validate SSE. As a result, we were able to learn an effective representation for text using SSE in a completely supervised fashion. To test the hypothesis that adaptive SSE is superior to the non-adaptive BoN model, we performed an empirical evaluation of SSE using three document labeling problems. A description of these document labeling tasks will be provided in Section 1.3. Section 1.4 will conclude this chapter with an overview of the remaining chapters of this dissertation.

1.3 Experimental Studies

We evaluate the proposed adaptive representation for quantized features using three varieties of document labeling problems: sentiment analysis, topic categorization and image classification. We use the broader notion of the term *document* to denote a free text or an image. For both modalities of the input signal, we use a non-adaptive bag-of-words representation as a main baseline, since this representation is known to achieve state-of-art classification performance on all three tasks. We discuss the baseline representation for text in Chapter 2, and defer the introduction of the BoW model for images until Chapter 5. To carry out the labeling task, we consider several classification models that we discuss in Section 2.3, in conjunction with both the baseline and the proposed SSE representations. This allows us to empirically validate the merit of the proposed adaptive representation compared to the non-adaptive models used in the current state-of-the-art

methods. In the remainder of this section, we describe the experimental setup that we used for each of the three tasks.

Varieties of the *sentiment classification* [58] problem predict sentiment polarity (e.g., positive, negative, or neutral) of text, as well as sentiment strength (e.g., 5-star review scale). In this dissertation, we consider binary and multi-class formulations for the sentiment analysis. We evaluate the performance on these tasks using two large-scale sentiment datasets: Amazon [1] and TripAdvisor [2]. Both datasets contain user-generated reviews where an overall sentiment for each review is quantified with an integer 1 through 5. A sentiment score of 1 star corresponds to the lowest (negative) sentiment, while a score of 5 stars corresponds to the highest (positive) sentiment. The TripAdvisor dataset contains neutral reviews (rated with 3-stars), while neutral reviews were omitted during the construction of the Amazon dataset by their authors. To make the evaluation complete, we consider binary classification into positive or negative sentiment, and star-scale classifications, which is missing in most published studies. We use four available labels (1,2,4 and 5 stars) to evaluate sentiment prediction on both datasets. In addition, we also consider a classification task with five labels (i.e. stars), since TripAdvisor contains neutral reviews as well. We populate the BoN baseline representation using n-grams with $n \in \{1, 2, 3, 5\}$, and use TF-IDF [59] to assign weights to BoN features. For both sentiment datasets, we follow the feature selection pre-processing used in [1] to limit the number of features in the BoN representation by retaining n-grams with the highest mutual information (MI) shared by the binary labels (positive or negative).

The second problem that we consider in our experiments is *topic categorization*. Topic categorization problems also deal with the classification of text documents, and identify a set of topics that an input text can describe. We use the Reuters dataset (RCV1) [3], which contains news articles, labeled with 103 topics. For topic categorization, we only consider the binary classification setting, and restrict our evaluations to the four topics with the largest number of positive examples in RCV1: CCAT (ALL Corporate-Industrial), GCAT (All Government and Social), MCAT (ALL Securities and Commodities Trading and Markets), and C15 (Corporate and Industrial Performance). Again, we populate the BoN baseline representation using n-grams with $n \in \{1, 2, 3, 5\}$ and TF-IDF weights. However, during feature selection pre-processing for RCV1 we limit BoN features to the most frequent n-grams.

It is also worth mentioning that raw text in all three datasets (Amazon, TripAdvisor and RCV1) was pre-processed, before extracting n-grams and building BoN representation. Specifically, all non-alphanumeric characters were "padded" with a whitespace characters on each side. In addition, all words were converted to lower case, and then all numbers were replaced with a special word "NUMBER". As an example, consider a sample sentence from a TripAdvisor review: "We stayed one night in the Sand Villa, in a room on the 2nd floor overlooking the pool." After pre-processing, the following text is obtained: "we stayed one night in the sand villa , in a room on the NUMBERnd floor overlooking the pool ." The same procedure was also used to pre-process the input data for the proposed representation.

Image classification assigns labels to an input image, where each label corresponds to an object that may appear in the image. We consider image classification in a multi-class single-label setting. The problem is evaluated using the dataset from the Image-Net Large Scale Visual Recognition Challenge 2011 (ILSVRC2011) [60]. The dataset contains images of 1000 categories of objects, where each category corresponds to a synset (set of synonymous nouns) in the WordNet taxonomy [61]. The categories are organized as leaf nodes in a hierarchy that corresponds to a subset of WordNet's synset hierarchy. The images are converted into a quantized signal, namely, a high-dimensional histogram of appearance features, using a bag-of-features (BoF) approach known as locality-constrained linear coding (LLC) [62]. For the baseline representation we use a non-adaptive (unsupervised) model called spatial pyramid matching (SPM) that was proposed by Lazebnik *et al.* [63]. SPM concatenates feature vectors, computed by partitioning each image into regions at multiple scales, to obtain an image-level vector representation. The proposed SSE method is a adaptive (supervised) alternative to the SPM procedure. We discuss the BoF image representation, and provide details about encoding with SSE two-dimensional sequences of quantized features, in Chapter 5

1.4 Overview

The rest of this dissertation is organized as follows. A detailed overview of the related approaches to text representation and classification will be presented in Chapter 2. The supervised sequence embedding (SSE) that encodes long phrases in text documents will be proposed in Chapter 3. The empirical evidence presented in Chapter 3 will suggest that SSE is a viable alternative to the bag-of-words representation when dealing with large-scale text classification. In Chapter 4 the supervised feature extraction in the proposed model will

further be investigated. Specifically, its ability to capture pseudo-subjectivity of phrases will be illustrated. In addition, an extension to the supervised sequence embedding that encodes global spatial distribution of phrases in text documents will be studied in Chapter 4. Finally, Chapter 5 will discuss an application of SSE to encode local spatial configuration of quantized features in images. This image representation will be benchmarked against a standard non-adaptive model called spatial pyramid matching using a large-scale image classification dataset.

Chapter 2: Background

Formally, a document can be defined as a sequence of features of an arbitrary length, where each feature is described with a single symbolic or real value, or a vector of values. It is common to recognize two forms of features in the signal domain. The features, described with symbolic values, are referred to as *sparse* or *quantized* features, while *continuous* or *dense* features are described with real values. A discretized audio signal is an example of dense features. Indeed, in the time domain, an audio signal can be considered as a one-dimensional sequence of real-valued intensity values. Similarly, pixel intensities in the image domain are an example of dense features organized in a two-dimensional sequence. In contrast, free text is naturally comprised of the *quantized* features, where a feature corresponds to a single word (unigram) or an n-gram (i.e., n consecutive words). Given a collection of text documents, all unique unigrams in the corpus are enumerated and placed into a *word dictionary*. Let $\mathcal{D}$ denote the underlying word (unigram) dictionary and $\mathcal{S}$ denote a set of all finite length sequences of words from $\mathcal{D}$. In addition, let $\Gamma_n = \Gamma(n) \subset \mathcal{S}$ denote a *vocabulary of n-grams* in the corpus, which is a collection of all unique phrases with at most n words. A sequence $\gamma_j = (w_j, w_{j+1}, \ldots, w_{j+n-1})$, with $n < N$, corresponds to an n-gram with offset j in a text document $\mathbf{x} = (w_1, \ldots, w_N)$. Furthermore, every $w_i \in \mathcal{D}$ corresponds to a so-called *selector* vector that encodes the word as a canonical basis vector $\mathbf{e}_{w_i}$, where

$$\mathbf{e}_{w_i} = \Big(0, \ldots, 0, \underset{\text{at index } w_i}{1}, \ldots, 0\Big)^\top. \tag{2.1}$$

Thus, a document $\mathbf{x} = (w_1, \ldots, w_N)$ can be viewed as a sequence of canonical vectors $(\mathbf{e}_{w_1}, \ldots, \mathbf{e}_{w_N})$. The notion of quantized features can be generalized to encode each feature with a sparse vector (or a histogram).

The proposed supervised sequence embedding (SSE) model is designed to efficiently map sequences of such quantized features into the representation domain. Using one of the so-called *signal quantization* procedures, it is possible to compute quantized features from the dense features in the signal domain. As such, the SSE model can also handle images or audio data comprised of dense features. The discussion of quanti-

zation procedures is deferred until Chapter 5, where application of the SSE model to image classification is discussed. Until then, text is assumed to be the input sensory data. In this chapter an overview of the related prior art systems for document labeling is presented. A document labeling problem is often formulated as *sequence classification*, where the primary goal of a representation is to process the arbitrary-length sequence into a manageable form where classification can be carried out. Document labeling systems can be grouped into four categories, based on the type of the representation involved: 1. *vector space*; 2. *document similarity*; 3. *statistical*; 4. *deep architecture or perceptron* models. Vector space models compress input texts into finite-dimensional vectors, so label inference can be carried out with a supervised classifier such as support vector machines. A representation is statistical if the underlying document labeling system is formulated as a probabilistic framework. Document similarity models encode each document in terms of its similarity to other documents in a corpus. Finally, a representation for text can be implemented as a deep architecture such as feed-forward neural network.

From a different perspective, supervised classifiers in document labeling methods can be categorized as *generative* or *discriminative*. A generative model is a full probabilistic model of all its parameters. Thus it is possible to simulate (i.e., generate) the values of any variable in that model. The Naïve Bayes [64, 65] classifier is one of the most popular generative models for document labeling tasks. Naïve Bayes estimates the joint distributions of word features and category labels to obtain the probability of a document belonging to each class, thus reducing classification to selecting the most probable class(es). Discriminative models estimate a distribution of their output variables conditional on the observed input quantities and thus the framework can only be used to sample the output variables for a given input (e.g., to make a prediction for a task instance). On the other hand, since discriminative models do not need to estimate the distribution of the input variables, they can generally express more complex relationships between the input and output quantities. As a result discriminative methods [66, 67] often perform better than generative models [68–70] on text classification tasks [71]. Amongst discriminative classifiers, support vector machines (SVM) [72] are one of the most popular models for the labeling problem.

The rest of this chapter is organized as follows. The related representations, including the bag-of-words model, are discussed in Section 2.1. An introduction to deep architectures and an overview of convolutional

neural networks (CNN) are presented in Section 2.2. An adaptive representation for dense features based on a hierarchy of CNNs is discussed. In addition, an adaptive representation for quantized features called *Lookup Temporal Convolution* (LTC) is presented. The LTC model for text data, first proposed by Collobert and Weston [73], was used to tackle a variety of word-level classification tasks for Natural Language Processing. The LTC model handles quantized features in text and relies on a convolutional projection layer to obtain latent embedding of n-grams. This dissertation adapts this model to document labeling problems, and then demonstrates that the SSE model improves upon LTC in terms of efficiency and effectiveness. Next, a discussion of the four types of classifiers used in document labeling systems is presented in Section 2.3. An overview of the evaluation methodologies, presented in Section 2.4, concludes this chapter.

For clarity of the presentation, an overview of some notations is in order. Throughout the manuscript, vectors or matrices will be denoted with boldface font (e.g., $\mathbf{G}$ or $\mathbf{b}$), and cursive script will be used for scalar variables and functions (e.g., M or $h(\cdot)$). The cardinality of a set will be denoted with $|.|$. Whenever appropriate, operator $\cdot$ will be used to emphasize the multiplication that involves scalar variables. Let $\mathcal{Y} = \{1, ..., C\}$ denote a set of category labels, and $\mathcal{X} \subset \mathcal{S}$ denote a collection of supervised (i.e., manually labeled) samples for a labeling task, where $\mathcal{X} = \{(\mathbf{x}_i, y_i)_{i=1,...,L} \mid \mathbf{x}_i \in \mathcal{X} \;\&\; y_i \in \mathcal{Y}\}$ and $|\mathcal{X}| = L$.

2.1 Related Representations

Most of the classifiers used for document labeling, discussed in Section 2.3, are supervised, while representations in these systems are non-adaptive, such as the BoW model. The BoW representation can be formulated as a *vector space* or a *statistical* model. BoW treats input as an unordered collection of unigram features, where each unigram is assigned with a weight, computed as a function of the word frequencies in the text. The word dictionary is a finite set, even if one considers all words in English language. However, the length of the text can be arbitrary – from a few words to a myriad. As a result, the principle goal of BoW is to deal with arbitrarily many features in a free text.

The BoW representation was originally developed for a statistical document labeling system, based on the Naïve Bayes (NB) classifier [74]. The NB classifier builds the frequency distribution of the unigrams, as the primary means for document labeling. As discussed in Section 2.3, NB makes two assumptions about the input text data – *invariance to feature location* and *conditional independence*. The first assumption,

which is implied by the term "bag" in the name of the representation, ignores positions of unigrams in a text. The conditional independence implies that appearance of each word (i.e., unigram) in a text is an independent event, given labels assigned to the document. Both of the NB assumptions, which keep the NB classifier tractable, will be formalized in Section 2.3. In addition, as will be discussed, Hidden Markov Models (HMM) can be used as an alternative to BoW in a probabilistic document labeling framework. The HMM model allows one to capture local dependency between features, which gives rise to a representation for n-grams in text.

The BoW representation for the NB classifier later inspired the *vector space* BoW representation. The vector space BoW model does not make the assumption about the conditional independence of the features, which is often violated in practice. However, the model is invariant to the locations of features. In other words, the vector space BoW representation also treats a document as an unordered collection of features. The vector-space BoW forms a high-dimensional vector for an input text. Specifically, for a text $\mathbf{x} = (w_1, \ldots, w_N)$, the BoW model uses a unigram dictionary to map $\mathbf{x}$ to $\tilde{\mathbf{e}}_{\mathbf{x}}$, a $|\mathcal{D}|$-dimensional representation:

$$\tilde{\mathbf{e}}_{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{e}_{w_i}. \tag{2.2}$$

It is a common practice to replace the sole non-zero entry of $\mathbf{e}_{w_i}$ with the inverse document frequency (IDF) of the word w_i in $\mathbf{x}$. As a result, the vector $\tilde{\mathbf{e}}_{\mathbf{x}}$ in (2.2) takes the form of a TF-IDF weighting. Specifically, for a document $\mathbf{x}$ from collection $\mathcal{X}$, every unigram $w_i \in \mathbf{x}$ will be assigned a TF-IDF weight, using the formula:

$$\text{tf-idf}(w_i, \mathbf{x}, \mathcal{X}) = \frac{1}{|\mathbf{x}|} \cdot \text{tf}(w_i, \mathbf{x}) \cdot \text{idf}(w_i, \mathcal{X}),$$

where

$$\text{idf}(w_i, \mathcal{X}) = \log \frac{|\mathcal{X}|}{|\{\mathbf{x}_j \in \mathcal{X} : w_i \in \mathbf{x}_j\}|},$$

and $\text{tf}(w_i, \mathbf{x})$ denotes the frequency of $w_i \in \mathbf{x}$ (i.e., the number of times w_i appears in $\mathbf{x}$). Alternative schemas for assigning weights to features in BoW representation exist. For example, BM25 or BM25F were shown to out-perform traditional TF-IDF on document retrieval tasks [75, 76].

The empirical evidence presented in Chapter 3 suggests that certain document labeling tasks can benefit

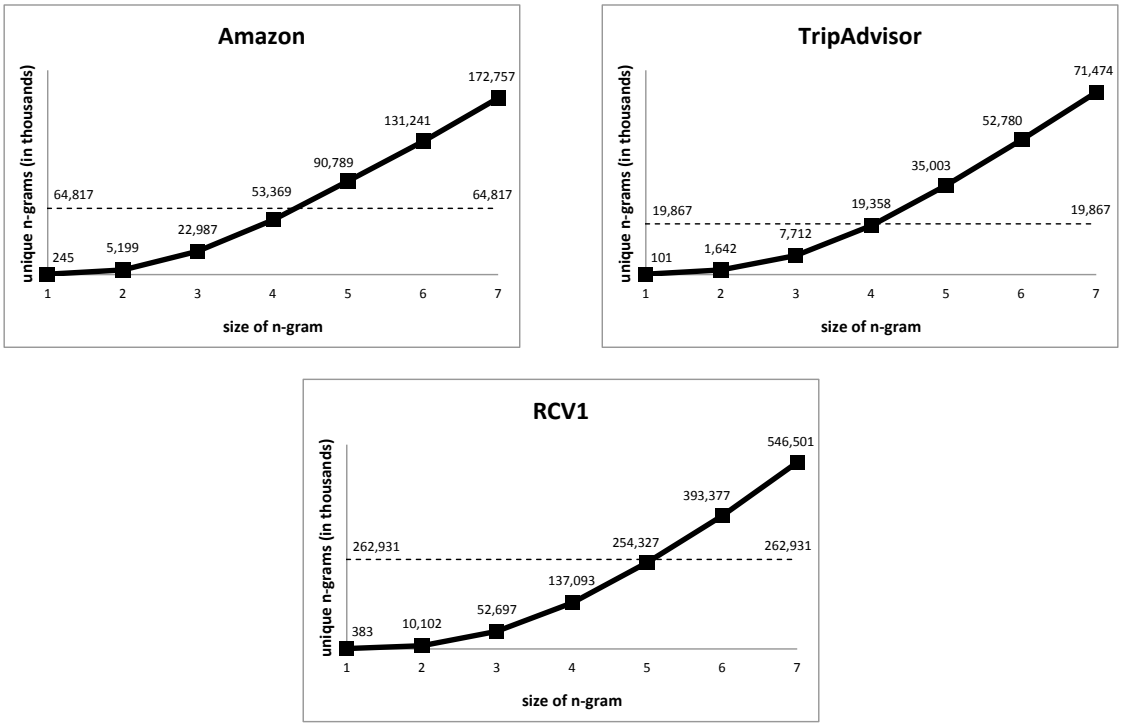


Figure 2.1: Number of unique n-grams in the Amazon [1], TripAdvisor [2] and RCV1 [3] datasets. The unique n-gram count is denoted with $|\Gamma_n| = O(|\mathcal{D}|^n)$ in the manuscript. Dashed lines indicate the length of each dataset (i.e., the total number of words in all of its documents). Numbers are in thousands.

from recognizing sequences of features (i.e., phrases or n-grams) in the representation domain. In order to mitigate the invariance to feature location in BoW, as discussed in Section 1.1, a natural extension is to include short phrases, or n-grams, as additional features in the representation domain [76]. All unique phrases of at most n words are added into the feature space. As a result, the bag-of-$n$grams (BoN) representation maps a free text into a $|\Gamma_n|$-dimensional vector space, so we have $\phi_{\text{BoN}}(\mathbf{x}) \equiv \tilde{\mathbf{e}}_{\mathbf{x}} \in \mathbb{R}^{|\Gamma_n|}$, where $|\Gamma_n| = O(|\mathcal{D}|^n)$ [35]. In the rest of the manuscript, the map $\phi(\cdot)$ will denote a representation and the map $g(\cdot)$ will denote a classifier.

The number of unique phrases in the Amazon [1], TripAdvisor [2] and RCV1 [3] datasets are provided in Figure 2.1. The number of unique n-grams in $|\Gamma_n|$ is bounded by the the total number of all phrases in a dataset. Let $\mathcal{N}$ denote the total number of all words (i.e., not just the unique ones) in the dataset. Then, the dataset contains at most $n \cdot \mathcal{N}$ phrases of n or fewer words. Thus, the number of unique n-grams is bounded by $n \cdot \mathcal{N}$ so we have $|\Gamma_n| \leq n \cdot \mathcal{N}$. This observation implies that the growth of phrases in Γ_n will eventually

become linear when n is increased. The numbers of Figure 2.1 clearly illustrate this phenomena. On the other hand, the total number of n-grams may be huge for $n \geq 3$ when dealing with a large corpus – e.g., there are over 52 million unique trigrams ($n = 3$) in RCV1. As such, a feature selection heuristic is required to keep the dimensionality of the BoN representation tractable in the case of a large-scale document labeling task.

In addition to vector space and statistical models, BoW can also be used in *document similarity*-based systems. In a document similarity model, the representation domain is implicitly defined in terms of pairwise document similarity values. In other words, such representation assigns numerical similarity scores to pairs of documents. The similarity values are usually computed from the features in the signal domain. As a result, the vector space BoW model gives rise to a document similarity representation. Let $\phi_{\text{BoW}}(\mathbf{x}_i) \equiv \tilde{\mathbf{e}}_{\mathbf{x}_i} \in \mathbb{R}^{|\mathcal{D}|}$ and $\phi_{\text{BoW}}(\mathbf{x}_i) \equiv \tilde{\mathbf{e}}_{\mathbf{x}_j} \in \mathbb{R}^{|\mathcal{D}|}$ denote the BoW representation for two documents $\mathbf{x}_i$ and $\mathbf{x}_j$. A similarity measure can then be defined as a function of $\tilde{\mathbf{e}}_{\mathbf{x}_i}$ and $\tilde{\mathbf{e}}_{\mathbf{x}_j}$, for example $\tilde{\mathbf{e}}_{\mathbf{x}_i}^{\top} \tilde{\mathbf{e}}_{\mathbf{x}_j}$. The majority of the document similarity methods are due to so-called *string kernels*. String kernels compare sequences of quantized features such as amino acids [77, 78] or text documents [79]. String kernels are generally designed so that a high similarity score implies that the two sequences (i.e., n-grams) have many features (i.e., unigrams) in common. As such, a critical component in string kernel research is the implementation of inexact matching between short sequence segments. These approaches give rise to a family of mismatch kernels [77]. String kernel approaches can be used to assign similarity scores to pairs of documents. As discussed in Section 2.3, pairwise similarity scores are sufficient to train a supervised classifier, such as support vector machine, to carry out document labeling.

Similar to BoN, representations due to string kernels are non-adaptive. String kernels are manually designed to produce the desired (inexact) matching of sequences, and their parameters can be biased using supervision. To be fair, there exists a number of adaptive prior art models for feature extraction. These representations can efficiently handle the so-called curse of dimensionality when modeling phrases. Such models for feature extraction are often implemented within a Hopfield net or a Boltzmann machine framework, and are best explained in the context of deep neural network architectures, presented in Section 2.2.

2.2 Deep Architectures

"Deep learning" architectures have received increasing attention over the past decade and have proved useful for learning complicated functions toward high-level feature abstractions in natural language processing (NLP) and computational vision [53]. The term "deep" refers to the length of processing chains that make up deep models. Intuitively, every module in the processing chain represents features at a different level of abstraction, defined as a composition of lower-level features. Some of the successful applications of deep architectures include text classification [73], language modeling [56] and object recognition in images [80, 81].

The modules or layers in the processing chain of a deep model are referred to as activation functions. Typical examples of activation functions are linear projections of the form $\mathbf{p} = \mathbf{A}\,\mathbf{x} + \mathbf{b}$ or non-linear transfer functions, such as $\tanh(\cdot)$ and sigmoid$(\cdot)$ that convert the unbounded range of the input into $[-1, 1]$ and $[0, 1]$, respectively. Deep architectures can be trained in both discriminative or generative frameworks. A deep generative model is referred to as a *Boltzmann machine*, which is trained using The Gibbs sampling method [82]. Formally, a deep discriminative model is referred to as a *Hopfield network*. Hopfield networks are often called *neural networks* or *perceptrons*.

A popular supervised method for training deep neural networks (NN) is called *error backpropagation* [83], which can be seen as a generalization of the delta rule. The activation functions at every NN module can be written in a more general form of multi-level functions:

$$g(\mathbf{x}_i) = f_T(f_{T-1}(...(f_1(\mathbf{x}_i))...)),$$

where $g(\mathbf{x}_i)$ denotes a task output for an individual sample $\mathbf{x}_i$. A document classification can then be described using the map

$$g:\ \mathbb{R}^{|\mathcal{D}|} \rightarrow \mathcal{Y}$$

that assigns a category label from the set $\mathcal{Y}$ to a document in the BoW representation. Let $\mathcal{L}(g(\mathbf{x}_i), y_i)$ denote a *loss functional* evaluated on a single supervised sample $(\mathbf{x}_i, y_i)$. The loss functional measures the discrepancy between the estimated and the preferred (i.e., ground-truth) output for the task. In the case of a regression task which assigns scalar real values to input samples x_i, $\mathcal{L}(g(\mathbf{x}_i), y_i) = \|g(\mathbf{x}_i) - y_i\|_2^2$ is a

possible choice for the loss functional, where $g(\mathbf{x}_i) \in \mathbb{R}$.

If the loss functional $\mathcal{L}$ and the activation functions $f_k,\ \forall k \in [1, T]$ are at least once differentiable, the derivative of $\mathcal{L}(\mathbf{x}_i, y_i)$ with respect to parameters in each activation function can be computed in a bottom-up fashion using backward recurrence. As a result, *gradient descent* can be used to update all NN parameters in the direction of the negative gradient at every layer which minimizes the objective $\mathcal{L}(\mathbf{x}_i, y_i)$. Specifically, let $\boldsymbol{\theta}_k$ denote a set of parameters in a NN layer f_k for all $k \in [1, T+1]$, where $f_{T+1} \equiv \mathcal{L}$ is the loss functional. The derivative of $\mathcal{L}$ with respect to the parameters $\boldsymbol{\theta}_k$ can be written in the following form:

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k} = \frac{\partial \mathbf{f}_{T+1}}{\partial \mathbf{f}_k} \frac{\partial \mathbf{f}_k}{\partial \boldsymbol{\theta}_k},$$

where $\frac{\partial \mathbf{f}_{T+1}}{\partial \mathbf{f}_k}$ is recursively calculated using

$$\frac{\partial \mathbf{f}_{T+1}}{\partial \mathbf{f}_k} = \frac{\partial \mathbf{f}_{T+1}}{\partial \mathbf{f}_{k+1}} \frac{\partial \mathbf{f}_{k+1}}{\partial \mathbf{f}_k},$$

and the terms $\frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_k}$ and $\frac{\partial \mathbf{f}_k}{\partial \boldsymbol{\theta}_i}$ denote the Jacobian matrices.

Perceptrons can be trained in a batch or online learning mode. In batch learning, the "true" gradient of the loss functional is calculated using all supervised samples for every update to the NN parameters. Stochastic gradient descent (SGD) is an online learning method for training perceptrons. Instead of calculating the true gradient of the objective with all of the labeled samples, SGD computes the gradient with a randomly chosen training sample and updates all NN parameters accordingly. This training procedure is depicted in Algorithm 1, which will be presented in Section 2.4. SGD is known to be scalable and has been shown to rival batch learning when dealing with large-scale datasets [52]. In contrast, when only a few labeled samples are available, batch learning may out-perform learning with SGD.

It should be noted that SGD minimizes the objective locally and as a result, proper initialization of the parameters in the deep model can affect the performance of the system. A possible remedy to this issue is to use *auto-associators*, also called *auto-encoders*, to initialize the parameters at every NN layer. Intuitively, an auto-encoder estimates the inverse of the projection operator at every NN layer. For instance, for a layer with linear projection $\mathbf{p} = \mathbf{A}\,\mathbf{x} + \mathbf{b}$, an auto-encoder estimates its inverse $\hat{\mathbf{x}} = \mathbf{A}_{\text{inv}}\,\mathbf{p} + \mathbf{b}_{\text{inv}}$. The parameters $\mathbf{A}_{\text{inv}}$

and $\mathbf{b}_{\text{inv}}$ are updated to minimize the error between input $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$, e.g., $\|\hat{\mathbf{x}} - \mathbf{x}\|_2$ [84]. Auto-encoders initialize parameters of the perceptron using unsupervised training – labels assigned to task instances are not considered during the training. It is possible to obtain a large collection of unsupervised samples for certain problems, since no manual labeling is often required. As a result, initializing NN parameters with auto-associators may improve the effectiveness of the labeling system whenever only few supervised task instances are available. For example, Glorot *et al.* [54] utilized stacked auto-encoders in an NN to obtain a latent embedding of text documents using only unsupervised samples. The representation was later used for sentiment classification. In addition to discriminative frameworks, auto-encoders can also be defined for a generative model. Semantic hashing, proposed by Salakhutdinov and Hinton [85], is one example of generative auto-associators used to compute 128 bit hash codes for text documents that preserve semantic relevance between the texts. In other words, the representation maps documents to memory addresses, so semantically similar texts are assigned to nearby addresses. This allows for fast retrieval of related documents for a given query. Semantic hashing is pre-trained as a generative model using Gibbs sampling [82], and the initialized weights are then used to form an NN auto-encoder which is fine-tuned using backpropagation. Both stages in training of this model are completely unsupervised.

Despite the success of the auto-encoders for parameter estimation, these models primarily rely on the non-adaptive BoW representation, as in the case of the aforementioned examples [54, 85]. As such, auto-associators can not be used to improve the quality of the representation. In other words, models for feature extraction in these systems do not contain the parameters which can be tuned via gradient descent. It is worth noting that several adaptive alternatives to BoW can be implemented within a deep architecture. For instance, Bengio *et al.* [56] proposed to learn a statistical language model that approximates the conditional probability distribution for n-grams: $\Pr(w_t \mid w_{t-1}, \ldots, w_{t-n+1})$. Namely, they learned embedding vectors $\mathbf{e}_{w_i}$ for every unigram w_i in the first layer of their representation model. The conditional probability distribution was estimated in the second layer of the system, computed as a function of $\text{vec}\big(\mathbf{e}_{w_t}, \mathbf{e}_{w_{t-1}}, \mathbf{e}_{w_{t-n+1}}\big)$, where the notation $\text{vec}(\cdot)$ denotes an operator that concatenates input vectors into a single column vector. In other words, projection defined on the low-dimensional embedding vectors of individual words was used to model n-grams in this representation. The parameters of the text representation developed by Bengio *et al.* [56] are

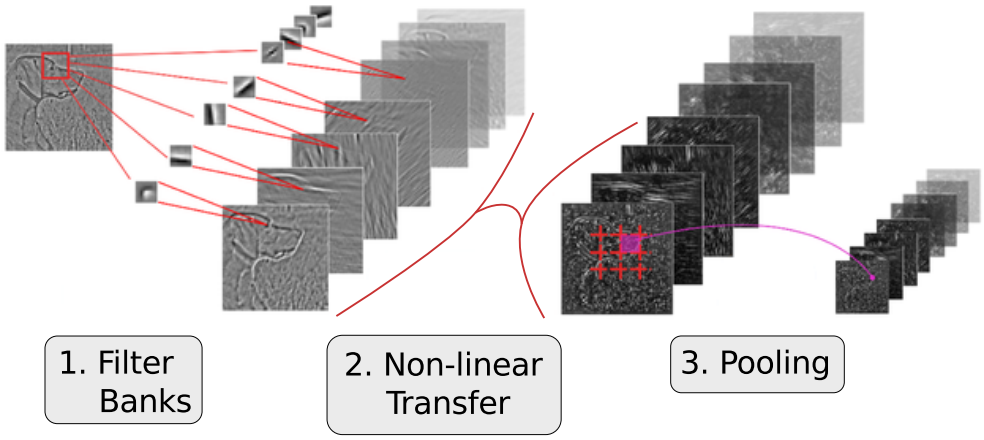


Figure 2.2: An illustration of a typical CNN "stage" for image representation [4]. Each stage in a CNN representation for images consists of three layers: 1. filter banks; 2. rectification and contrast normalization; 3. pooling. Image copyright holder: Koray Kavukcuoglu, `http://koray.kavukcuoglu.org/research.html`

tunable via supervision. This representation was later used by Morin [57] to compute a hierarchical language model with the information extracted from the WordNet taxonomy. A different adaptive representation was proposed by Socher *et al.* [55] that computes a low-dimensional representation of a document using individual embedding vectors of the unigrams. Instead of modeling phrases directly in the representation, a document-level embedding $\mathbf{d_x}$ of text $\mathbf{x} = (w_1, \ldots, w_N)$ is obtained using the so-called recursive auto-encoder. The projection operator is defined recursively

$$\mathbf{d_x}^{(i)} = \mathbf{A}\,\mathrm{vec}\Big(\mathbf{d_x}^{(i-1)}, \mathbf{e}_{w_i}\Big) + \mathbf{b},$$

where $i \in \{1, \ldots, N\}$, $\mathbf{d_x} = \mathbf{d_x}^{(N)}$, and $\mathbf{d_x}^{(0)} = \mathbf{0}$. The representation, computed with recursive auto-encoders, was successfully applied to perform simultaneous parsing and classification of both text and image data [55], as well as to predict sentiment distribution in a semi-supervised setting [86]. It is worth noting, as will be described later in the section, that an approach similar to [56] is used in the *Lookup Temporal Convolution* (LTC) model to tackle a variety of word-level classification tasks for Natural Language Processing [73].

Convolutional neural networks (CNN) [87] can be arranged into a multi-stage representation model for

continuous sensory data, such as audio, image and video data [88]. The CNN representation is a top-down adaptive model with feature generation and selection. The representation assumes the input data is comprised of the dense features. For example, let us consider a case of a CNN representation for grayscale images, where an input signal is a 2D array $\mathbf{x} \in \mathbb{R}^{w \times h}$ of dense features – i.e., pixel intensities. An illustration of a typical stage in CNN representation can be found in Figure 2.2. The stage in a CNN representation consists of three activation functions or layers: 1. a filter bank layer, 2. a non-linear transfer function, 3. a feature pooling layer [88]. A filter bank layer computes an embedding of all patches of $m \times m$ pixels in an input image $\mathbf{x}$ using projection operators

$$\mathbf{p}_k = \mathbf{A}_k * \mathbf{x} + \mathbf{b}_k,$$

where $*$ denotes a 2D discrete convolution operator, each 2D matrix $\mathbf{A}_k$ is referred to as a filter kernel, and $k = \{1, \ldots, K\}$. In other words, the filter bank layer computes multiple latent embeddings p_k of the input $\mathbf{x}$. Popular choices for the non-linear transfer layer include $\tanh$, rectified sigmoid, as well as a variety of shrinkage, truncation, rectification, and normalization operators [89]. Pooling layers, which perform sub-sampling of filter bank features, compute image representation at coarse resolutions. The layer aggregates features over local regions in an input sample, which effectively reduces the resolution in the representation domain. Element-wise $\max(\cdot)$ or mean$(\cdot)$ functions, applied to each region independently, are often used at the pooling layer.

One of the main motivations behind the multi-layered approach in the CNN-based model for image representation is to deal with the unknown scale of observations in images, which impedes processing image structures as features in the representation domain. In this, the CNN representation follows a common approach that encodes visual information at multiple scales. Indeed, at every stage of the CNN model, pooling layers compute image representation at coarser levels. Similarly, the scale-space representation [7], discussed in Chapter 1, encodes image structures at different scales with a family of gradually smoothed images to detect the appropriate scale for each structure of interest. The consecutive application of multiple 3-layer stages in the CNN representation to an input $\mathbf{x}$ will result in a latent image representation denoted with $\mathbf{d}_\mathbf{x} \in \mathbb{R}^M$. It should also be noted that parameters of CNN layers can be initialized with an appropriately defined auto-associator [89].

The CNN representation for images is motivated by the structure of the mammalian visual cortex [90]. The *simple cells* in the cortex are orientation-selective neurons that respond to lines at certain angles, while *complex cells* combine the responses from the simple cells that form the more complex representation of the visual signal. Conceptually, the feature pooling layer in CNN models resembles the complex cells that aggregate the responses from the simple cells. The filter bank layer can then be seen as a collection of simple cells, each activated for specific features in the input. Instead of hand-crafting the parameters for each $\mathbf{A}_k$ used in the CNN representation, all of the parameters of the model are jointly optimized using the backpropagation algorithm. As such, the CNN representation is a top-down adaptive model with feature generation in the filter bank layers, and feature selection in the pooling layers. CNN representation can be combined with a task model to form an end-to-end system where parameters of both models are jointly trained to compute the task. To the best of our knowledge, such an end-to-end method was first successfully applied to handwritten digit recognition by LeCun *et al.* [91]. CNN-based methods for tackling other vision problems were later introduced. Naturally, CNN-based methods were applied to optical character recognition (OCR) [92, 93], including texts in Arabic [94] and Chinese [95]. In addition, CNN models were successfully used for long-range obstacle detection in the DARPA-sponsored LAGR challenge for off-road autonomous vehicle navigation [96]. Other applications of CNN models include real-time face detection [97] and logo recognition [98].

Lookup Temporal Convolution The goal of the LTC representation is to design a mapping of text $\mathbf{x}$ into a low-dimensional latent space. We represent $\mathbf{x}$ in terms of its constituting n-grams using a sliding window mechanism, and embed the n-grams into a latent space. The embedding of text $\mathbf{x}$ will be formed as a combination of the embedding of its individual n-grams. Once the latent embedding of $\mathbf{x}$ is obtained, one of the classification models, discussed in Section 2.3, can be used to carry out the document labeling task. For brevity and clarity, we denote the document embedding with $\phi_{\text{LTC}}(\mathbf{x}) \equiv \mathbf{d}_{\mathbf{x}}$, where $\mathbf{d}_{\mathbf{x}} \in \mathbb{R}^M$. The mapping of a phrase γ_j into a latent space is denoted with $\varphi_{\text{LTC}}(\gamma_j) \in \mathbb{R}^M$. To avoid confusion, we drop the bias terms in our definitions for the embedding projections.

The LTC model implements the mapping $\varphi_{\text{LTC}}(\cdot)$ as a two-stage process. The first stage of $\varphi_{\text{LTC}}(\cdot)$ computes embedding vectors for individual unigrams in $\mathbf{x}$. Similarly to the representation proposed by Ben-

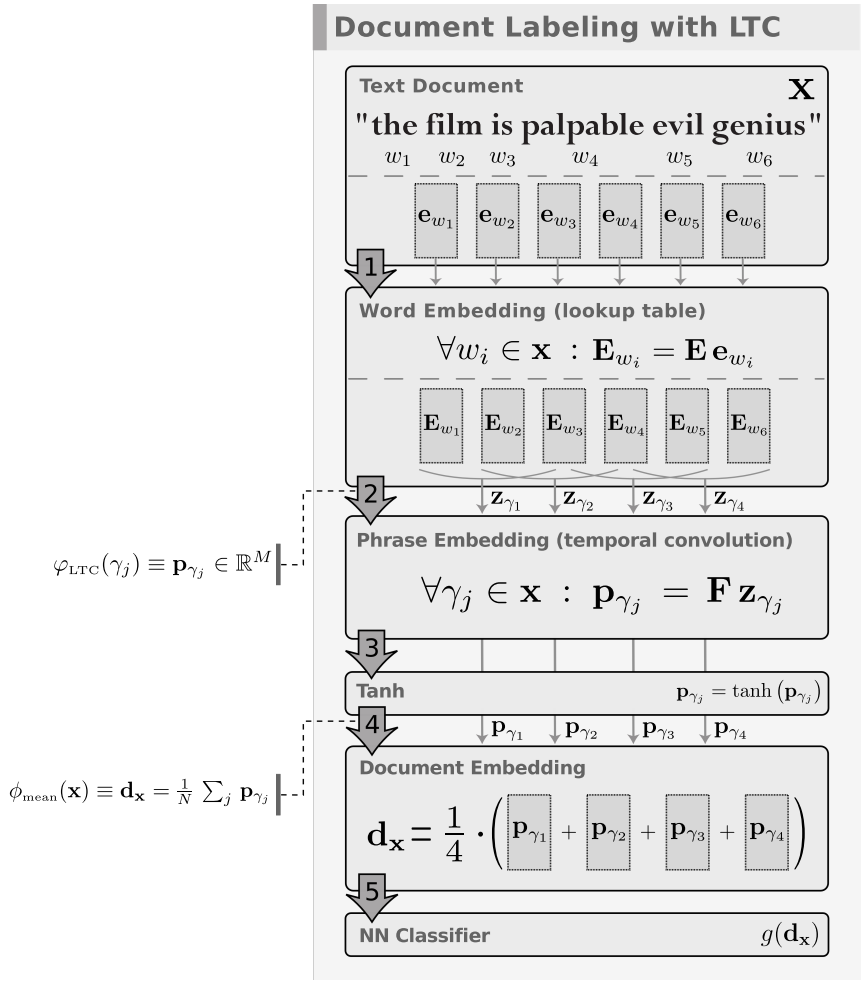


Figure 2.3: Lookup Temporal Convolution (LTC). The LTC representation is coupled with a NN classifier into document labeling system.

gio [56], the motivation for this step is to incorporate semantic information associated with every word in a low-dimensional vector space. The second stage of LTC then proceeds to build latent n-gram embedding vectors using a convolution projection with a sliding window of size n. As illustrated in Figure 2.3, when we set $n = 3$, the first n-gram is $\gamma_1 = (w_1, w_2, w_3)$, the second n-gram will be $\gamma_2 = (w_2, w_3, w_4)$, etc. Intuitively, supervised feature selection takes place at the second stage of the LTC model. At this stage an M-dimensional representation is computed for an n-gram from n individual m-dimensional embedding vectors, which are computed for unigrams that make up the phrase. The scalar parameter M denotes the dimensionality of the latent space used for n-gram embedding. The parameter m denotes the dimensionality of the "intermediate" latent space used for the embedding of unigrams. Both latent spaces are low-dimensional, which implies that

$m \ll |\mathcal{D}|$ and $M \ll |\mathcal{D}|$. Specifically, in the first stage of the LTC model, a latent embedding of a unigram w_i, referred to as a *lookup table* $\mathbf{E}_{w_i}$, is defined as

$$\mathbf{E}_{w_i} \quad = \quad \mathbf{E}\,\mathbf{e}_{w_i}, \tag{2.3}$$

where $\mathbf{e}_{w_i}$ is defined in (2.1), and $\mathbf{E} \in \mathbb{R}^{m \times |\mathcal{D}|}$ is a matrix of unigram embedding parameters. Here, $\mathbf{E}_{w_i} \in \mathbb{R}^m$ denotes the w_i-th column of matrix $\mathbf{E}$, which is an embedding vector for the word $w_i \in \mathcal{D}$.

The embedding of all n-grams $\gamma_j \in \mathbf{x}$ can then be defined as a convolutional projection of the form $\varphi(\mathbf{x}) = \mathbf{F} * [\mathbf{E}_{w_1}, \ldots, \mathbf{E}_{w_N}]$, where $*$ denotes a discrete 1D convolution operator with window size n applied to the columns of the matrix $[\mathbf{E}_{w_1}, \ldots, \mathbf{E}_{w_N}] \in \mathbb{R}^{m \times N}$, and $\varphi(\mathbf{x}) \in \mathbb{R}^{M \times N}$ is a matrix, whose j-th column contains latent embedding vector for n-gram γ_j. Specifically, given a phrase γ_j, we define

$$\mathbf{z}_{\gamma_j} = \text{vec}\big(\mathbf{E}_{w_j}, \mathbf{E}_{w_{j+1}}, \ldots, \mathbf{E}_{w_{j+n-1}}\big),$$

where $\mathbf{z}_{\gamma_j} \in \mathbb{R}^{n \cdot m}$ is a vector comprised of the embedding vectors for the individual unigrams in γ_j. We then define the embedding of the n-gram as

$$\varphi(\gamma_j) \equiv \mathbf{p}_{\gamma_j} = \tanh\left(\mathbf{F}\mathbf{z}_{\gamma_j}\right), \tag{2.4}$$

where the projection matrix $\mathbf{F} \in \mathbb{R}^{M \times nm}$ maps the vector $\mathbf{z}_{\gamma_j}$ into an M-dimensional latent space, and $\tanh(\cdot)$ denotes the non-linear *hyperbolic tangent* function, which is applied element-wise to convert the unbounded range of the input vector into $[-1, 1]$.

In other words, LTC constructs a low-dimensional latent embedding for all phrases $\gamma_j \in \mathbf{x}$ by first projecting each word into a latent space, followed by a second projection step to obtain the latent embedding of each n-gram. Collobert and Weston [73] applied a similar strategy to six word-level classification tasks for Natural Language Processing. The representation based on SSE model, discussed in Chapter 3, computes the latent embedding of n-grams using a single projection operator. The experimental evaluations presented in the following chapters indicate that SSE computes a more effective representation for text, which in turn

significantly improves the labeling performance over the LTC model. In addition, the time required to train the system with SSE is significantly lower when compared to LTC. Training an LTC model involves many vector multiplications to calculate the gradients $\partial\mathcal{L}/\partial\mathbf{E}$ and $\partial\mathcal{L}/\partial\mathbf{F}$ due to the multiplicative coupling of $\mathbf{E}$ and $\mathbf{F}$. In contrast, these computations are largely avoided when training SSE.

2.3 Task Models for Document Labeling

One of the earliest task models applied to document labeling was the Naïve Bayes (NB) classifier [74]. The NB classifier is a statistical model, motivated by the conditional independence assumption, which was originally proposed in the context of document retrieval over four decades ago [99]. As was mentioned in Section 2.1, the formulation of the NB classifier gives rise to the the bag-of-words model for text. Specifically, let U_i denote a random variable that corresponds to a unigram w_i in text $\mathbf{x} = (w_1, \ldots, w_N)$, and J denote a random variable for document label $y_k \in \mathcal{Y}$. The NB classifier approximates the probability distribution $\Pr(J \mid U_1, \ldots, U_N)$, and the decision rule is formulated using the maximum a posteriori:

$$\begin{aligned} y^* &= \arg\max_{y \in \mathcal{Y}} \Pr(J = y_k \mid U_1 = w_1, \ldots, U_N = w_N) \\ &= \arg\max_{y \in \mathcal{Y}} \Pr(J = y_k) \Pr(U_1 = w_1, \ldots, U_N = w_N \mid J = y_k), \end{aligned}$$

where $\Pr(J)$ (i.e., class prior) and conditional probability distribution $\Pr(U_1, \ldots, U_N \mid J)$ (i.e., likelihood) can be approximated with relative frequencies from the supervised samples. In other words, unigram frequencies and ground-truth labels in supervised samples are used to estimate the discrete distribution $\Pr(U_1, \ldots, U_N \mid J)$, which is stored as a multi-dimensional array in $\mathbb{R}^{C \times |\mathcal{D}|^N}$. Consequently, the following two assumptions are introduced to avoid exponential explosion of the parameter space and keep the problem tractable.

The first assumption is *invariance to feature location*, which is implied by the term "bag" in the name of the representation. The BoW model ignores the positions of the unigrams in the input texts. Specifically, we have $\Pr(U_j = w_i \mid J) = \Pr(U_k = w_i \mid J)$ for any word $w_i \in \mathcal{D}$, and two random variables U_j and U_i . The second assumption is the *conditional independence* of features given labels. This implies that any random

variable U_i is conditionally independent from all other random variables U_j, and the following holds:

$$\forall i \neq j : \Pr(U_i \mid J, U_j) = \Pr(U_i \mid J).$$

As a result, the decision rule in the NB classifier can be restated as follows:

$$y^* = \arg\max_{y \in \mathcal{Y}} \prod_{i=1}^{N} \Pr(J) \Pr(U = w_i \mid J).$$

This procedure, which estimates the parameters of the NB model, is commonly referred to as the maximum likelihood estimate. It is not hard to see that the latter formulation of the NB decision rule requires $\mathbb{R}^{|\mathcal{Y}| \times |\mathcal{D}|}$ and $\mathbb{R}^{|Y|}$ parameters to estimate likelihood and prior, i.e., distributions $\Pr(U = w_i \mid J)$ and $\Pr(J)$, respectively. It is also a common practice to smooth the term frequencies with IDF (or other weighting schema) in the maximum likelihood estimate [74]. Similarly to the vector space definition of BoW, the IDF component of the score diminishes the weight of unigrams that occur in many documents in the corpus. The motivation for IDF is that occurrences of less frequent words in texts provide better evidence for the categorization.

The NB model has been successfully applied to text categorization problems [100–102]. However, due to the aforementioned assumptions made to reduce the size of tunable parameters in NB, BoW can not properly capture dependencies among consecutive unigrams that appear in text [74]. For example, the proximity of the unigrams in the phrase "not very good" is ignored in BoW representation. As such, BoW is not able to accurately capture the true sentiment expressed in the text. Hidden Markov Models (HMM) can be used to capture the dependence among the consecutive features in a statistical framework [103]. In contrast to the NB classifier, the parameters of HMMs require iterative optimization that approximates the maximum likelihood estimate locally. The expectation-maximization algorithm is a popular generative procedure used for training HMMs [104]. Discriminative training can also be used to compute maximum likelihood for HMM, e.g., using the gradient descent method [105].

There exists a large body of empirical evidence that *vector space* models outperform statistical task models for document labeling [71, 101, 102]. In Section 2.1 the BoW vector space representation for text was discussed. Support vector machines (SVM) are among the most popular vector space task model for document

labeling. SVMs are known for their state of the art performance on tasks such as sentiment classification [1] and text categorization [71]. The SVM classifier is formulated as a convex quadratic programming optimization, which is usually solved by an interior point method [106]. For the purpose of this discussion, we consider a rudimentary formulation of the SVM optimization. Let $\mathbf{d}_{\mathbf{x}} \in \mathbb{R}^{|\mathcal{D}|}$ denote a BoW representation for text $\mathbf{x}$, and let $\mathcal{Y} = \{-1, 1\}$. In other words, we consider a binary classification problem with a negative -1 and a positive 1 labels. Intuitively, SVM computes a vector $\mathbf{v}$ and a scalar b such that for all supervised samples $\{(\mathbf{d}_{\mathbf{x}_i}, y_i) \mid i = 1, \ldots, L\}$ we have $y_i \cdot (\mathbf{v}^\top \mathbf{d}_{\mathbf{x}_i} - b) \geq 1$. The geometric interpretation is that a set of points $\mathbf{d} \in \mathbb{R}^{|\mathcal{D}|}$ that satisfy $\mathbf{v}^\top \mathbf{d} + b = 0$ is a *maximum margin hyperplane* that separates positive and negative supervised samples in $\mathcal{X}$. Indeed, for the samples with a positive label we have $\mathbf{v}^\top \mathbf{d}_{\mathbf{x}_i} - b \geq 1$, while for all negative samples the following holds: $\mathbf{v}^\top \mathbf{d}_{\mathbf{x}_i} - b \leq -1$. It is not hard to see that maximizing the distance between two hyperplanes $\mathbf{v}^\top \mathbf{d}_{\mathbf{x}} - b = -1$ and $\mathbf{v}^\top \mathbf{d}_{\mathbf{x}} - b = 1$ will ensure "good" separation of the positive and negative samples. Moreover, the distance between these hyperplanes is $\frac{2}{\|\mathbf{v}\|}$, where $\|\mathbf{v}\|^2 = \mathbf{v}^\top \mathbf{v}$. As such, the SVM optimization is formulated to maximize the margin between two hyperplanes [107]:

$$\begin{aligned} \min_{\mathbf{v}, b} \quad & \frac{1}{2}\|\mathbf{v}\|^2 \\ \text{s.t.} \quad & y_i \cdot (\mathbf{v}^\top \mathbf{x}_i - b) \geq 1, \\ & \forall i \in \{1, \ldots, L\}. \end{aligned} \tag{2.5}$$

It can be shown that by writing the optimization (2.5) in the Lagrangian dual form and setting

$$\mathbf{v} = \sum_{i=1}^{L} \lambda_i \cdot y_i \cdot \mathbf{d}_{\mathbf{x}_i},$$

the SVM may be reduced to [107]:

$$\begin{aligned} \max_{\lambda_i \geq 0} \quad & \frac{1}{2}\|\mathbf{v}\|^2 - \sum_{i=1}^{L}\sum_{j=1}^{L} \lambda_i \cdot \lambda_j \cdot y_i \cdot y_j \cdot \kappa\left(\mathbf{d}_{\mathbf{x}_i}, \mathbf{d}_{\mathbf{x}_j}\right), \\ \text{s.t.} \quad & \sum_{i=1}^{L} \lambda_i \cdot y_i = 0, \\ & \lambda_i \geq 1, \ \forall i \in 1, \ldots, L. \end{aligned} \tag{2.6}$$

Here λ_i are Lagrangian multipliers (i.e., dual variables), and $\kappa\left(\mathbf{d}_{\mathbf{x}_i}, \mathbf{d}_{\mathbf{x}_j}\right) = \mathbf{d}_{\mathbf{x}_i}^{\top}\, \mathbf{d}_{\mathbf{x}_i}$. In the dual form, the above optimization requires only inner product values $\kappa\left(\mathbf{d}_{\mathbf{x}_i}, \mathbf{d}_{\mathbf{x}_j}\right)$ between the supervised samples $\mathbf{x}_i$ and $\mathbf{x}_j$. This property of SVM, also known as the *kernel trick*, gives rise to families of non-linear SVM classifiers. Intuitively, the use of non-linear similarity or kernel function $\kappa(\cdot, \cdot)$ in (2.6) is equivalent to a non-linear transformation of the input vector space $\mathbb{R}^{|\mathcal{D}|}$ into a new high-dimensional space where the similarity between the samples is defined. A popular non-linear kernel is the *radial basis function* (RBF), defined as $\kappa(\mathbf{d}_{\mathbf{x}_i}, \mathbf{d}_{\mathbf{x}_j}) = \exp(-\alpha\|\mathbf{d}_{\mathbf{x}_i} - \mathbf{d}_{\mathbf{x}_j}\|^2)$ for some hyper-parameter $\alpha > 0$. Other popular choices of kernel functions include polynomial or hyperbolic tangent [107].

The optimization (2.5) does not account for mislabeled samples. Indeed, it is reasonable to assume that no hyperplane exists that perfectly separates positive from negative training examples. In this case, it is desirable to compute the so-called *soft maximum-margin hyperplane* that separates the supervised samples as best as possible. The idea, first proposed by Cortes and Vapnik [72], is to use an additional *slack variable* ϵ_i to measure a degree of misclassification for the supervised sample $\mathbf{x}_i$. As a result, each constraint in (2.5) can be written as $y_i \cdot (\mathbf{v}^{\top}\, \mathbf{x}_i - b) \geq 1 - \epsilon_i$, and the new optimization objective $\frac{1}{2}\|\mathbf{v}\|^2 + C \cdot \sum_{i=1}^{N} \epsilon_i$ measures the trade-off between large margin and misclassification penalty. The hyper-parameter C, also known as the SVM penalty parameter, controls this trade-off.

The above formulation of SVM computes a binary classifier. When $|\mathcal{Y}| \geq 3$, the *one-vs-all* approach for multi-class SVM is to reduce the problem into $|\mathcal{Y}|$ binary classifications [108]. In this formulation, each binary classifier corresponds to one of the class labels $y_j \in \mathcal{Y}$ and predicts whether or not label y_j is assigned to an input sample. The binary classifier with the highest output determines the final prediction label for the sample. Alternatively, multi-class SVM can be formulated as a single optimization without decomposing

it into $|\mathcal{Y}|$ binary classifications [109]. It is also worth noting that using non-linear kernels for document labeling with SVM results in marginal, if any, improvement over the linear kernel (i.e., $\kappa\left(\mathbf{d}_{\mathbf{x}_i}, \mathbf{d}_{\mathbf{x}_j}\right) = \mathbf{d}_{\mathbf{x}_i}^{\top}\, \mathbf{d}_{\mathbf{x}_i}$) [1, 101]. This can be attributed to the high-dimensional nature of the BoW representation, which allows for good separation of classes in the original BoW vector space [110] In addition, solving the SVM optimization with a non-linear kernel is known for its slow convergence. On the other hand, a number of efficient training procedures have been recently introduced for linear SVM in both primal [111] and dual [112] forms. This in turn allows one to train linear SVM to handle large-scale classification tasks. Here, the term "large" refers to the number of the supervised samples in $\mathcal{X}$ and the number of features used in the representation domain.

The BoW model preserves lexical and ignores compositional semantics of the natural language. The latter is due to feature location invariance in the model. In practice, when combined with an appropriate task model (e.g., SVM), the BoW representation can be very effective at document labeling [71]. This suggests that estimating a frequency distribution of the unigrams in the corpus is sufficient for categorization. Indeed, the non-adaptive top-down BoW model gives rise to surprisingly effective *document similarity* methods for document labeling. As we mentioned in Section 2.1, the document similarity representation assigns numerical similarity (or distance) values to pairs of documents. Let $\mathbf{d}_{\mathbf{x}_i} \in \mathbb{R}^{|\mathcal{D}|}$ and $\mathbf{d}_{\mathbf{x}_j} \in \mathbb{R}^{|\mathcal{D}|}$ denote the BoW representation for two documents. A popular similarity measure in the BoW representation domain is *cosine similarity* [113], defined as the cosine of the angle between the two BoW vectors:

$$\frac{\mathbf{d}_{\mathbf{x}_i}^{\top}\, \mathbf{d}_{\mathbf{x}_j}}{\|\mathbf{d}_{\mathbf{x}_i}\| \cdot \|\mathbf{d}_{\mathbf{x}_j}\|}. \tag{2.7}$$

Having defined a document similarity measure, the $k-$nearest neighbor (k-NN) classifier can now be stated as an example of a model with instance-based or "lazy" learning. Classification methods with *instance-based learning* make label predictions for new samples by examining training instances. The training of the k-NN classifier consists of constructing the BoW vectors for all supervised samples. Given a query, label prediction for the sample proceeds as follows. First, training documents which are the most similar to the query are identified. Then the similarity scores between the supervised samples and the query are used to accumulate weights for the corresponding classes. The category label that receives the highest weight is returned as the

prediction for the given query sample. This simple k-NN classifier with the cosine similarity measure, defined on the BoW unigrams model for text, results in rather effective document labeling systems. Indeed, empirical evidence suggests that the performance of k-NN classifiers with the BoW representation can rival the performance of the NB classifier [113]. However, the performance of SVM methods is generally significantly better than the k-NN (and NB) classifiers [71, 101, 102]. It is important to note that labeling systems based on document similarity are not limited to using k-NN classifier. For example, the aforementioned kernel trick for SVM allows one to use a similarity measure defined by a variety of string kernels [114], which were discussed in Section 2.1.

Three classes of labeling systems have been considered in this section so far, namely statistical, vector space, and document similarity methods. Most modern document labeling systems involve an adaptive classifier. Indeed, even k-NN is implicitly adaptive since the supervised samples are used when label prediction is made, although training of the k-NN classifier is not formulated as a numerical optimization. On the other hand, text classification methods predominantly rely on the non-adaptive BoW or BoN representation. One notable exception includes labeling systems based on HMMs. In HMM methods the representation and task models are implemented as a single component that allows it to capture short-term dependencies (e.g., a sequence of five consecutive unigrams) within a unified statistical framework [115]. The maximum likelihood estimate for the parameters in HMMs are often obtained using an iterative method called expectation-maximization (EM) [116]. However, the EM algorithm can exhibit slow convergence and is prone to finding local optimal since the objective is often a non-convex function. In practice, HMM-based document labeling systems are not as effective as methods that rely on SVM or even NB classifiers [71, 117]. When dealing with text labeling problems, SVM combined with the BoW representation result in the state of the art performance when a sufficient number of supervised samples is available [1, 71, 101].

Deep architectures have received renewed attention in recent years for their ability to combine feature extraction and a task model into a single unified framework. All tunable parameters in this framework are jointly biased to perform the required task. This property enables adaptive representations to learn complicated functions in natural language processing (e.g., LTC [73]) and computational vision (e.g., [53]). In contrast, representation and task models in most other document labeling methods are implemented as two

independent components, where only the task model is adaptive (i.e., contains tunable parameters). This thesis develops a novel SSE model for text and image data that improves upon LTC and advances the state of the art document labeling by computing more meaningful representation for sequences of quantized features (e.g., n-grams or image patches) in a latent space. In addition, the SSE-based text classification system perform statistically significantly better than the comparable methods that rely on the non-adaptive BoW or BoN models.

The SSE model for text $\mathbf{x}$ computes a low-dimensional embedding of all n-grams in $\mathbf{x}$. Let $\varphi_{\text{SSE}}(\gamma_j) \in \mathbb{R}^M$ denote an embedding vector for phrase $\gamma_j \in \mathbf{x}$. The latent n-grams are then combined to form an M-dimensional embedding vector for the entire text, denoted with $\mathbf{d}_{\mathbf{x}}$. The SSE model is coupled with a perceptron classifier to compute a document labeling task. Perceptron classifiers are implemented as an Energy-Based Model (EBM) [118] that, for a given input $\mathbf{d}_{\mathbf{x}}$, associates a scalar energy value with every configuration of the output variables. In the case of an EBM classifier, each output variable corresponds to one of the category labels $y_i \in \mathcal{Y}$. Let $\boldsymbol{\beta}_{y_i} \in \mathbb{R}^M$ denote an embedding vector that corresponds to the class label y_k, i.e., vector $\boldsymbol{\beta}_k$ contains the coefficient weights for the k-th candidate class. Then $\xi(\mathbf{d}_{\mathbf{x}}, \boldsymbol{\beta}_k)$ denotes the likelihood that $\mathbf{d}_{\mathbf{x}}$ belongs to class y_k, which is inversely proportional to the energy values assigned to each pair $(\mathbf{d}_{\mathbf{x}}, \boldsymbol{\beta}_k)$. As such, a perceptron classifier predicts a label for the input $\mathbf{d}_{\mathbf{x}}$ using a decision functional:

$$g(\mathbf{d}_{\mathbf{x}}) = \underset{k \in \{1..|Y|\}}{\arg\max}\ \xi(\mathbf{d}_{\mathbf{x}}, \boldsymbol{\beta}_k). \tag{2.8}$$

Training of this classifier consists of tuning its parameters so that the *loss functional* $\mathcal{L}(\cdot)$ is minimized. The loss function measures the discrepancy between the estimated and preferred distribution of energies using supervised samples. This allows one to use a variety of loss functionals in the formulation of the perceptron classifier. We consider the two most popular perceptron classifiers: *hinge loss* and *negative log-likelihood loss*. It is worth noting that the hinge loss function is also used in the SVM formulation in conjunction with a quadratic regularizer [118]. The likelihood measure $\xi_{\text{HNG}}(\mathbf{d}_{\mathbf{x}}, \boldsymbol{\beta}_k)$ for the hinge loss is defined as

$$\xi_{\text{HNG}}(\mathbf{d}_{\mathbf{x}}, \boldsymbol{\beta}_k) = \mathbf{d}_{\mathbf{x}}^{\top}\, \boldsymbol{\beta}_k. \tag{2.9}$$

Let $(\mathbf{x}_k, y_k)$ denote a supervised sample where y_k is the ground-truth label for $\mathbf{x}_k$. In addition, let $\boldsymbol{\beta}_{y_k}$ denote an embedding vector for label y_k and $\mathbf{d}_{x_k}$ denote latent embedding vector for document $\mathbf{x}$. Define $\mathbf{y}_k^-$ as the most offending label which is not assigned to $\mathbf{x}_k$:

$$y_k^- = \underset{y_j \neq y_k}{\arg\max}\ \xi(\mathbf{d}_{\mathbf{x}_k}, \boldsymbol{\beta}_{y_j}).$$

The principal idea behind the hinge loss is to ensure a large margin separation between energies $\xi(\mathbf{d}_{\mathbf{x}_k}, \boldsymbol{\beta}_{\mathbf{y}_k})$ and $\xi(\mathbf{d}_{\mathbf{x}_k}, \boldsymbol{\beta}_{\mathbf{y}_k^-})$. In particular, the hinge loss for a single sample $\mathbf{d}_{\mathbf{x}_k}$ is then defined as

$$\mathcal{L}_{\text{HNG}}(\mathbf{d}_{\mathbf{x}_k}, \boldsymbol{\beta}_{y_k}, \boldsymbol{\beta}_{y_k^-}) = \max\left(0, \mu - \xi(\mathbf{d}_{\mathbf{x}_k}, \boldsymbol{\beta}_{y_k}) + \xi(\mathbf{d}_{\mathbf{x}_k}, \boldsymbol{\beta}_{y_k^-})\right), \tag{2.10}$$

where μ is the margin hyper-parameter. In the multi-class setting, the response values $\xi(\cdot, \cdot)$ for different classes are normalized. And the *hinge loss*, defined on the entire training set $\mathcal{X}$, will take the form:

$$\mathcal{L}_{\text{HNG}}(\mathcal{X}) = \sum_{k \in \{1 \ldots |\mathcal{X}|\}} \max\left(0, \mu - -\xi(\mathbf{d}_{\mathbf{x}_k}, \boldsymbol{\beta}_{y_k}) + \xi(\mathbf{d}_{\mathbf{x}_k}, \boldsymbol{\beta}_{y_k^-})\right). \tag{2.11}$$

It is important to note that the formulation of the perceptron classifier with hinge loss (2.11) requires margin hyper-parameter μ. Another popular perceptron classifier is based on the *negative log-likelihood loss* (NLL) [118] which requires no hyper-parameters to be set. The formulation of the NLL classifier is motivated by the maximum conditional probability principle: minimizing the NLL loss is equivalent to the maximum conditional likelihood estimation. In other words, training the NLL classifier sets the NN parameters that maximize the conditional probability of the ground-truth category y_k given the embedding vector $\mathbf{d}_{\mathbf{x}_k}$ [118]. Specifically, define the likelihood measure for the NLL classifier as

$$\xi_{\text{NLL}}(\mathbf{d}_{\mathbf{x}_k}, \boldsymbol{\beta}_j) = \frac{\exp(\boldsymbol{\beta}_j^\top \mathbf{d}_{\mathbf{x}_k})}{1 + \sum_{i \in \{1..|Y|\}} \exp(\boldsymbol{\beta}_i^\top \mathbf{d}_{\mathbf{x}_k})}. \tag{2.12}$$

The NLL loss function can then be defined as

$$\mathcal{L}_{\text{NLL}}(\mathcal{X}) = -\sum_{k \in \{1..|\mathcal{X}|\}} \log \xi_{\text{NLL}}(\mathbf{d}_{\mathbf{x}_k}, \boldsymbol{\beta}_{y_k}). \tag{2.13}$$

It is worth noting that NLL is sometimes referred to as the *cross-entropy loss.* Indeed, it is not hard to see that NLL classifier, defined by loss (2.13), minimizes the cross-entropy between sought and observed conditional probability distributions over all training samples [119].

2.4 Evaluation Methodologies

The ultimate goal of a machine learning system is to tune its parameters to match the actual output with the preferred (ground-truth) output as defined by supervised samples. The premise is that after the parameter tuning, the system will effectively compute the task on new samples that were not seen during training. We evaluate the effectiveness of each method using standardized benchmarks. Specifically, suppose we are given a dataset that contains supervised sample pairs $(\mathbf{x}_i, y_i)$, i.e., document $\mathbf{x}_i$ and its ground-truth sample y_i. We split the dataset into three subsets: *training*, *development* and *testing*. The ratio between the number of samples in the training and testing subsets is $70\%/30\%$. The development subset is obtained by removing $\sim 5\%$ of the original samples from either the training or testing subsets.

The training set is used to bias the parameters of a document labeling system, while the development set is used to identify the best-performing hyper-parameters in each method. Once the training procedure terminates, the effectiveness of the system is evaluated using the testing set. Our training procedure for the perceptrons is depicted in Algorithm 1. This setup allows us to qualitatively compare the effectiveness of multiple labeling systems and identify more effective ones for the given task. We use two metrics to quantify the effectiveness of labeling systems, namely, micro- and macro-average classification error. The *macro-average classification error* is defined as the mean of per-category classification errors, which are computed for every class label independently and then averaged to produce the macro-average error. The macro-average error is a popular classification metric when distribution of labels assigned to supervised samples is not uniform. In other words, macro-average error is useful when the number of supervised samples in each category varies significantly. The *micro-average classification error* is an average error computed

Algorithm 1 The online training procedure for neural networks using stochastic gradient descent. This procedure was used to train all perceptrons in our empirical evaluations.

Input: $\mathcal{X}_{\text{trn}}$, $\mathcal{X}_{\text{tst}}$, and $\mathcal{X}_{\text{dev}}$ are the training, testing and development sets, respectively; λ is the learning rate

Output: Optimized NN parameters $\boldsymbol{\Theta}^{\text{opt}} = \{\boldsymbol{\theta}_1^{\text{opt}}, \ldots, \boldsymbol{\theta}_{T+1}^{\text{opt}}\}$ from all layers f_k, where $k \in [1, T+1]$ and $f_{T+1} \equiv \mathcal{L}$ is the loss functional

```
EpochSize ← 5 · |X_dev|                          # setting epoch size
Θ ← ∅                                            # Θ denotes current set of parameters
for k = 1 → T + 1 do
    θ_i ← N(0, 1)                                # randomly init. parameters at every layer
    Θ ← Θ ∪ {θ_i}                                # Θ contains parameters from all layers
end for
Θ^opt ← Θ                                        # Θ^opt denotes best-performing parameters

repeat
    for t = 1 → EpochSize do                     # train for one epoch
        (x_{i_t}, y_{i_t}) ← rand(X_trn)         # select random sample from the training set X_trn
        P ← 1                                    # P maintains ∂f_{T+1} / ∂f_k
        for k = T + 1 → 1 do
            ∂L/∂θ_k ← P ∂f_k/∂θ_k                # compute derivative of L with respect to θ_k
            θ_k ← θ_k − λ · ∂L/∂θ_k               # update parameters of f_k to minimize the loss
            P ← P ∂f_{k+1}/∂f_k
        end for
    end for                                      # end of one epoch
    if evalErr(Θ^opt, X_dev) > evalErr(Θ, X_dev) then   # Θ improves classification on X_dev
        Θ^opt ← Θ                                # update best-performing parameters Θ^opt
    end if
until evalErr(Θ^opt, X_dev) does not improve for 3 epochs

tstErr ← evalErr(Θ^opt, X_tst)                   # compute classification error for the testing set X_tst
return Θ^opt, tstErr
```

regardless of the labels assigned to the supervised samples. In other words, micro-average error is a measure for classification accuracy, which is invariant to the distribution of the category labels in the testing set.

In addition, we use a z-test [120] to compare the performance of the labeling methods in order to identify the ones that perform statistically significantly better than other systems. Specifically, we compare the difference in micro-average classification errors in terms of its statistical significance. Assume we have two systems S_1 and S_2, each tested on N samples, where each sample is considered as an independent trial. Let p_1 and p_2 denote random variables corresponding to micro-average classification errors for S_1 and S_2 using N samples, respectively. Thus, the number of samples that S_1 and S_2 predict incorrectly are $p_1 \cdot N$ and $p_2 \cdot N$, respectively. Without loss of generality, assume that $p_1 > p_2$. We are interested in estimating the probability

that S_2 performs significantly better than S_1 using N samples. The z-test proceeds to estimate the probability that the null hypothesis H_0 holds, where H_0 denotes the event when S_2 performs the same or worse than S_1 (i.e., when $p_1 \geq p_2$). Assuming that the distribution of the difference of the two random variables $p_1 - p_2$ is approximately normal, for large N we can estimate the normalized z-score for the observed values of p_1 and p_2. In other words, we use a standard normal table to approximate the probability $\Pr(p_1 - p_2 \geq 0)$, which indicates the likelihood that system S_1 is more effective that S_2. Intuitively, the smaller the probability $\Pr(p_1 - p_2 \geq 0)$, the more likely the null hypothesis H_0 is rejected, thus S_2 is more likely to outperform S_1.

A large body of our work is related to sentiment analysis. Specifically, supervised sentiment classification [58], which aims at predicting whether the expressed opinion in the text is positive, negative, or neutral. Over the past decade, the widespread use of electronic media has produced large volumes of user-generated content in the form of blogs, reviews, tweets, news and other articles, often combining text and image data. Identifying subjective statements and rating opinions in such content is of great importance for many practical applications such as marketing (e.g. ad placement, brand trending, etc.), consumer protection (e.g., online search and retrieval of recommendations or reviews), and even disaster relief operations [121]. SSE was developed as an alternative to the BoN model that encodes n-grams in the representation domain. SSE does not require feature selection to control the dimensionality of the representation domain, since the parameter space of SSE grow linearly with the size of the n-gram this model can encode.

The motivation for SSE was the observation that sentiment analysis can greatly benefit from encoding phrases, rather than unigrams, in the representation domain, and a classifier can take advantage of phrase semantics when learning to predict sentiment. As such, the principal goal of this dissertation is three-fold. In Chapter 3 the hypothesis that SSE is a better alternative than BoN will be investigated. Then, in Chapter 4, a study of the adaptive property of SSE will be presented, where we will consider our hypothesis that SSE retains phrase semantics in the low-dimensional representation domain. Finally, in Chapter 5 we will consider whether SSE can be applied to other modalities of sensory data. Specifically, the SSE model will be applied to encode configurations of image patches with low-dimensional latent vectors, and the 2D variant of SSE will be applied to large-scale image classification task.

Chapter 3: Supervised Sequence Embedding

The bag-of-words model for text data, discussed in Chapter 2.1, only preserves lexical semantics of unigrams in the representation domain [71]. In theoretical linguistics, *lexical semantics* study the meaning of lexical units and their relations (e.g., synonymy, antonymy, hyponymy, or hypernymy) [122], where a lexical unit, denoting a physical object or a semantic concept, is often encoded with several words. If positions of unigrams are ignored, which is the case for the BoW model, varying degree of sentiment in phrases, such as "very good", "good", "not very good" and "not good", will be lost. This shortcoming of BoW may lead to information loss in the representation domain and results in performance degradation of labeling tasks. Empirical evidence will be presented to support the hypothesis that short phrases are more effective at labeling tasks, such as sentiment analysis and document categorization (see Section 3.4). Zadrozny [123] was among the first to study semantics, encoded as a composition problem. The so-called *compositionality principle* states that "the meaning of a complex expression is fully determined by its structure and the meanings of its constituents" [124]. Unfortunately, the compositional property of semantics by Zadrozny [123] does not have direct computational implications, i.e., such compositional functions are not necessarily Turing-computable.

Motivated by the above observation, supervised sequence embedding (SSE) attempts to provide an efficient encoding of sensory data features in the representation domain, resembling weak compositional semantics. Clearly, the generalized form of the BoN model for all possible values of n can also be motivated as the aforementioned compositional semantics. Indeed, augmenting the BoW model with n-grams is analogous to using an exterior (i.e., wedge) product to combine individual unigrams into phrases for all values of n. However, as discussed in Chapter 2.1, this leads to an exponential explosion of features (i.e., $O(|\mathcal{D}|^n)$) in the BoN domain. In contrast, we propose to encode an n-gram as an aggregate of embedding vectors for individual unigrams in the phrase. As will be illustrated, the parameter space of the SSE representation grows linearly with the size of the n-gram. In what follows, a systematic overview of the supervised sequence embedding will be presented. The empirical validation will illustrate that the adaptive model based on SSE outperforms non-adaptive BoN for *sentiment analysis* and *text categorization*.

The structure of this chapter is as follows. In Section 3.1 the SSE model for latent n-grams will be formally introduced. Section 3.2 will provide an overview of relevant prior-art methods for tackling the text labeling tasks considered in this chapter. The baseline methods and the benchmark datasets used in our experimental studies will be discussed in Section 3.3. The discussion of experimental results, which include sentiment analysis and a limited investigation for text categorization, will be presented in Section 3.4.1 and Section 3.4.2, respectively. Section 3.5 will conclude this chapter with a discussion of the obtained experimental results.

3.1 SSE Representation for Text

In this section, supervised sequence embedding (SSE) will be presented, which is a projection operator that encodes sequences of n consecutive features in a string of text (i.e., an n-gram) using a sliding window mechanism. Each feature in the signal domain (i.e., a unigram) will participate in n individual embeddings (one per position in the sliding window of size n), resulting in $O(n \cdot |\mathcal{D}|)$ features in the representation domain. In contrast, using the BoN model that recognizes n-grams from Γ_n as features in the representation domain results in a $|\Gamma_n|$-dimensional BoN vector for text, where $|\Gamma_n| = O(|\mathcal{D}|^n)$. Due to its prohibitive dimensionality, the BoN model often relies on feature selection pre-processing to control the number of features in the representation domain. Conversely, SSE constructs n-grams in the latent space, and does not require any pre-processing to reduce the size of the feature space. The adaptive SSE representation will be coupled with a perceptron classifier, discussed in Section 2.3, to form an end-to-end multi-layer neural network for document labeling. Training the proposed labeling system with backpropagation facilitates feature selection in the SSE representation. In Chapter 4, convincing empirical and anecdotal evidence will be presented that supports this hypothesis.

The formulation for the SSE representation was motivated by the LTC method [73], discussed in Section 2.2. The SSE and LTC models encode text phrases in a low-dimensional latent space, where the embedding vectors are then combined to form a latent embedding of the whole document, so the document labeling can be carried out. Following the notations introduced in Chapter 2, the SSE-based labeling system will be described in terms of the three mappings $\varphi(\cdot)$, $\phi(\cdot)$ and $g(\cdot)$, applied to an input text $\mathbf{x} = (w_1, \ldots, w_N)$; see Figure 3.1 for an illustration. Following the notations from the previous chapter, the map $\phi(\mathbf{x}) \equiv \mathbf{d}_{\mathbf{x}} \in \mathbb{R}^M$

will denote document-level representation for a text x. The map $g(\mathbf{d}_x) \in \mathcal{Y}$ will denote a classifier that assigns labels to the representation of $\mathbf{x}$. In addition, the mapping $\varphi_{\text{SSE}}(\cdot)$ will denote phrase-level embedding that projects an n-gram $\gamma_j \in \mathbf{x}$ onto an M-dimensional latent space, so $\varphi_{\text{SSE}}(\gamma_j) \in \mathbb{R}^M$. The notation $\varphi_{\text{SSE}}(\mathbf{x})$ will denote a projection of all n-grams from $\mathbf{x}$ into an M-dimensional space, thus $\varphi_{\text{SSE}}(\mathbf{x}) \in \mathbb{R}^{M \times N}$.

The n-gram embedding $\varphi_{\text{LTC}}(\cdot)$ in the LTC model involves two projection steps, depicted in Figure 2.3. On the other hand, the mapping $\varphi_{\text{SSE}}(\cdot)$ in the SSE model, as shown in Figure 3.1, is implemented with a single projection. Similarly to SSE, the parameter space of the LTC model grows linearly with the size of the n-gram. Unlike SSE, which maintains n individual embedding vectors per unigram, the LTC model maintains a single vector for each unique word in $\mathcal{D}$. LTC computes latent phrases with a 1D convolutional projection, defined on the unigram embedding vectors. In the SSE model, a latent n-gram γ_j is defined as sum of the latent vectors for the unigrams, each selected based on the unigram position in the phrase. These observations imply that the SSE model maintains a larger set of tunable parameters than LTC. This, in turn, may allow SSE to learn more complicated representations for text, and improve effectiveness on certain computational tasks, such as large-scale document labeling. Empirical evidence to test this hypothesis will be presented in Section 3.4. In addition, the large-scale sentiment analysis and document categorization experiments will indicate that the SSE method outperforms standard baseline document labeling systems with a bag-of-n-grams representation for $n \in \{1, 2, 3, 5\}$. Finally, the results, presented in Section 3.4.1, will show that training time for the SSE model rivals the BoN-based perceptron systems, which is significantly lower than what is needed to train LTC.

3.1.1 Latent Embedding of n-Grams

In contrast to the two-stage approach employed in the LTC model, SSE forms latent n-grams with a single projection. Similar to the LTC model, the formation of n-grams is carried through a sliding window of length n. The SSE model encodes a phrase γ_j as an aggregate of the unigram embedding vectors. A total of n individual embedding vectors are assigned to each unigram feature, one for each position in γ_j. The motivation is that maintaining separate embedding vectors, based on the positions of unigrams in phrases, will allow us to capture a wider range of feature interactions in the representation domain, and improve the effectiveness of the SSE-based systems.

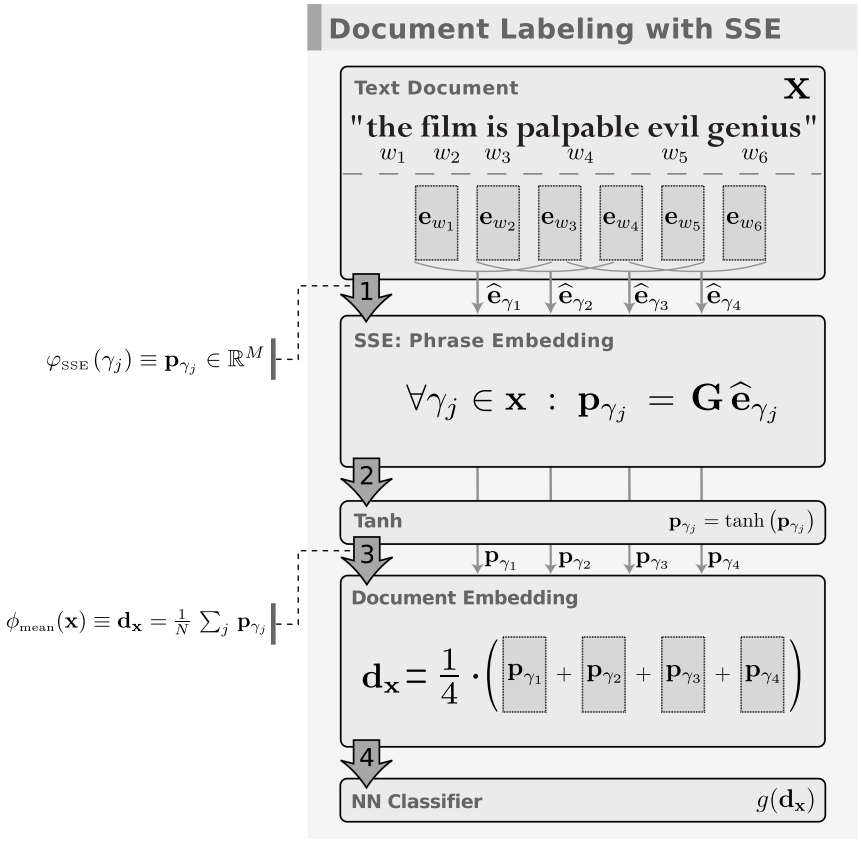


Figure 3.1: Illustration of document labeling system with the supervised sequence embedding (SSE).

In the formation of latent n-grams, SSE concatenates single word selectors to represent each phrase. Specifically, a phrase γ_j is encoded with a sparse vector $\widehat{\mathbf{e}}_{\gamma_j}$:

$$\widehat{\mathbf{e}}_{\gamma_j} \;=\; \mathrm{vec}\big(\mathbf{e}_{w_j}, \mathbf{e}_{w_{j+1}}, \ldots, \mathbf{e}_{w_{j+n-1}}\big) \tag{3.1}$$

that concatenates word selectors for unigrams in γ_j into an $n \cdot |\mathcal{D}|$-dimensional vector with n non-zero entries, each set to 1. It is worth emphasizing that unlike the BoN model, SSE does not require a weighting schema (e.g., TF-IDF) to smooth entries in (3.1). The latent embedding of an n-gram γ_j is then defined as

$$\varphi(\gamma_j) \equiv \mathbf{p}_{\gamma_j} = \tanh\left(\mathbf{G}\,\widehat{\mathbf{e}}_{\gamma_j}\right), \tag{3.2}$$

where $\tanh(\cdot)$ is the hyperbolic tangent and $\mathbf{G} \in \mathbb{R}^{M \times n \cdot |\mathcal{D}|}$ is a projection matrix that maps $\widehat{\mathbf{e}}_{\gamma_j}$ into a latent space with dimension M. Matrix $\mathbf{G}$ maintains n latent embedding vectors for every word $w_i \in \mathcal{D}$, depending on its position inside the n-gram. In other words, $\mathbf{G}$ has n "column chunks", each having $|\mathcal{D}|$ columns. The

i-th column chunk is a full lookup table for every word, as it contains one embedding for every word in $\mathcal{D}$. We note that training an LTC model using back-propagation requires many vector multiplications to calculate the gradients $\partial\mathcal{L}/\partial\mathbf{E}$ and $\partial\mathcal{L}/\partial\mathbf{F}$, due to the multiplicative coupling of $\mathbf{E}$ and $\mathbf{F}$. These computations are largely avoided in the SSE, but it contains a larger set of parameters (i.e., higher degree of freedom in the parameter space) compared to LTC. This limits the applicability of SSE to tasks having medium- or small-scale training datasets.

3.1.2 Extending Latent n-Grams to Document Embedding

Both the SSE and the LTC models compute a low-dimensional representation of all n-grams from a given text $\mathbf{x}$, which allows us to encode lexical semantics of short phrases in the representation domain. In this dissertation, only document-level labeling tasks will be considered, since the large-scale datasets for these tasks are freely available online. The latent n-grams extracted from $\mathbf{x}$ are combined to form a vector representation for $\mathbf{x}$. In what follows, a function $\phi(\cdot)$ is defined to compress the information from N latent phrases $\varphi_{\text{SSE}}(\gamma_j) \equiv \mathbf{p}_{\gamma_j} \in \mathbb{R}^M, \forall\gamma_j \in \mathbf{x}$ into a document embedding vector $\mathbf{d}_{\mathbf{x}}$, where $|x| = N$ and $\phi(\mathbf{x}) \equiv \mathbf{d}_{\mathbf{x}} \in \mathbb{R}^M$. The label prediction for an input text is computed using a perceptron classifier, e.g., the function $g_{\text{NLL}}(\mathbf{d}_{\mathbf{x}})$ that was defined in Section 2.3.

While there are many ways to combine latent n-grams $\mathbf{p}_{\gamma_j}$ into a document embedding vector $\mathbf{d}_{\mathbf{x}}$, we use the centroid of latent phrases from $\mathbf{x}$ as the aggregate function in this chapter:

$$\phi_{\text{mean}}(\mathbf{x}) \equiv \frac{1}{N}\sum_{j=1}^{N}\mathbf{p}_{\gamma_j}. \tag{3.3}$$

Intuitively, in a sentiment prediction task, the sentiment of a document is related to the aggregated polarity of all its n-grams; that is, accumulating sentiments expressed in individual phrases yields a good estimate of the sentiment expressed in the whole document. Other choices for $\phi(\cdot)$ include $\phi_{\text{max}}(\mathbf{x}) \equiv \max_{j=1}^{N}(\mathbf{p}_{\gamma_j})$, which selects the maximum value along each latent dimension [73]. Our investigation illustrated that the function $\phi_{\text{mean}}(\cdot)$ defined in (3.3) is a more appropriate choice when dealing with labeling tasks, such as sentiment prediction. As such, the results in Section 3.4 will only consider the function $\phi_{\text{mean}}(\cdot)$. The discussion of alternative models for $\phi(\cdot)$ is deferred until Chapter 4, where we will study the adaptive property of the SSE

representation. In particular, a variant for $\phi(\cdot)$ will be formulated as a weighted sum of latent unigrams, and the weights will be learned from the positions of n-grams within documents.

3.2 Related Methods

Sentiment analysis, also known as *opinion mining*, is often formulated as an instance of text classification, that predicts sentiment labels for an input text. A variety of approaches to text classification and categorization were discussed in Section 2.3. A thorough survey of automated text classification methods is available in the work by Sebastiani [71]. In addition to text categorization systems, there exist a number of methods specifically designed to tackle sentiment analysis. This section provides a short overview of these methods; the reader is referred to an opinion mining survey by Pang and Lee [58], and a more recent study by Liu and Zhang [125]. A detailed account of the latest developments in sentiment analysis methods can also be found in the book by Liu [126] on this subject.

Sentiment analysis methods, according to the definition coined by Pang and Lee [58], deal with "computational treatment of opinion, sentiment, and subjectivity in text". It is common to identify several prominent directions of the opinion mining research [127]. *Sentiment and subjectivity classification* is the most studied area of opinion mining, where input text is usually labeled in terms of its sentiment polarity and objectivity (i.e., objective vs subjective). Sentiment labels can be assigned to whole documents or to individual sentences. In the former formulation, the problem is often tackled as a document labeling task with two categories, corresponding to either positive or negative sentiment (or opinion) in online reviews [1, 128, 129], blogs [130], news articles [131], and other content [132]. In the sentence-level sentiment classification, statements in the input text (e.g., phrases or whole sentences) are first classified in terms of their subjectivity, and subjective sentences are then classified based on sentiment polarity (i.e., positive or negative). An *objective* statement is said to express some factual information, while a *subjective* sentence relays personal feelings or beliefs [127]. The sentence-level sentiment analysis has also been extensively studied in the literature [133–137]. Other notable methods to tackle sentiment classification attempt to capture sentiment and topical information in the representation domain, so that various aspects of opinions can be predicted [138–140].

Active research topics in opinion mining also include the so-called *feature-based* and *comparative* sentiment analysis [127]. Feature-based sentiment analysis is formulated as a structured data extraction that

identifies the following entities in free text: object of the review (product or service), opinion holder (author), time, sentiment of opinion (positive or negative), as well as optional aspects that are relevant to the product [141–145]. Extracting opinion from comparative sentences is often related to feature-based sentiment analysis. However, in the former, sentiment is not expressed directly (e.g., "product-X is amazing"), but through comparative statements about objects (e.g., "product-X is better than product-Y, but not as good as product-Z") [146–148]. *Opinion search and retrieval* is another promising area of sentiment analysis research that deals with indexing, retrieval and querying of opinionated documents [149–151]. *Opinion spam* detection is yet another research topic of sentiment analysis, which is defined as the task of catching fake or "bogus" opinions that attempt to mislead readers by giving "undeserving positive" or "malicious negative" opinions for products and services [152–154].

From another perspective, the proposed SSE model encodes compositional semantics of input text in the representation domain. This information, compressed into a low-dimensional latent vector, is then used by a classifier as evidence for labeling. As such, a number of models related to SSE can be identified. The most popular vector space approach populates BoW representation with n-gram features (i.e., BoN). It is worth noting that a weighting schema for BoN, called *Delta TF-IDF*, was specifically designed for sentiment classification by Martineau and Finin [155]. However, it has been reported that Delta TF-IDF does not result in consistent improvement over standard TF-IDF [156], especially when multiple large-scale datasets are considered [157]. As discussed in Chapters 1 and 2, BoN is a non-adaptive model for feature extraction, that requires pre-processing to retain effective features and control dimensionality of the representation domain. Several adaptive alternatives to BoN have been proposed in recent years that capture compositional semantics of text in the representation domain. For example, lookup temporal convolution (LTC), proposed by Collobert and Weston [73] and discussed in Section 2.2, encodes text phrases with a sliding window projection operator. LTC was implemented using a neural network architecture, while a generative model to encode windows of n words using restricted Boltzmann machines was recently proposed by Dahl *et al.* [158]. Another notable development among adaptive representations for capturing compositional semantics used recursive auto-encoders (RAE) to predict sentiment distributions in a semi-supervised setting [86]. This RAE system learned an embedding vector for every unigram in text, and a computed low-dimensional embedding

(i.e., interpretation vector) of a whole document recursively. At every step, the RAE procedure maintained a current interpretation vector, and combined it with an embedding vector for the next unigram in text via a projection matrix W. In a sequel to that work, the authors proposed learning individual matrices for every word that combined current interpretation and unigram embedding vectors [159]. The latter model allowed the RAE system to learn fine-grained sentiment distributions of adverb-adjective pairs. It is important to note that the RAE-based representation requires a matrix multiplication to encode every unigram in text, while SSE relies on vector summation to encode phrase-level compositional semantics in the representation domain. Consequently, both RAE systems [86, 159] can learn finer semantic compositionality than SSE. However, the SSE approach is sufficient to encode sentiment strength and orientation of n-grams, and is more efficient, and can handle larger-scale tasks, that can be problematic to tackle with RAE systems.

3.3 Experimental Setup

Our main hypothesis is that the adaptive SSE model, which computes a supervised embedding of text phrases, is a viable alternative to the standard non-adaptive BoN model. In particular, the SSE model is more effective than BoN when dealing with large-scale supervised labeling problems, where short phrases, rather than individual unigrams, provide evidence for classification. Binary and multi-class sentiment classification is one example of such tasks. Section 3.4.1 will demonstrate that encoding longer n-grams can be of use when predicting sentiment strength and orientation. As such, the study of *binary and multi-class sentiment classification* will constitute a large body of our empirical evaluations presented in Section 3.4. In addition, the results for *binary document categorization* will be presented, indicating that the SSE model can be applied to other text labeling problems as well. An overview of the experimental setup will be presented in this section. The datasets used in our experiments will be discussed in Section 3.3.1. The baseline methods used to benchmark the proposed SSE model will be presented in Section 3.3.2.

3.3.1 Datasets

Sentiment analysis tasks were evaluated with two large-scale datasets: Amazon [1] and TripAdvisor [2]. The Amazon collection[1] consists of reviews for consumer products on the website `www.amazon.com`, orga-

[1] `http://www.cs.jhu.edu/~mdredze/datasets/sentiment/`

nized into 25 categories, ranging from books, DVDs, and video games to a variety of electronics, automotive, and beauty products. Each review in the Amazon dataset is a snippet of text (usually one or two paragraphs long) that describes a user experience with a product. Each review is also assigned a *sentiment score*, expressed as an integer 1 through 5, that quantifies the user satisfaction with the product on the so-called *five-star scale*. The TripAdvisor[2] dataset contains customer reviews for hotels across the globe. A typical review in TripAdvisor consists of a text snippet that documents a customer's stay at a hotel, as well as numerical sentiment scores for variety of aspects: rooms, location, cleanliness, etc. The sentiment scores for specific aspects are often missing in the TripAdvisor corpus, so we only consider "overall rating" in our evaluations, which is available for almost all reviews in the collection. Both of these collections are considered some of the largest sentiment classification datasets currently available.

Sentiment expressed with a score of 1 star corresponds to the lowest (most negative) sentiment, while the highest (most positive) sentiment is expressed with 5 stars. The TripAdvisor dataset contains *neutral* reviews rated with 3 stars, while neutral reviews were omitted during the construction of the Amazon dataset by their authors. Sentiment rated with 1 or 2 stars will be referred to as *negative,* while *positive* will identify sentiment expressed with 4 or 5 stars. Formulating sentiment analysis as a binary classification task, with two classes that, for example, measure sentiment orientation or subjectivity of a statement, is a prominent approach among related systems that were considered in Section 3.2. As such, *binary sentiment classification*, which assigns sentiment orientation to a statement as either positive or negative, will be considered in Section 3.4. In addition, two multi-class formulations of sentiment classification will be considered that predict sentiment into four or five categories. The five-star or ten-star scales are often used online to record opinions or emotions of users in a quantitative way. It is our belief that multi-class formulation is relevant to sentiment analysis, since it allows us to detect a written expression of emotion at a finer-grained scale. In the binary sentiment classification, the supervised samples in Amazon and TripAdvisor were assigned a positive or a negative score, based on the original five-star scale label, while ignoring the neutral reviews. In the four-class setup, all the reviews except those rated with 3 stars were considered, while five-class formulation involved the neutral samples that were available in the TripAdvisor collection.

[2] `http://times.cs.uiuc.edu/~wang296/Data/`

Table 3.1: Distribution of samples in the datasets.

Dataset	Training	Development	Testing
Amazon-v1	237,900	20,000	110,562
Amazon-v2	257,900	10,000	100,562
TripAdvisor-v1	46,268	3,000	21,116
TripAdvisor-v2	55,306	5,000	10,078
TripAdvisor-wnt-v1	64,445	3,000	28,907
TripAdvisor-wnt-v2	76,483	6,000	13,869
RCV1-23k	23,149	10,000	771,265
RCV1-380k	380,000	10,000	414,414

The Amazon and TripAdvisor collections contain significantly more positive than negative reviews, implying that the datasets are *unbalanced* in terms of binary sentiment (orientation). Unbalanced datasets are likely to have a well-performing binary classifier with a prediction bias towards the label with more supervised samples. In other words, because of the dominating effect of the majority class, this classifier favors predicting the label of the larger class at the expense of correct predictions for the minority category, which contains fewer supervised samples. Indeed, as an extreme example, consider a dataset for binary classification where 99% of supervised samples are assigned with a positive label, and the rest are negative. A trivial classifier that always predicts the positive label will achieve micro-average classification rate of 99% for this dataset, without making a single correct prediction for samples from the negative category. As such, it is preferred to use a balanced collection of supervised samples to train a classifier. This observation motivated the construction of balanced versions of the Amazon and TripAdvisor collections that were used in empirical evaluations in this dissertation.

Following the notions discussed in Section 2.4, balanced sets for *training* and *testing* were created from the samples in TripAdvisor and Amazon. The training and testing sets are balanced in terms of the binary sentiment orientation, meaning each consists of an equal number of positive and negative samples. In total, 70% of the samples were placed into the training set, while 30% were used for testing. In the case of the Amazon collection, training and testing sets for each of the 25 categories were created independently, and then concatenated to form the final sets. The neutral reviews from TripAdvisor were sampled into training and testing sets with a 70%/30% ratio, and added to the appropriate sets with non-neutral samples. *Development*

Table 3.2: Distribution of labels in the sentiment datasets.

Dataset	Split	Review Score					Total
		⋆	⋆⋆	⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆⋆	
Amazon-v1	Training	67,069	51,761	0	31,146	87,924	237,900
	Development	5,656	4,464	0	2,595	7,285	20,000
	Testing	31,228	24,053	0	14,345	40,936	110,562
Amazon-v2	Training	72,725	56,225	0	33,741	95,209	257,900
	Development	2,868	2,196	0	1,240	3,696	10,000
	Testing	28,360	21,857	0	13,105	37,240	100,562
TripAdvisor-wnt-v1	Training	10,129	13,404	17,408	10,224	13,280	64,445
	Development	484	617	769	469	661	3,000
	Testing	4,539	6,019	7,791	4,448	6,110	28,907
TripAdvisor-wnt-v2	Training	11,851	15,801	21,177	11,926	15,728	76,483
	Development	1,073	1,392	1,000	1,115	1,420	6,000
	Testing	2,228	2,847	3,791	2,100	2,903	13,869

sets were constructed by randomly removing samples from the training or the testing sets. Two versions of the sentiment collections were created, each denoted with *ver. 1* and *ver. 2*, or *v1* and *v2* for short. Henceforth, the term *dataset* will refer to a collection of (supervised) samples, organized into three disjoint sets: training, testing, and development. In v1 of the Amazon and TripAdvisor datasets, the development set was obtained from the training sets, while the second version (v2) used testing sets to make more samples available for training. The datasets with neutral reviews will be identified with a "wnt" suffix, i.e., *TripAdvisor-wnt-v1* or *TripAdvisor-wnt-v2*. Table 3.1 provides some statistics for the v1 and v2 of Amazon and TripAdvisor. These datasets are also available online[3]. The processed v1 and v2 datasets are unbalanced in terms of multi-class sentiment label. Table 3.2 provides statistics for the number of samples for each five-star label in the datasets. The results in Section 3.4.1 will report micro-average and macro-average classification error rates for v1 and v2 of the Amazon and TripAdvisor datasets. These measures of classification effectiveness were discussed in Section 2.4. Macro-average classification error is defined as a mean of per-category classification errors, computed independently for every unique category label, and then averaged to produce the macro-average error, while micro-average classification error is an average error computed regardless of the labels assigned to supervised samples.

In addition to sentiment analysis tasks, limited text categorization experiments will be considered to sup-

[3] `http://mst.cs.drexel.edu/datasets/`

port the hypothesis that the SSE representation can be utilized in document labeling methods. Specifically, Reuters corpus (RCV1) [3] dataset was used to benchmark labeling systems on binary topic categorization tasks. Every news article in RCV1 is categorized with one or more topics, drawn from a hierarchy (i.e., taxonomy) of 103 topics. These topics are relevant to the content described in the news articles and correspond to a broad range of concepts, e.g., Equity (M11) and Bond (M12) markets, Labour (C42), Fashion (GFAS), Consumer Prices (E131), etc. The original Reuters Corpus (RCV1) contains training and testing sets with 23,149 and 781,265 supervised samples, respectively. A new dataset was also created with 380,000 training samples. Following the procedure of Lewis *et al.* [3], documents with IDs between 2,286 and 383,792 were used for training in this new dataset. From here on, the dataset with the original (smaller) training set will be denoted by *RCV1-23k*, while the collection with more training samples will be identified as *RCV1-380k*. Development sets for both RCV1 datasets were obtained by randomly sampling 10,000 documents from the testing samples. The numbers of samples per each set in RCV1-23k and RCV1-380k are presented in Table 3.1.

In our experiments, we only consider binary formulation of topic categorization. In this setting, each classifier predicts whether a topic label is assigned to a document. We limit the study of topic categorization to the four largest categories in RCV1. Namely, CCAT (ALL Corporate-Industrial); GCAT (All Government and Social); MCAT (ALL Securities and Commodities Trading and Markets); and C15 (Corporate and Industrial Performance). It is also worth noting that the RCV1 datasets are unbalanced in terms of the number of positive and negative samples that are available for these four topics. As such, macro-average classification error will be used to benchmark classification performance.

3.3.2 Baseline Systems

The SSE model for feature extraction is an adaptive alternative to the BoN model. Thus, the NN labeling system with SSE representation was benchmarked against standard text classification baselines, which rely on the BoN model for feature extraction. Phrases of up-to five words long were encoded in the representation domain, and BoN features were smoothed with a TF-IDF weighting schema, defined in Section 2.1. The BoN representation using n-grams will be denoted with *BoN-*ng, and $|\mathbf{\Gamma_n}|$ will denote the number of unique n-grams in the corresponding training sets with $n \in \{1, 2, 3, 5\}$. The numbers of unique n-grams in $\mathbf{\Gamma_n}$

for some datasets are available in Table 2.1. Clearly, it is desirable to reduce the number of features in the BoN domain when $n \geq 2$. In the case of sentiment datasets, we follow the method used by Blitzer *et al.* [1] to limit the number of features in the BoN domain. All n-grams in Γ_n are sorted in descending order with respect to the mutual information (MI) they share with binary labels (i.e., positive or negative sentiment orientation). Let $J \in \{-1, +1\}$ denote a random variable that assigns binary sentiment to a document, and let $W_{\gamma_j} \in \{-1, +1\}$ denote an random variable for the event that n-gram $\gamma_j \in \Gamma_n$ appears in text. Mutual information of two random variables J and W_{γ_j} is defined as

$$\sum_{J \in \{-1,+1\}} \sum_{W_{\gamma_j} \in \{-1,+1\}} \Pr(J, W_{\gamma_j}) \cdot \log \left(\frac{\Pr(J, W_{\gamma_j})}{\Pr(J) \cdot \Pr(W_{\gamma_j})} \right).$$

For the Amazon-v1 dataset, 10,000 n-grams with the highest mutual information were selected per category, and these 25 sets of phrases were concatenated to form the final BoN vocabulary for the dataset. For the v2 sentiment datasets, a larger vocabulary was used. For Amazon-v2, MI was used to retain 25,000 phrases per category. For TripAdvisor-v1 and TripAdvisor-wnt-v1, MI-based feature selection was used to limit the vocabulary size to 127,000 n-grams. In the case of v2 of TripAdvisor, the vocabulary was limited to 500,000 phrases with highest MI. The size of vocabularies for TripAdvisor datasets was selected to match the size of bigrams used in Amazon-v1 and Amazon-v2. For clarity, the vocabulary of filtered n-grams will be denoted with $\widehat{\Gamma}_n$. For RCV1-23k and RCV1-380k, the size of the vocabulary $\widehat{\Gamma}_n$ was limited to the 500,000 most frequent n-grams in the corpus. The SSE model accepts input text data in its original form, where a document is tokenized by whitespace characters into a sequence of words, and every word is replaced with its integer ID in the unigram vocabulary Γ_1. It is important to note that the vocabulary Γ_1 can be populated dynamically, so a new embedding vector is created every time an unknown word is encountered. However, to ensure a fair comparison of SSE with the BoN baseline, the SSE representation encoded only unigrams that were present in the filtered vocabulary $\widehat{\Gamma}_1$ for the BoW representation. The missing words from the BoW vocabulary were replaced with a "magic" ID $= 0$ that denotes an "unknown" word.

The proposed SSE-based labeling system was benchmarked against standard baseline classifiers, defined in the BoN domain. A linear SVM classifier, as discussed in Section 2.3, is a natural choice for the high-dimensional BoN representation, since efficient training algorithms exist to solve linear SVM optimization.

For instance, there are batch learning algorithms [112, 160] with complexity $O(|\mathcal{X}| \cdot s)$, where s denotes the average number of non-zero entries in the $|\widehat{\Gamma}_n|$-dimensional BoN vectors. In other words, the running time of these algorithms scales linearly with $|\mathcal{X}|$ and s. More recently, an online learning algorithm, based on stochastic primal-dual updates, that scales sub-linearly in term of number of samples $|\mathcal{X}|$, was proposed by Hazan *et al.* [161]. In addition, linear SVM is known to achieve state-of-the-art performance on a variety of text classification methods, including sentiment analysis [67] and text categorization [3, 101, 102]. Intuitively, the exceptional performance of linear SVM can be explained by the high-dimensional nature of the BoN domain, which allows us to find good maximum margin separation between the classes, without transforming the domain with a non-linear kernel. An efficient implementation of the L_2-regularized linear SVM classifier, which is available in the LIBLINEAR[4] software toolkit, was used in the experiments. The penalty parameter C in the formulation of linear SVM was set using the grid search with

$$C = \{2^{-8}, 2^{-7}, \ldots, 2^{10}, 2^{11}\},$$

performed on the development sets. The reported SVM classification errors were then computed using the testing sets with the optimal value of the parameter C.

In addition to linear SVM, a perceptron classifier was trained using the baseline BoN representation. The perceptron baseline was trained using an identical procedure to the one used for the SSE-based system. This will provide additional evidence suggesting that the improvement in sentiment classification is due to the representation (i.e. SSE vs BoN), and not the training procedure or the choice of classifier. All perceptron-based methods were implemented using the Torch5[5] machine learning library. The development set was used to select the best model during the training of all perceptron networks. During the training procedure described in Algorithm 1, NN parameters were evaluated at regular intervals on the entire development set, and the best performing parameters were retained. After the training phase, these parameters were used to compute classification error for testing samples. We report this number for all experiments in Section 3.4.

In addition to SVM and perceptron classifiers trained in the BoN domain, the LTC-based labeling system

[4] http://www.csie.ntu.edu.tw/~cjlin/liblinear/
[5] http://torch5.sourceforge.net/

was considered. The formulation of SSE was motivated by the LTC model that was introduced in Section 2.2. Similar to SSE, LTC gives rise to an adaptive representation that encodes text phrases in a low-dimensional latent space. Using the LTC-based method as a baseline will allow us to study the improvement of the SSE model over LTC in terms of classification accuracy and training time. It is worth mentioning that adaptive models from prior work, discussed in Section 2.2, can also be used to benchmark the SSE model. However, to the best of our knowledge these models do not scale well to handle large datasets like those considered in our evaluations. For instance, Socher *et al.* [55] used collections that are an order of magnitude smaller than Amazon, and two orders of magnitude smaller than the image dataset (ILSVRC2011) used in Chapter 5. As a result, we limit our study to the three baseline labeling systems: linear SVM, perceptron classifier, and the LTC-based method.

The perceptron classifiers were trained with a fixed learning rate $lr = 0.05$. The dimensionality of the latent space in SSE and LTC representations was set to $M = 50$ in all experiments. These parameters were chosen from studying related supervised embedding methods, as well as our prior experience in designing perceptron-based classification systems. Thus, the hyper-parameters lr and M were not subjected to empirical selection. The length of latent phrases that are recognized by LTC and SSE models (i.e., n-gram size) was set after evaluating SVM performance when the BoN model is populated with longer phrases. These results will be presented in Table 3.3. The length of latent phrases that both SSE and LTC could encode was set to $n = 5$. In our preliminary (unreported) studies indicated that setting $n = 3$ for SSE and LTC slightly reduced classification accuracy of both methods.

3.4 Experimental Results

The previous section presented details of our empirical evaluations. Specifically, a comprehensive study of the SSE model was proposed, which is primarily based on the binary and multi-class sentiment classification tasks. The SSE representation was benchmarked against BoN, coupled with a linear SVM classifier, which is a standard baseline for a variety of text classification tasks, including sentiment analysis. In addition to linear SVM, a linear perceptron model, constructed by replacing SSE model with non-adaptive BoN representation, was also used in the experiments. Furthermore, a labeling method with LTC representation, depicted in Figure 2.3, was used as another baseline. All three perceptron systems used an identical classification model,

Table 3.3: Micro-average error for sentiment classification using linear SVM with the BoN representation, where $n \in \{1, 2, 3, 5\}$. Micro-average error is the average misclassification rate over all testing samples. The numbers marked with † are statistically significantly better than **SVM BoW-1g** (BoW with unigrams) with $p < 0.0001$. The numbers marked with $\diamond$ are statistically significantly better than **SVM BoW-2g** (BoW with unigrams and bigrams) with $p < 0.05$.

Dataset ver.	Method	Amazon		TripAdvisor		
		$2 \cdot \star$	$4 \cdot \star$	$2 \cdot \star$	$4 \cdot \star$	$5 \cdot \star$
v1	**SVM BoW-1g**	11.10	30.31	8.89	33.54	43.93
	SVM BoW-2g	7.45†	25.28†	7.47†	32.27	42.34†
	SVM BoW-3g	**7.13**$^{\dagger}_{\diamond}$	**25.02**†	**7.25**†	**32.22**	**42.20**†
	SVM BoW-5g	7.34†	25.67†	7.43†	32.55	42.31†
v2	**SVM BoW-1g**	10.68	29.66	8.97	33.76	44.02
	SVM BoW-2g	6.60†	**23.39**†	7.60	32.05	**42.17**
	SVM BoW-3g	**6.39**$^{\dagger}_{\diamond}$	23.45†	**7.46**†	32.00	43.07
	SVM BoW-5g	6.48†	23.53†	7.53	**31.93**	44.02

defined by *negative log-likelihood* loss in (2.13), and Algorithm 1 was used to train these NN methods. In other words, the three perceptron systems were implemented within an identical NN framework with different models for feature extraction, which, in turn, allowed us to directly study the effect of SSE, LTC, and BoN representations on labeling performance. For clarity, each of the three perceptron systems will be identified by its representation, i.e., **SSE**, **LTC**, and **Prc BoW-ng** with $n \in \{1, 2, 3, 5\}$. Similarly, the labeling method comprised of BoN and SVM will be denoted with **SVM BoW-ng**. In addition, *binary sentiment classification setup* will be denoted by $|\mathcal{Y}| = 2 \cdot \star$, or $2 \cdot \star$ for short. Similarly, $5 \cdot \star$ will denote the *multi-class setup with all five categories* of sentiment, when reviews rated with 1–5 stars are considered. And the *multi-class setup with four categories*, when only reviews rated with either 1, 2, 4, or 5 stars are used, will be denoted with $4 \cdot \star$. Section 3.4.1 will discuss sentiment classification experiments, while the results for binary text categorization, performed on four topics from RCV1 [3], will be presented in Section 3.4.2.

The development of SSE and LTC models were motivated by the observation that encoding n-grams allows BoN to capture compositional semantics in the representation domain. This may improve classification in the case of some text labeling problems, such as sentiment classification. To the best of our knowledge, most prior work did not consider trigrams or longer phrases in their BoN representations. For instance, Wang and Manning reported in their recent work [162] that bigrams could improve sentiment and topic classification. In order to investigate the effect of longer n-grams on text categorization, we trained linear

Table 3.4: Macro-average error for sentiment classification using linear SVM with the BoN representation, where $n \in \{1, 2, 3, 5\}$. Macro-average error is the mean of per-category misclassification rates. These per-category misclassification rates are computed for every class label independently and then averaged together to produce the macro-average error. The Amazon and TripAdvisor datasets are balanced in terms of binary sentiment orientation. The macro-average errors for $2 \cdot \star$ are omitted from this table, since they match the micro-average results in Table 3.4

Dataset ver.	Method	Amazon	TripAdvisor	
		$4 \cdot \star$	$4 \cdot \star$	$5 \cdot \star$
v1	**SVM BoW-1g**	37.99	35.58	46.03
	SVM BoW-2g	30.44	**33.67**	**44.29**
	SVM BoW-3g	**30.06**	33.84	44.33
	SVM BoW-5g	30.66	34.49	44.49
v2	**SVM BoW-1g**	35.78	35.41	46.41
	SVM BoW-2g	28.26	33.68	**44.68**
	SVM BoW-3g	**27.98**	33.50	45.12
	SVM BoW-5g	28.02	**33.45**	46.41

SVM on datasets from Section 3.3.1 using BoN with $n \in \{1, 2, 3, 5\}$. The micro-average and macro-average error rates for the sentiment datasets are presented in Tables 3.3 and 3.4, respectively. These results indicate that trigrams in BoN is better than bigrams for binary and multi-class sentiment prediction on the Amazon dataset. Indeed, the result for SVM BoW-3g on Amazon-v2 in Table 3.3 is statistically significantly better than SVM BoW-2g with p-value $p < 0.05$, even when the same number of features (i.e., $|\widehat{\Gamma}_2| = |\widehat{\Gamma}_3| = 500,000$) were used in both BoN representations. In addition to sentiment analysis, the classification results in Table 3.5 (micro-average error) suggest that topic classification can also benefit from longer phrases in BoN domain. It appears that bigrams encode sufficient lexical information in the representation domain for effective topic labeling on RCV1-23k and RCV1-380k, since SVM BoW-3g does not improve classification performance to any significant extent. Macro-average error rates in Table 3.9 provide additional support for this claim. It is also worth mentioning that the results in Tables 3.3, 3.4, 3.5, and 3.9 show that using more samples for training result in better classification, e.g., SVM test errors for v2 of sentiment datasets is lower than for v1; and the same holds for RCV1-380k and RCV1-23k.

3.4.1 Sentiment Analysis

Sentiment classification results are presented Tables 3.6 and 3.7, which report *micro-average* and *macro-average*, respectively. Since v1 and v2 of sentiment datasets are balanced in terms of binary sentiment

Table 3.5: Micro-average error for topic classification using linear SVM with the BoN representation, where $n \in \{1, 2, 3, 5\}$. Micro-average error is the average misclassification rate over all testing samples. The numbers marked with † are statistically significantly better than **SVM BoW-1g** (BoW with unigrams) with $p < 0.0001$. The numbers marked with $\diamond$ are statistically significantly better than **SVM BoW-2g** (BoW with unigrams and bigrams) with $p < 0.005$.

Model	**RCV1-23k**				**RCV1-380k**			
	CCAT	GCAT	MCAT	C15	CCAT	GCAT	MCAT	C15
SVM BoW-1g	6.32	4.38	4.00	3.87	4.67	3.53	3.04	3.04
SVM BoW-2g	5.70†	**4.10**†	3.75†	3.46†	**4.02**†	**3.31**†	2.73†	2.60†
SVM BoW-3g	**5.68**†	4.15†	**3.73**†	**3.38**$^{\dagger}_{\diamond}$	4.03†	3.37†	**2.72**†	**2.55**†
SVM BoW-5g	5.79†	4.26	3.83	3.39†	4.10†	3.34†	2.80†	2.59†

Table 3.6: Micro-average error for sentiment classification. Micro-average error is the average misclassification rate over all testing samples. The numbers marked with † are statistically significantly better than **SVM BoW-3g** with $p < 0.0001$.

Dataset ver.	**Method**	**Amazon**		**TripAdvisor**		
		$2 \cdot \star$	$4 \cdot \star$	$2 \cdot \star$	$4 \cdot \star$	$5 \cdot \star$
v1	**SVM BoW-3g**	7.13	25.02	7.25	32.22	42.20
	Prc BoW-3g	7.41	27.49	7.31	31.99	41.29
	SSE	**7.04**	**23.59**†	**6.59**	**27.60**†	**37.56**†
	LTC	7.12	27.10	8.33	33.40	42.69
v2	**SVM BoW-3g**	6.39	23.45	7.46	32.00	43.07
	Prc BoW-3g	6.55	23.00	7.54	33.94	43.05
	SSE	**5.69**†	**22.40**†	**6.90**	**33.90**	**42.21**
	LTC	7.05	-	8.49	-	-

orientation, the results for the $2 \cdot \star$ setup match results in Table 3.6 and are omitted from Table 3.7. In the five experiments conducted for each version (v1 and v2) of sentiment datasets, the SSE method outperforms the SVM and Prc baselines that use BoN representation with up-to 5-grams. In addition, the SSE model outperforms LTC in all experiments. We followed the procedure from [157] that initialized the LTC weights with LSI embedding vectors prior to the supervised training. Conversely, the SSE representation did not require this initialization procedure, so the weights were set randomly before the training. Finally, according to the training times for the Amazon-v1 dataset with $4 \cdot \star$ presented in Table 3.8, the time required to learn the SSE representation is significantly shorter than what is required to train LTC. In addition, the training time for SSE is comparable to the training time of the BoN perceptron method.

Table 3.7: Macro-average error for sentiment classification. Macro-average error is the mean of per-category misclassification rates. These per-category misclassification rates are computed for every class label independently and then averaged together to produce the macro-average error. The Amazon and TripAdvisor datasets are balanced in terms of binary sentiment orientation. The macro-average errors for $2 \cdot \star$ are omitted from this table, since they match the micro-average results in Table 3.6

Dataset ver.	Method	Amazon	TripAdvisor	
		$4 \cdot \star$	$4 \cdot \star$	$5 \cdot \star$
v1	**SVM BoW-3g**	30.06	33.84	44.33
	Prc BoW-3g	33.17	32.53	42.83
	SSE	27.88	**28.00**	**38.61**
	LTC	34.22	33.89	43.74
v2	**SVM BoW-3g**	27.98	33.50	45.12
	Prc BoW-3g	26.45	34.73	43.58
	SSE	25.30	34.22	42.88

Table 3.8: Training times for $4 \cdot \star$ setup on the Amazon-v1 dataset.

Method	Time (hrs)
SVM BoW-1g	1.5
SVM BoW-2g	2.3
SVM BoW-3g	2.9
SVM BoW-5g	3.3
Prc BoW-3g	3.0
SSE	8.0
LTC	36.5

3.4.2 Topic Categorization

Table 3.9 presents topic classification results for the RCV1 datasets. Since the number of positive samples in RCV1 is significantly smaller than the number of negative samples, we only consider the macro-average classification error for this dataset. The SSE method outperforms the SVM baseline in all but two experiments (CCAT and C15 in RCV1-380), although the improvement for CCAT in RCV1-23k is rather small.

3.5 Discussion

The details of the proposed adaptive SSE model were presented in this chapter. The labeling performance of SSE was benchmarked against the standard BoN representation with $n = \{1, 2, 3, 5\}$, as well as the adaptive LTC model, described in Section 2.2. The empirical evidence presented suggests that SSE outperforms LTC in sentiment classification and binary topic categorization tasks, while learning the SSE representation takes significantly less time (see Table 3.8). In addition, the SSE model performs statistically significantly better

Table 3.9: Macro-average error for binary topic categorization on RCV1. Macro-average error is the mean of per-category misclassification rates.

Method	RCV1-23k				RCV1-380k			
	CCAT	GCAT	MCAT	C15	CCAT	GCAT	MCAT	C15
SVM BoW-1g	6.45	5.66	5.70	7.95	4.73	4.67	4.33	5.77
SVM BoW-2g	5.82	5.42	5.60	7.62	**4.07**	4.47	3.95	4.93
SVM BoW-3g	5.79	5.53	5.59	7.46	4.08	4.55	3.98	**4.88**
SVM BoW-5g	5.89	5.72	5.75	7.55	4.15	4.59	4.05	4.92
SSE	**5.74**	**4.79**	**4.41**	**6.21**	4.29	**3.81**	**3.42**	5.76

on most of the evaluated tasks. In contrast to the BoN model that requires feature selection, our method relies only on the unigram features and avoids feature selection pre-processing. As such, this chapter presented experimental results in support of the thesis: the adaptive SSE model is superior to the non-adaptive BoN for large-scale text labeling, when dealing with tasks for which n-grams constitute better evidence for classification than individual unigrams do. The adaptive property of SSE facilitates supervised feature extraction that learns low-dimensional representation of unigrams, and encodes phrase semantics as an aggregate of embedding vectors for the unigrams in a phrase. In other words, encoding an n-gram with SSE is an efficient procedure involving only n vector additions. This simple model for compositional semantics encodes n-grams in a low-dimensional latent space, where sufficient evidence is retained in the representation domain to carry out document labeling. In the next chapter, a study of adaptive property of SSE will be presented. Specifically, opinion mining experiments will be used to test our hypothesis that the SSE representation captures sentiment expressed in n-grams, rather than individual words that comprise each phrase.

Chapter 4: Feature Extraction with SSE

An empirical study was presented in Chapter 3 that benchmarked SSE against the standard BoN model. The investigation validated the hypothesis that the SSE model is a viable alternative to BoN, especially when dealing with large-scale labeling problems. It was also mentioned that encoding n-grams with SSE was motivated by the notion of compositionality, which states that the meaning of a complex expression (e.g., n-gram) is defined by the meanings of its elements as well as its structure. With respect to the notion of compositionality, the BoN model can be seen encoding phrases with exterior product selected as the composition function. In that light, the SSE model, which treats an n-gram as a cumulative of its words, can encode much weaker compositional semantics than BoN. On the other hand, similar to the argument provided in Chapter 3, the choice of an aggregate composition function in SSE avoids exponential explosion of features in the representation domain. We believe that the SSE model can still capture phrase semantics, relevant to a specific classification problem. However, unlike BoN, where extraction of n-grams amounts to enumerating symbolic features (i.e., unigrams), SSE requires training to tune the parameters of the compositional semantics model. In other words, SSE has to learn an adaptive representation to capture specific phenomena, which is exhibited by the training data provided in the raw sensory format. As such, in this Chapter the adaptive property of SSE that allows it to encode phrase semantics with a compositional model will be investigated further.

We introduced the notion of an adaptive property for computational models in Chapter 1. Machine learning methods are used to tune the parameters of adaptive models in order to capture the phenomena of interest, exhibited by the training data. The training data usually amounts to a set of input and possibly output exemplars for a given adaptive model. In general, the goal of training an adaptive model is to estimate the underlying probability distribution of the data, or to encode the process that generates output given input exemplars. The adaptive paradigm implies that the same model can be applied to a variety of scenarios in a straightforward manner, as long as an appropriate set of training data is available. For example, a linear SVM classifier can be used to tackle many labeling tasks that admit vector space BoN (or BoW) representations of its sensory data. However, training an SVM does not influence feature extraction that takes place in the BoN

model. Thus, SVM can only leverage knowledge that is retained in the representation domain as evidence for labeling. In this context, an adaptive model for feature extraction would allow a system to encode relevant information directly from training samples, provided in the format of raw sensory data. This implies that data-specific knowledge will be incorporated in the representation and passed on as input to the classifier.

The rest of this chapter is organized as follows. Prior to presenting our argument that SSE encodes lexical semantics of phrases, or *phrase semantics* for short, some related models for adaptive representation and feature embedding will be considered in Section 4.1. These methods will provide additional motivation for the notion of adaptive representations. Next, it will be (implicitly) demonstrated in Section 4.2, that the SSE representation captures relevant semantics of n-grams when trained to predict document-level sentiment. Specifically, the ability of SSE to detect strongly opinionated n-grams will be illustrated. For clarity of the presentation, these phrases will be referred to as *pseudo-subjective* n-grams. The results in this section will motivate a generalization of the SSE model, which will learn to assign n-grams with "confidence" scores, and which will then be used to form document-level embedding from latent phrases. This extension to SSE, which will be referred to as the *weighted supervised sequence embedding (SSE-W)*, will be formally defined in Section 4.3. Consequently, the generalized SSE-W model will be used to quantify pseudo-subjectivity of phrases with scalar quantities in Section 4.4. The empirical evidence will be presented in Section 4.5, showing that SSE-W can be formulated to encode the global structure of each document in the representation domain. Finally, Section 4.6 will present a discussion of the experimental results from this chapter that highlight the properties of feature extraction with the proposed SSE model. In addition, in Section 4.6.1 a study of the effect of feature selection on the BoN representation will be presented. This study will consider how feature selection, which is a form of feature extraction for the BoN model, affects binary sentiment classification accuracy.

4.1 Background

The SSE model is an example of feature extraction that combines dimensionality reduction with feature generation in a single adaptive model, where the goal of feature generation is to encode phrase semantics in the representation domain. In general, the goal of dimensionality reduction is to transform the representation domain into a low-dimensional space while preserving (and enhancing) relevant information. For example,

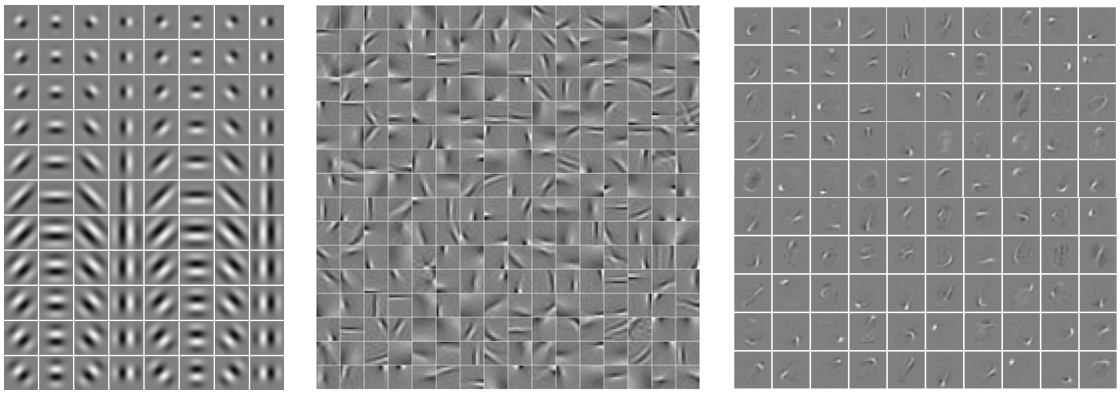
let us consider the dimensionality reduction for BoW vectors proposed by Weston *et al.* [163], which is a discriminative NN model, trained in a supervised fashion. This embedding learns low-dimensional (e.g., 50) vectors for both documents and labels using backpropagation that minimizes the so-called WARP loss function. The WARP loss, which will be discussed in more detail in Chapter 5, enforces the correct ranking of similarity scores between documents and labels, and allows for large-scale annotation of both text documents and images.

Semantic hashing [85] is an example of an adaptive generative model for dimensionality reduction of text data in the BoW domain. This embedding is comprised of three layers of restricted Boltzmann machines that recursively maps a BoW vector with word counts (i.e., TF-only weights) to 128-bit addresses, while preserving semantic similarity between the respective documents. In other words, similar documents in the BoW domain will be assigned to nearby addresses. Here, similarity is measured as the cosine of the angle between two BoW vectors. Mapping BoW to binary addresses allows for efficient retrieval of relevant documents by returning texts whose addresses fall inside a Hamming ball of a fixed radius from the query address. The dimensionality reduction using semantic hashing was evaluated on document retrieval using two public datasets: 20-Newsgroups and Reuters Corpus Volume II [85]. Document retrieval using semantic hashing was benchmarked against the baseline BoW method with TF-IDF weights, which ranked documents in terms of cosine similarity defined in the BoW domain. Retrieving documents using semantic hashing resulted in a slightly lower performance, measured in terms of precision and recall, than the baseline. However, using hashing addresses to preselect a small set of relevant documents (e.g., top 100 or 1000), and then re-ranking them using cosine similarity between the respective BoW vectors, improved document retrieval over the baseline. This observation implies that semantic hashing learns to capture "coarse" topical similarity between documents, that the original BoW representation with TF-IDF could not – i.e., semantic hashing can be used to quickly prune false-positive documents and select a small subset of relevant texts.

Both of the aforementioned models perform dimensionality reduction in the representation domain, and thus are limited to encoding lexical semantics of individual unigrams, captured by the BoW model. Consequently, these embeddings are also prone to the curse of dimensionality of the BoN model, when encoding lexical semantics of phrases is required. In contrast, the SSE model defines phrases as an accumulation of the

embedding vectors for single words in each phrase, thus avoiding learning an exponentially growing number of parameters when encoding larger n-grams. The structure of an n-gram in SSE is encoded by maintaining n individual vectors for every unigram in the dictionary, one for each position in the phrase. In this respect, SSE is similar to the LTC model, discussed in Section 2.2, which was also formulated under the compositionality assumption to capture phrase semantics. However, the composition functional in LTC is defined with a convolutional projection, which involves matrix-vector multiplication to encode every n-gram. The LTC model learns a single embedding vector for every dictionary word, and encodes phrase structure by projecting a vector, composed of latent unigrams from each phrase, into a low-dimensional space with a learned matrix. Another example of representations that make the assumption about compositionality of semantics is based on recursive auto-encoders (RAE) [86, 159]. These models encode phrase structure by recursively combining a current interpretation vector with an embedding vector for the next unigram in the phrase using a learned projection matrix. Conversely, SSE requires only vector additions and multiplications with scalars to compute latent representations of n-grams. This implies that the proposed embedding model is an efficient computational procedure.

Unlike the BoN representation, the SSE, LTC and RAE models give rise to an adaptive representation, which encodes domain-specific lexical semantics of phrases. In other words, the adaptive property of these models "drives" feature extraction towards underlying phenomena, exhibited by the sensory data exemplars. In order to emphasize this point, let us consider an example of learning filter weights in the multi-stage CNN representation that was discussed in Chapter 2. It was mentioned that the multi-stage architecture of the CNN representation is motivated by the structure of the mammalian visual cortex [90]. Indeed, the filter bank layers resemble the simple cells, while their responses are combined in the pooling layers to form a more complex representation, which is akin to the properties of complex cells in the cortex. The parameters of the multi-stage CNN model can be tuned using auto-encoders based on sparse coding, e.g., sparse encoding symmetric machine (SESM) [167], or its approximation, such as the predictive sparse decomposition (PSD) algorithm [165]. Surprisingly, filter banks in CNN that were trained on natural images resemble Gabor wavelets that, in turn, were designed to mimic "receptive-field profiles of simple cells in the mammalian visual cortex" [169]. To illustrate this observation, a visualization of randomly selected filter weights from the



(a) 2D Gabor wavelets. The images were obtained using the MATLAB implementation [164].

(b) CNN model trained with PSD [165] on natural images [166]. Image copyright holder: Koray Kavukcuoglu, http://koray.kavukcuoglu.org/research.html

(c) CNN model trained with SESM [167] on images of handwritten digits [168]. Image copyright holder: Marc'Aurelio Ranzato, http://www.cs.nyu.edu/~ranzato/research/projects.html

Figure 4.1: Illustration of learned weights for the CNN-based image representation. **(a)** 2D Gabor wavelets. **(b)** Filter bank weights from the second CNN stage, learned from natural images using PSD algorithm [165]. The weights resemble "localized" variants of 2D Gabor wavelets. **(c)** Filter bank weights from the second CNN stage, learned from images in the MNIST dataset of handwritten digits using SESM algorithm [167]. These filters learn to detect rudimentary strokes that compose handwritten decimal numerals.

second stage of the CNN representation, trained with the PSD algorithm [165], can be found in Figure 4.1b, while 2D Gabor wavelets [164] are illustrated in Figure 4.1a. The filter banks, learned on natural images, bear resemblance to a "localized" version of the Gabor filters. Interestingly, when applied to a different domain of image data, the CNN representation will learn to detect image structures, specific to the new domain. For example, application of the SESM algorithm [167] to images of handwritten digits (e.g., the MNIST dataset [170]) will produce filters to detect rudimentary strokes that compose handwritten decimal numerals (please refer to Figure 4.1c for an illustration).

Until now, the adaptive property of feature extraction was motivated by examples of trained CNN filters that recognize image structures specific to the training data. The proposed SSE model was introduced to capture a different phenomena in the training data. It was earlier mentioned that the primary goal of SSE is to compress lexical semantics of n-grams into a low-dimensional space while retaining relevant semantic knowledge. In the context of opinion mining, this knowledge amounts to the sentiment which is expressed by each phrase. The results to support this argument will be presented in the following section.

4.2 SSE: Detecting Pseudo-Subjective Phrases

In manual opinion mining a human labeler is tasked with predicting overall sentiment, which is expressed with a five-star scale. A human labeler will often interpret a document-level score as an aggregate of positive and negative subjective statements. Subjective statements are defined as snippets of text, such as phrases, sentences or paragraphs, that express opinions or qualitatively describe personal experiences. In other words, a human reader identifies and quantifies sentiment in all subjective statements in a given text, and then judges the overall sentiment score as a composition of opinions expressed in these phrases. Semantic reasoning is used to quantify each opinion in terms of its strength and polarity. Clearly, both phrases "product-X is pretty good" and "product-X is awesome" identify with positive sentiment, yet the latter corresponds to a more favorable opinion. On the other hand, a review may contain both positive and negative opinionated statements that may influence the document-level sentiment score – e.g., one-star-review contains only negative subjective statements, while a review, rated with two stars, is comprised mostly of negative opinions, but contains some phrases with positive sentiment as well. Consider a sample two-star-review from the testing set of TripAdvisor-v2, where the raw text was pre-processed as described in Section 1.3. The subjective statements in the sample review, that express positive or negative opinions are marked with ☺ and ☹, respectively:

"*noisy air conditioning on NUMBERnd floor*☹ *! ! we stayed one night in the sand villa , in a room on the NUMBERnd floor overlooking the pool .* *the room was comfortable*☺ *.* *there was a loud rumbling noise*☹ *, seemingly from something like a big central air conditioner , that continued all night . it was about as loud as a plane during flight - - certainly not , but not pleasant either .* *the staff was pleasant and helpful*☺ *, but because of the noise i would not stay there again*☹".

Clearly, the opinion expressed in the review above is negative overall, due to the loud noise in the hotel room. However, there are some positive aspects in that review as well (i.e., comfortable room and helpful staff), which merit the 2-star rating for the overall sentiment. This example provides an intuitive justification for computing document-level sentiment as an aggregate function of the opinions in individual phrases, which is the approach taken by the proposed SSE model. Indeed, the mapping $\phi_{\text{mean}}(\mathbf{x})$ in Section 3.1.2 defines embedding of text $\mathbf{x}$ as a centroid of its latent phrases. One notable difference in the analogy of the SSE model with a human labeler is that SSE "treats" all n-grams equally, while a person would base her judgment on in-

terpreting only subjective phrases in each review. As such, SSE is expected to capture sentiment information for all n-grams in the representation domain. The mapping $\phi_{\text{mean}}(\cdot)$ can then be seen compressing this knowledge when forming the document-level embedding vector. Thus, sentiment of individual phrases would be retained in the obtained latent representation of the document, and leveraged as evidence for labeling, when training a classifier.

In what follows, our treatment of SSE, which is aimed to illustrate its ability to learn domain-specific representations for text, is motivated by the idea of quantifying subjectivity of phrases within the computational model. It was mentioned earlier that SSE weights all latent phrases equally when forming the document-level embedding vector, since the centroid function $\phi_{\text{mean}}(\cdot)$ is used to combine the latent phrases. It was also discussed that opinion mining by a human reader involves only subjective statements to produce the judgment about the overall sentiment, expressed in a text. As such, it may be beneficial to account for the subjectivity of each phrase when computing document-level embedding. Henceforth, q_j will denote a weight which will be assigned to the corresponding phrase γ_j in text $\mathbf{x}$ that encodes its subjectivity – larger values q_j correspond to higher likelihood that subjective opinion is expressed in the phrase γ_j. Two approaches to learning the weights q_j will be considered in the following sections of this chapter, and for the sake of this discussion, we assume that the quantities q_j are defined for every γ_j. In general, the subjectivity of a statement γ_j, quantified by q_j, can be viewed as a confidence measure that indicates the significance of each phrase for specific label inference. From a computational perspective, the significance of a phrase γ_j should correlate with its contribution to the document-level embedding. It is worth noting that mapping $\phi_{\text{mean}}(\cdot)$ defines $\forall \gamma_j \in \mathbf{x} \,:\, q_j = \frac{1}{N}$, where $|\mathbf{x}| = N$. In other words, $\phi_{\text{mean}}(\cdot)$ assigns uniform weights q_j to all phrases in the text. As such, the mapping $\phi_{\text{mean}}(\cdot)$ naturally admits a generalized formulation as a weighted average functional, when non-uniform convex combination parameters q_j are assigned to latent n-grams. This mapping, denoted with $\phi_{\text{wght}}(\cdot)$, will be formally defined in Section 4.3.

In order to illustrate our point that SSE captures phrase-level sentiment an SSE model, trained on the $4 \cdot \star$ setup of TripAdvisor-v2, was used to select *pseudo-subjective phrases* in sample reviews. Specifically, an NLL classifier, defined in Section 2.3, was trained to predict a conditional probability distribution over all category labels given text embedding vector $\mathbf{d}_{\mathbf{x}} \in \mathbb{R}^M$. In other words, NLL classifier returns a vector with

conditional probabilities that indicate likelihood of a document $\mathbf{d}_{\mathbf{x}}$ belonging to one of the four (i.e., $|\mathcal{Y}| = 4$) categories. The definition of the mapping $\phi_{\text{mean}}(\cdot)$ implies that document $\mathbf{x}$ and its n-gram embedding vectors span the same M-dimensional latent space. As such, a trained document-level NLL classifier may be used to predict conditional probability distributions over sentiment labels given individual phrases. Unfortunately, both sentiment analysis datasets lack sentence-level ground-truth opinion labels, so this claim can not be validated empirically. Thus, a different strategy is proposed to illustrate that the adaptive SSE learns to encode sentiment of individual phrases in the representation domain. Specifically, the SSE model coupled with the NLL classifier was trained on TripAdvisor-v2 with the $4 \cdot \star$ setup, and every 5-gram γ_j was assigned the weight q_j, defined as the maximum probability value among the four labels:

$$q_j \;=\; \max_{i \in [1,4]} \boldsymbol{\beta}_i^{\top}\, \varphi(\gamma_j), \tag{4.1}$$

where $\boldsymbol{\beta}_i \in \mathbb{R}^M$ denotes the NLL classifier coefficient vector for class i, and $\varphi(\gamma_j) \in \mathbb{R}^M$ denotes the latent embedding of the phrase γ_j. In other words, weight q_j defined by (4.1) quantifies the maximum response attained by the classifier over the four sentiment labels. This approach is guided by our hypothesis that contrary to objective sentences, the trained document-level NLL classifier assigns a probability distribution which is biased towards a particular category.

As it turns out, considering the top three non-overlapping 5-grams, sorted by their respective weights q_j, defined in (4.1), the following phrases were selected from the aforementioned review: "noisy air conditioning on", "staff was pleasant and helpful" and "would not stay there again". The SSE representation was able to capture phrase-level sentiment in the aforementioned sample review. Interestingly, similar results were observed when additional reviews were considered. Specifically, reviews that are shorter than 100 words were randomly chosen among the test samples in TripAdvisor-v2, one for each sentiment score. Table 4.1 shows these reviews along with the three non-overlapping phrases with highest scores, defined in (4.1). It is worth noting that the weights in Table 4.1 were obtained before the application of the LogSoftMax layer (see Figure 3.1).

In this section, a trained SSE model was used to select pseudo-subjective phrases by assigning weights q_j, defined by (4.1), to every n-gram $\gamma_j \in \mathbf{x}$. The motivation for computing the weights q_j that quantify

Table 4.1: Summarization for select TripAdvisor reviews, obtained as the top three non-overlapping 5-grams. The SSE model was trained on TripAdvisor-v2 with the $4 \cdot \star$ setting. The trained SSE model was then used to sort phrases $\gamma_j \in \mathbf{x}$ in a review text $\mathbf{x}$. The weight for each phrase γ_j is defined by (4.1).

Review Text	Rating	5-gram	Weight
disappointing choice this **is one of the worst** large hotels i have ever visited . the suite i had was filthy , and the food from room service **was barely edible (the** caesar salad was dangerously inedible) . there is no wifi . two lamps do not work . feels like a decrepit ocean liner . despite the view and the location , i would **avoid this place at all** cost .	⋆	is one of the worst avoid this place at all was barely edible (the	34.1 31.3 28.6
noisy air conditioning on NUMBERnd floor ! ! we stayed one night in the sand villa , in a room on the NUMBERnd floor overlooking the pool . the room was comfortable . there was a loud rumbling noise , seemingly from something like a big central air conditioner , that continued all night . it was about as loud as a plane during flight - - certainly not , but not pleasant either . the **staff was pleasant and helpful** , but because of the noise i **would not stay there again** .	⋆⋆	staff was pleasant and helpful noisy air conditioning on would not stay there again	30.6 22.1 20.6
very nice experience the frenchmen **is a very nice place** to stay . the rooms were decorated nicely and the courtyard with the **jacuzzi and pool were beautiful** . above all , the staff was probably the friendliest i ' ve ever encountered . very outgoing and pleasant . **the only bad thing i** could say about it is that the rooms were just a little small , but for a single person or a close couple , it was fine .	⋆ ⋆ ⋆⋆	the only bad thing i jacuzzi and pool were beautiful is a very nice place	17.3 16.7 16.3
stylish and great staff i stayed at the hotel globus in may NUMBER as a single female traveller . the room was small but very stylish and spotless . **the staff were all fantastic** and very friendly . **good breakfast and excellent location** for the railway station and easy reach of all florence ' s attractions . i ' m going back to florence in december and will be staying there again .	⋆ ⋆ ⋆ ⋆ ⋆	the staff were all fantastic stylish and great staff i good breakfast and excellent location	26.8 22.1 20.6

pseudo-subjectivity of phrases was two-fold. The primary goal was to illustrate the capacity of the adaptive SSE model to learn latent representation for n-grams to capture their semantics, which amounts to phrase sentiment in the case of opinion mining. The second goal was to introduce the concept of significance for phrases as evidence for labeling, which can be interpreted as a generalization of subjectivity in the case of labeling tasks other than sentiment classification. One approach to learn the scalar weights q_j that quantify the pseudo-subjectivity of phrases will be considered in Section 4.4. This approach will take advantage of the generalized formulation of the SSE model, described in Section 4.3, that computes a document-level representation as a weighted average of its phrases.

4.3 SSE-W: Weighted Supervised Sequence Embedding

The anecdotal evidence presented in Section 4.2 illustrated the ability of SSE to learn latent representation of n-grams that captures phrase semantics. It was shown that the maximum responses from the trained NLL classifier correspond to the statements with the strongest subjective opinions. From another perspective, this observation motivated the notion of significance of certain phrases as evidence for classification, which can be used to adjust contribution of each n-gram when forming document-level embedding vectors. The proposed extension to SSE, referred to as the *weighted supervised sequence embedding* or *SSE-W* for short, is presented in this section. Document embedding in SSE-W will be denoted with the mapping $\phi_{\text{wght}}(\cdot)$ that defines a latent document as a weighted average of its individual n-grams. It is worth mentioning that in the following definition of $\phi_{\text{wght}}(\cdot)$, the function to assign weights q_j for every phrase γ_j is assumed to be known. These weights q_j can then be used as combination parameters to form an embedding vector for the document $\mathbf{x} = (w_1, \ldots, w_N)$:

$$\phi_{\text{wght}}(\mathbf{x}) \equiv \mathbf{d}_{\mathbf{x}} = \sum_{j=1}^{N} q_j \cdot \mathbf{p}_{\gamma_j}, \tag{4.2}$$

where $\mathbf{d}_{\mathbf{x}} \in \mathbb{R}^M$, and each q_j is the so-called *convex combination parameter* for phrase γ_j, so we have:

$$\sum_{j=1}^{N} q_j = 1.$$

It is worth mentioning that the original SSE model, proposed in Chapter 3, is a special case of SSE-W, where uniform combination parameters are used. In addition, let $\varphi_{\text{SSE}}(\mathbf{x}) \equiv \mathbf{P}_{\mathbf{x}} \in \mathbb{R}^{M \times N}$ denote the latent embedding of all n-grams in $\mathbf{x}$. Then the definition of $\phi_{\text{wght}}(\cdot)$ in (4.2) can be re-written as a function of parameters $\mathbf{P}_{\mathbf{x}}$ and $\mathbf{q}_{\mathbf{x}}$:

$$\phi_{\text{wght}}(\mathbf{P}_{\mathbf{x}}, \mathbf{q}_{\mathbf{x}}) \;=\; \mathbf{P}_{\mathbf{x}} \,\text{vec}(q_1, q_2, \,\ldots,\, q_N) \;=\; \varphi_{\text{SSE}}(\mathbf{x})\, \mathbf{q}, \tag{4.3}$$

where convex combination weights $q_j, \forall j \in [1, N]$ are arranged into a column vector $\mathbf{q} \in \mathbb{R}^N$, and $\phi_{\text{wght}}(\mathbf{P}_{\mathbf{x}}, \mathbf{q}_{\mathbf{x}}) \equiv \mathbf{d}_{\mathbf{x}} \in \mathbb{R}^M$.

The combination functional $\phi_{\text{wght}}(\cdot)$ is differentiable with respect to $\mathbf{q}_{\mathbf{x}}$ and $\mathbf{P}_{\mathbf{x}}$, so gradient descent can be used to tune both sets of parameters. Given the partial derivative $\frac{\partial \mathcal{L}}{\partial \mathbf{d}_{\mathbf{x}}} \in \mathbb{R}^M$ of the loss functional $\mathcal{L}(\cdot)$ with respect to the document embedding vector $\mathbf{d}_{\mathbf{x}}$, partial derivatives $\frac{\partial \mathcal{L}}{\partial \mathbf{q}_{\mathbf{x}}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{P}_{\mathbf{x}}}$ are computed using backpropagation [83]:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}_{\mathbf{x}}} \;=\; \frac{\partial \phi_{\text{wght}}}{\partial \mathbf{q}_{\mathbf{x}}}(\mathbf{P}_{\mathbf{x}}, \mathbf{q}_{\mathbf{x}}) \;\frac{\partial \mathcal{L}}{\partial \mathbf{d}_{\mathbf{x}}} \;=\; \mathbf{P}_{\mathbf{x}}^{\top} \frac{\partial \mathcal{L}}{\partial \mathbf{d}_{\mathbf{x}}}$$

and

$$\frac{\partial \mathcal{L}}{\partial \mathbf{P}_{\mathbf{x}}} \;=\; \frac{\partial \phi_{\text{wght}}}{\partial \mathbf{P}_{\mathbf{x}}}(\mathbf{P}_{\mathbf{x}}, \mathbf{q}_{\mathbf{x}}) \;\frac{\partial \mathcal{L}}{\partial \mathbf{d}_{\mathbf{x}}} \;=\; \mathbf{q}_{\mathbf{x}} \frac{\partial \mathcal{L}}{\partial \mathbf{d}_{\mathbf{x}}}.$$

It is also worth noting that a SoftMax NN layer is used in the implementation of SSE-W to enforce the constraint $\sum_{j=1}^{N} q_j = 1$ (please refer to Figure 4.2 for an illustration of the NN architecture for SSE-W).

Having established the feasibility of learning convex combination parameters in SSE-W, the functionals that can learn the parameters q_j from the training data have yet to be discussed. Specifically, the following two approaches will be considered. In the following section, an approach learning combination parameters q_j that quantify pseudo-subjectivity of phrases will be presented. This model computes the parameters q_j using a second SSE projection. The second model will learn convex combination parameters from positions of n-grams in the text. The latter approach, which will be detailed in Section 4.5, is motivated by our hypothesis that statements in the beginning or at the end of each review are more likely to express subjective opinions. As such, these statements provide strong evidence for sentiment classification, and should contribute according to their significance when forming the document embedding for this review. For example, the first sentence

in the two-star review from Table 4.1 describes the broken air conditioner in the room, while the last one informs the reader that the reviewer has no intention of staying at this hotel ever again.

4.4 SSE-W: Learning to Quantify Pseudo-Subjectivity in Phrases

In the previous section, details of the SSE-W model were presented. In this section, an approach will be presented that learns weights q_j to encode phrase subjectivity with a scalar value. This will provide an additional illustration of the capability of SSE to encode phrase semantics in the representation domain. Specifically, the convex combination parameter q_j for phrase γ_j will be assigned with a latent projection of this n-gram into a scalar space, using the second SSE model:

$$q_j = \text{sigmoid}\left(\mathbf{H}\,\widehat{\mathbf{e}}_{\gamma_j}\right), \tag{4.4}$$

where $\widehat{\mathbf{e}}_{\gamma_j}$, defined in (3.1), denotes a vector comprised of word selectors for unigrams in γ_j, and $\mathbf{H} \in \mathbb{R}^{1\times n\cdot|\mathcal{D}|}$ denotes a parameter set for the second SSE projection that assigns non-negative scores q_j to every n-gram $\gamma_j \in \mathbf{x}$. Intuitively speaking, the latent embedding vectors $\varphi_{\text{SSE}}(\gamma_j) \in \mathbb{R}^M$ that are assigned with large weights q_j will dominate the latent embedding of the whole text, defined by $\phi_{\text{wght}}(\cdot)$.

The following procedure was used to evaluate SSE-W with combination weights defined by (4.4). The unigram embedding vectors of SSE-W (i.e., $\mathbf{G} \in \mathbb{R}^{M\times n\cdot|\mathcal{D}|}$) were initialized using the SSE model, pre-trained on the binary setting $2\cdot$ of a sentiment dataset. These embedding parameters $\mathbf{G}$ remained fixed, and the parameters $\mathbf{H}$ were optimized using the multi-class setting (i.e., $4\cdot\star$ for Amazon, $5\cdot\star$ for TripAdvisor) of the same dataset. The parameters of the classifier were also optimized during training. The motivation for this setup is that the unigram embedding parameters $\mathbf{G}$ of the SSE model, pre-trained on the binary sentiment classification setting, would encode phrase semantics in terms of sentiment orientation. Then, training SSE-W for multi-class sentiment classification would learn the parameters $\mathbf{H}$, so the weights $\mathbf{q}_j$ encode pseudo-subjectivity of phrases with respect to their significance at predicting overall sentiment of the reviews. In this way, we follow the earlier assumption that document-level sentiment can be expressed as a convex combination of opinions expressed in the individual subjective phrases.

To illustrate our point, this SSE-W model was trained using Amazon-v2 dataset, and the learned param-

Table 4.2: Selected 5-grams and their combining weights in the SSE-W model. The weights are computed using the model $q_j = \widehat{\mathbf{G}}\,\widehat{\mathbf{e}}_{\gamma_j}$ trained on the Amazon dataset with the binary $2 \cdot \star$ classification setting.

5-gram	Weight
is an extremely good book	3.58
is just a good book	3.19
book is a good buy	2.94
overall a very good book	2.84
book is a good choice	2.81
book is a very good	2.76
book is still very good	1.70
a good book just because	1.15
unless good books are just	1.05

eters $\mathbf{H}$ were used to sort all 5-grams from the testing set of Amazon-v2. Several top-scoring 5-grams that contained the words "good" and "book" can be found in Table 4.2. This anecdotal evidence suggests that the aforementioned variant of SSE-W encodes pseudo-subjectivity of each phrase with a scalar value. Indeed, the obtained list contains phrases that carry positive or negative sentiment. The estimated weights q_j, presented in Figure 4.2, support this argument. For example, one can argue that "is an extremely good book" carries stronger (positive) sentiment than "book is a good choice", which in turn has stronger sentiment than "a good book just because".

4.5 SSE-W: Incorporating Global Document Structure

In this section we investigate a variant of the SSE-W model that allows one to incorporate global structure of the document in the representation domain. The proposed SSE-W model is similar to the work of Lebanon *et al.* [171], where the authors propose a novel semi-parametric generative model for an unsupervised embedding of documents as smooth curves in $\mathbb{R}^{|\mathcal{D}|}$, while preserving the spatial information of phrases within a document. SSE-W is also similar to the recent work by Huang *et al.* [172], where global context was modeled with a scoring function that measured semantic similarity for a phrase-document pair.

Intuitively, latent n-grams in SSE encode local semantic structure (i.e., phrase semantics), while global structure of the document is ignored when these phrases are combined to form the latent document embedding vector. For example, the function $\phi_{\text{mean}}(\cdot)$ described in Section 3.1.2 defines the document embedding as a centroid of all latent phrases in text. Thus, positions of phrases in a document $\mathbf{x}$ are not incorporated in

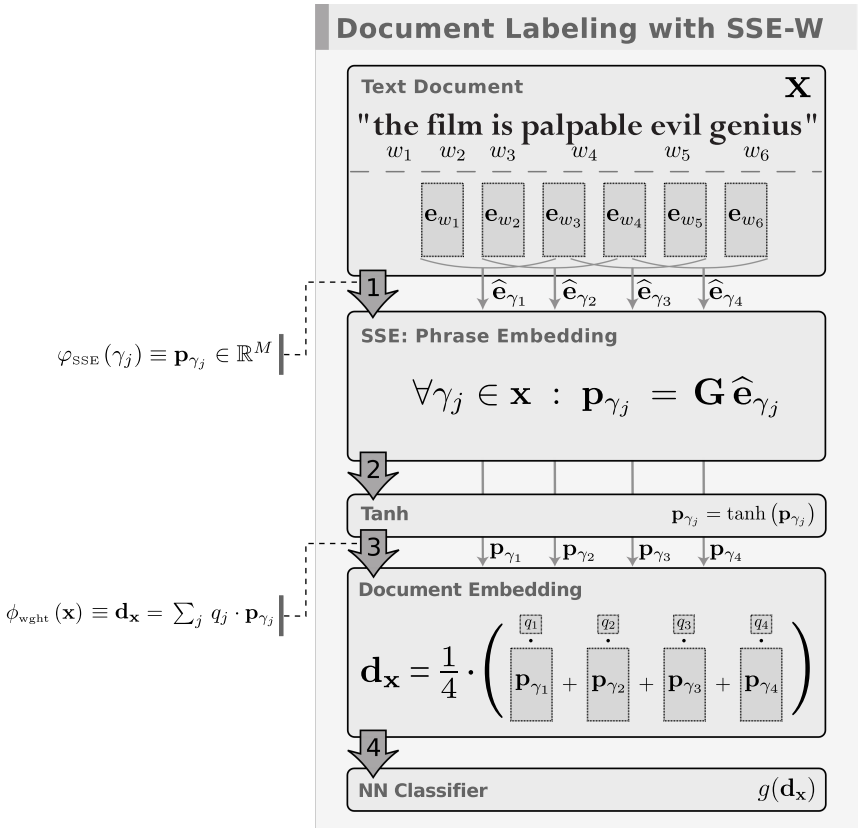


Figure 4.2: Document labeling with weighted supervised sequence embedding (SSE-W). Document embedding is obtained using the map $\phi_{\text{wght}}(\mathbf{x})$, which is defined as a weighted average of latent phrases from text$\mathbf{x}$.

the representation domain. In this section, we propose to mitigate this shortcoming by learning combination weights q_j from the locations of the n-grams γ_j in text $\mathbf{x}$. The proposed formulation was motivated by our hypothesis that the location of a phrase may influence its significance as evidence for labeling. The following mixture model is proposed to infer the convex combination quantities for each phrase $\gamma_j \in \mathbf{x}$:

$$q_j = \frac{1}{Q} \sum_{k=1}^{K} \text{sigmoid}\left(a_k \cdot \frac{j}{N} + b_k\right), \tag{4.5}$$

where a_k, b_k are parameters to be learned, K specifies the number of mixture quantities, sigmoid$(\cdot)$ is a non-linear transfer function, so $\mathbf{q}_j \geq 0,\ \forall \gamma_j \in \mathbf{x}$, and $Q = \sum_{j=1}^{N} q_j$ ensures that $\sum_{j=1}^{N} q_j = 1$. To avoid ambiguity, the combination quantities defined by (4.5) will be referred to as the *spatial weights*.

The aforementioned hypothesis was empirically evaluated using ver. 1 and 2 of the Amazon and TripAd-

Table 4.3: Micro-Average Classification Error Rate for the sentiment datasets. $2 \cdot \star$ denotes binary classification setting, while $4 \cdot \star$ and $5 \cdot \star$ identify multi-class setting with four and five categories, respectively. For each version of the sentiment datasets (v1 or v2), the numbers marked with † (or ‡) are statistically significantly better than **SVM BoW-3g** with $p < 0.0001$ (or $p < 0.001$).

		Amazon		**TripAdvisor**		
Dataset ver.	**Method**	$2 \cdot \star$	$4 \cdot \star$	$2 \cdot \star$	$4 \cdot \star$	$5 \cdot \star$
v1	**SVM BoW-3g**	7.13	25.02	7.25	32.22	42.20
	Prc BoW-3g	7.41	27.49	7.31	31.99	41.29
	SSE	7.04	23.59†	6.59	**27.60**†	**37.56**†
	SSE-W	**7.00**	**23.11**†	**6.43**‡	27.68†	38.09†
v2	**SVM BoW-3g**	6.39	23.45	7.46	32.00	43.07
	Prc BoW-3g	6.55	23.00‡	7.54	33.94	43.05
	SSE	5.69†	22.40†	**6.90**	33.90	42.21
	SSE-W	**5.63**†	**22.05**†	7.01	**31.41**	**40.76**‡

visor datasets. Figure 4.2 provides an illustration of the SSE-W labeling system. The number of mixtures used in (4.5) was fixed to $K = 3$, which was motivated by the assumption that n-grams in the beginning or at the end of each text provide the best evidence for predicting labels. The micro-average error rates for the proposed SSE-W variant, provided in Table 4.3, support our hypothesis. In the following, **SSE** denotes the proposed model for latent phrase embedding with combining function $\phi_{\text{mean}}(\mathbf{x})$, which was defined in (3.3). The SSE model with $\phi_{\text{wght}}(\mathbf{x})$ from (3.3), where convex combination parameters q_j are defined in (4.5), is denoted **SSE-W**.

Trained SSE models were used to initialize the parameters of the unigram embedding vectors in SSE-W, while the parameters in (4.5) were initialized randomly. Both sets of parameters were then updated during supervised training. In the case of the Amazon datasets, SSE-W outperforms SSE in a multi-class setting, though the statistical significance of the improvement is rather small. Indeed, SSE-W outperforms SSE with $p < 0.05$ and $p < 0.005$ for v1 and v2 of the Amazon dataset, respectively. In the case of the TripAdvisor datasets with multi-class settings, SSE-W outperforms the SSE model only when ver. 2 of TripAdvisor is evaluated. This can be attributed to the larger training set in TripAdvisor-v2 and TripAdvisor-wnt-v2 datasets. However, we suspect that the training of SSE models on TripAdvisor-v2 terminated prematurely, so the models did not reach their full labeling capacity. Then, during the training of the corresponding SSE-W models, a better solution was found, which explains the fact that, in the case of TripAdvisor-v2 and TripAdvisor-wnt-v2, SSE-W resulted in statistically significantly better sentiment prediction than SSE with

Table 4.4: Macro-average classification error rate for the sentiment datasets. $2 \cdot \star$ denotes binary classification setting, while $4 \cdot \star$ and $5 \cdot \star$ identify multi-class setting with four and five categories, respectively.

Dataset ver.	Method	Amazon	TripAdvisor	
		$4 \cdot \star$	$4 \cdot \star$	$5 \cdot \star$
v1	**SVM BoW-3g**	30.06	33.84	44.33
	Prc BoW-3g	33.17	32.53	42.83
	SSE	27.88	**28.00**	**38.61**
	SSE-W	**27.78**	28.26	38.68
v2	**SVM BoW-3g**	27.98	33.50	45.12
	Prc BoW-3g	26.45	34.73	43.58
	SSE	25.30	34.22	42.88
	SSE-W	**24.61**	**32.25**	**40.54**

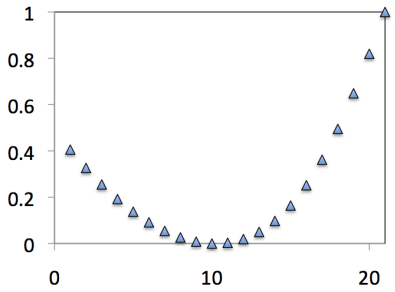


Figure 4.3: Illustration of spatial weights in SSE-W model trained on the Amazon dataset. The values of the spatial weights were computed for a "synthetic" text with 25 words. The weights are scaled into the range $[0, 1]$ for illustration purposes.

p-value $p < 0.0001$. The macro-average results presented in Table 4.4 agree with the micro-average numbers in Table 4.3.

In the case of binary sentiment classification, results are inconclusive. This suggests that spatial re-weighting of phrases in SSE-W only becomes relevant when predicting sentiment strength, expressed with a multi-star scale. Conversely, when predicting binary sentiment, the presence of certain phrases, regardless of their locations in the text, is sufficient for the labeling task. Finally, to make the evaluation of SSE-W complete, an illustration of the spatial weights was created. Figure 4.3 plots learned weights q_j with $j \in \{1, \ldots, 25\}$ for the SSE-W model trained on Amazon-v2 with the binary classification setting, where the combination quantities q_j are computed using (4.5). The illustrated weights have a straightforward interpretation – phrases or sentences that appear in the beginning or at the end of each review are more likely to express strong sentiment that defines the polarity of the review. This observation confirms our earlier assumption.

In addition to sentiment datasets, the performance of SSE-W was benchmarked against SSE on binary

Table 4.5: Macro-average classification error rate for the RCV1 dataset.

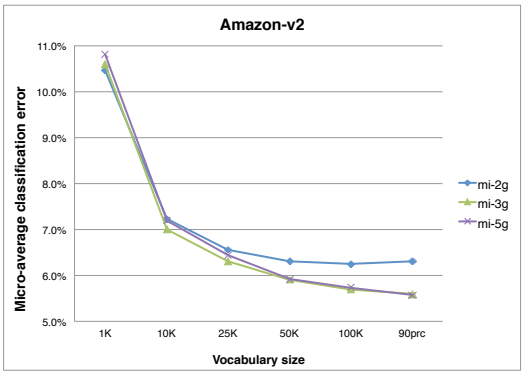
Method	RCV1 23k				RCV1 380k			
	CCAT	GCAT	MCAT	C15	CCAT	GCAT	MCAT	C15
SVM BoW-2g	5.82	5.42	5.60	7.62	**4.07**	4.47	3.95	4.93
SSE	5.74	4.79	**4.41**	6.21	4.29	**3.81**	**3.42**	5.76
SSE-W	**5.71**	**4.70**	4.45	**5.50**	4.15	**3.81**	3.47	**4.28**

topic categorization tasks using the RCV1 dataset. Table 4.5 presents these results, obtained with macro-average classification error. It follows that SSE-W does not improve classification over the SSE model for the MCAT topic, and the improvements are rather small for the CCAT and GCAT topics. On the other hand, the improvement of SSE-W over the SSE method is significant for C15. We speculate that these results can be attributed to the nature of the topics considered. Indeed, MCAT, CCAT and GCAT are high-level topics in RCV1, with each assigned to news articles that describe broad ranges of concepts. On the other hand, C15 identifies articles only related to corporate and industrial performance, thus allowing the SSE-W model to identify the spatial distribution of the effective phrases for this topic.

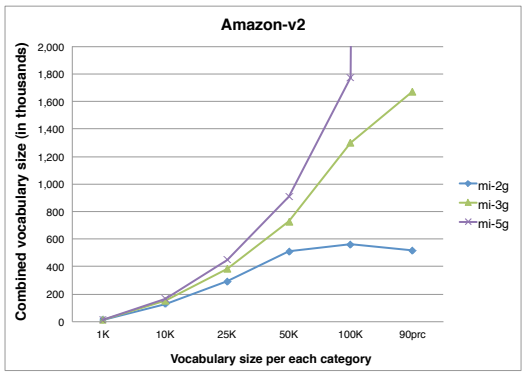
4.6 Discussion

The experimental results presented in this chapter demonstrated the ability of the SSE model to encode lexical semantics of phrases in a low-dimensional latent space. Specifically, trained SSE and SSE-W models were used in Sections 4.2 and 4.4 to detect pseudo-subjective phrases in online reviews. In addition to capturing phrase semantics, weighted supervised sequence embedding (SSE-W) that was proposed in Section 4.3 can be used to encode global document structure in the representation domain. This SSE-W model was empirically evaluated using sentiment classification and binary document categorization tasks. The results presented in Section 4.5 suggest that the SSE-W model may improve the accuracy of multi-class text labeling when dealing with large-scale datasets.

The ability of the SSE representation to capture lexical semantics of n-grams in the representation domain makes it a viable alternative to the BoN model. The BoN representation encodes phrase semantics by considering each unique phrase as an independent feature in the vocabulary Γ_n. The numbers presented in Figure 2.1 suggest that the size of Γ_n for $n \geq 3$ may be huge (i.e., $\Gamma_n = O(|\Gamma_1|^n)$), especially when dealing



(a) Micro-average classification error



(b) vocabulary size $|\widehat{\Gamma}_n|$ (numbers in thousands)

Figure 4.4: **(a)** Micro-average classification error for SVM BoW-ng where $n = \{1, 2, 3, 5\}$, trained on Amazon-v2 with the $2 \cdot \star$ setup. **(b)** Number of unique n-grams in concatenated vocabularies $\widehat{\Gamma}_n$ obtained for the 25 categories in Amazon-v2. Feature selection with mutual information was used to obtain different sizes of the BoN vocabulary $\widehat{\Gamma}_n$. The setups denoted by **1K, 10K, 25K, 50K, 100K** limit $\widehat{\Gamma}_n$ to the specified number of thousands of the top words from Γ_n. **90prc** denotes the setup when the top 90% of all unique n-grams from Γ_n are placed into the filtered vocabulary $\widehat{\Gamma}_n$. Phrases from each category in Amazon-v2 were filtered separately and the resulting 25 vocabularies were concatenated to form the final set $\widehat{\Gamma}_n$. The **90prc** setup contains over 65 million 5-grams in $\widehat{\Gamma}_n$.

with large-scale datasets. Consequently, a feature selection heuristic is required to control the dimensionality of the BoN representation, denoted with $|\widehat{\Gamma}_n|$. A feature selection heuristic is a form of feature extraction for the BoN model (see Section 1.1). It assigns a score to each n-gram in the vocabulary Γ_n, retains a small subset of the top-scoring phrases, and places them into a filtered vocabulary $\widehat{\Gamma}_n$. Thus, only the phrases in $\widehat{\Gamma}_n$ are recognized by the BoN model. In our experimental results presented in Chapters 3 and 4, the SSE and SSE-W models were benchmarked against the BoN representation with a typical setup where a filtered vocabulary $\widehat{\Gamma}_n$ was populated with a pre-determined number of features (e.g., $127,000$ or $500,000$ top-scoring phrases). As such, the effect of feature selection on text labeling performance requires further investigation. Section 4.6.1 will provide the results for our preliminary investigation of sentiment classification using the BoN representation when different sizes of the vocabulary $\widehat{\Gamma}_n$ are used.

4.6.1 Feature Selection with the BoN Representation

The dimensionality of the BoN representation that was used as a baseline in Chapters 3 and 4 remained fixed in all of our empirical studies. Thus, one critical issue that has yet to be addressed is the effect of the dimensionality of the BoN representation on text classification. One possible approach to address this

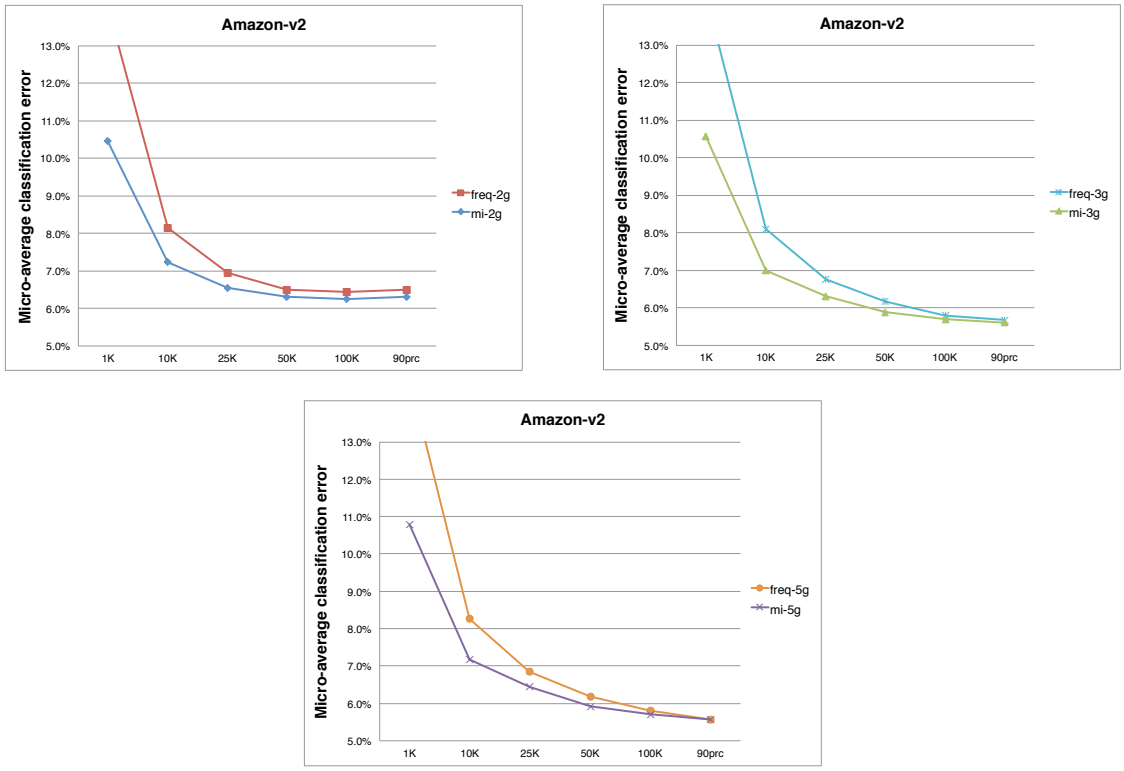


Figure 4.5: Comparison of feature selection using the mutual information (**mi-**n**g**) and term frequency (**freq-**n**g**) heuristics. Micro-average classification error on Amazon-v2 with $4 \cdot \star$.

shortcoming is to evaluate the text classification performance using the same dataset when a feature selection is used to create different sizes of the vocabulary $\widehat{\Gamma}_n$.

In our preliminary investigation, the Amazon-v2 dataset was used to study the effect of the dimensionality of the BoN vectors on binary sentiment classification. Similar to the setup used in our previous experiments, the filtered vocabularies for Amazon-v2 were constructed for each category separately, and the resulting 25 vocabularies were concatenated to form the final set $\widehat{\Gamma}_n$. We considered selecting 1, 10, 25, 50, and 100 thousands of the top n-grams for each category. In addition, a setup when 90% of the top phrases for each category were retained. Figure 4.4 presents the micro-average classification errors for different sizes of $\widehat{\Gamma}_n$ where $n = \{1, 2, 3, 5\}$. The feature selection using mutual information was used in this experiment. The BoN representation with phrases of three words or longer (i.e. 3- and 5-grams) outperformed BoN with bigrams on the binary sentiment classification task. The lowest classification errors using 3- and 5-grams were achieved with the setup that placed the top 90% of all n-grams into $\widehat{\Gamma}_n$.

Table 4.6: Micro-average error for binary sentiment classification. **SVM Opt-BoW-ng** for $n = \{1, 2, 3, 5\}$ correspond to the minimum errors for Amazon-v2 reported in Figure 4.4. The numbers marked with † are statistically significantly better than **SVM Opt-BoW-2g** with $p < 0.0001$.

Method	**Amazon-v2** with $2 \cdot \star$
SVM Opt-BoW-2g	6.25
SVM Opt-BoW-3g	5.60†
SVM Opt-BoW-5g	**5.57**†
SSE	5.69†
SSE-W	5.63†

Feature selection with mutual information is only one possible heuristic that is often used to select the top-scoring n-grams from Γ_n. The so-called term frequency heuristic is another popular choice for feature selection that retains the most frequent phrases from Γ_n. Figure 4.5 compares micro-average classification errors on Amazon-v2 for the BoN representations obtained using feature selection with the mutual information and term frequency heuristics. These results suggest that feature selection with mutual information performs slightly better than the term frequency heuristic. A thorough evaluation of other popular feature selection approaches such as Information Gain [37] or Chi-Square statistics [38], will be deferred for our future work.

The results presented in Figure 4.4 indicate that using larger vocabularies $\widehat{\Gamma}_n$ for the BoN representation can significantly improve sentiment classification, when dealing with large-scale datasets such as Amazon-v2. Table 4.6 lists the lowest micro-average errors from Figure 4.4. the BoN representations with longer n-grams when $n = \{3, 5\}$ perform statistically significantly better than bigrams with $p < 0.0001$. In addition to the improved results for sentiment classification with the BoN representation, Table 4.6 provides the results for SSE and SSE-W that were reported in Sections 3 and 4, respectively. The classification error for SVM Opt-BoW-5g is statistically significantly better than SSE and SSE-W with $p < 0.2$ and $p < 0.4$, respectively. It is important to note that the testing set in Amazon-v2 is rather large – it contains 100,562 samples. This implies that the performance of BoN with trigrams or 5-grams is only marginally better than the performance of the SSE-based methods. As a consequence, we conclude that SSE is effective at encoding long phrases in the representation domain. It matches the performance achieved by the BoN representation, and avoids dealing with a variety of feature selection heuristics that are used to control the dimensionality of the BoN representation.

Chapter 5: Image Representation using SSE

The supervised sequence embedding (SSE) was used in previous chapters to project text phrases into a latent space. The adaptive SSE model was designed to encode phrase semantics under the compositionality assumption, where the structure of the n-grams was encoded as a function of embedding vectors for the individual unigrams. From one perspective, the SSE representation learns to retain the information about the local structure of free texts (i.e., configuration of the unigram features), which is relevant to a specific labeling task. Each word in a text document can be seen as a symbolic feature, which is encoded by a word selector vector with a single non-zero entry. As such, a free text can be seen as a one-dimensional sequence of sparse features, where SSE captures the local spatial distribution of features in the representation domain.

In this chapter, we argue that the same principal can be used to capture spatial configuration (or distribution) of sparse features for sequences in higher dimensions. In order to support this claim, it will be shown that given a 2D sequence of sparse features the same SSE model can be used to encode spatial configuration of these features locally, using two-dimensional sliding windows (e.g., 2×2). Specifically, the SSE model will be applied to encode spatial distribution of sparse features that describe image segments. Applying this framework hinges upon the existence of an image representation that forms sparse high-dimensional vectors from raw image data (i.e., pixel intensities). In other words, this image representation transforms images into a two-dimensional sequence of sparse features, where each feature corresponds to an image segment described by a sparse vector. An approach that converts raw image data into sparse vectors or histograms is a prevalent model for feature extraction in methods that tackle large-scale image classification. These vectors are obtained through the *quantization* of low-level image descriptors (LID) or features, where different types of LID are designed to capture a variety of appearance and shape information in images.

The outcome of the quantization procedure is a transformation that forms image representation, which is analogous to the bag-of-words model. This model for feature extraction from images shares the same downside as BoW – the locations of image features are ignored in the representation domain. Due to this similarity, the above model is often referred to as the *bag-of-features* (BoF) representation for images. It

has been shown that the BoF model can be extended to encode spatial distribution of image features using *Spatial Pyramid Matching* (SPM) [63]. The SSE representation for images can be viewed as an adaptive alternative to SPM, where the SSE layer captures the local structure of the quantized features. The SSE layer is followed by a linear projection that encodes image structure globally. The proposed representation will be benchmarked against the standard SPM model that encodes spatial configuration of the BoF vectors. The use of identical quantization procedure to construct the BoF vectors used by SSE and SPM will allow us to directly compare these models.

The rest of this chapter is organized as follows. The presentation will begin with an overview of popular BoF models in Section 5.1. Next, the application of the adaptive SSE model to encode spatial configuration of image features will be discussed in Section 5.2. Implementation details of the BoF pipeline will be presented in Section 5.3. Both SPM and SSE representations will be benchmarked using an image classification dataset from the Image-Net Large Scale Visual Recognition Challenge 2011 (ILSVRC2011) [60]. The experimental results will be presented in Section 5.4. We will conclude this chapter with a discussion presented in Section 5.5.

5.1 Background

Extensive research efforts in recent years produced significant leap in classification accuracy when dealing with large-scale image datasets. State of the art image classification systems primarily rely on the bag-of-features (BoF) paradigm that transforms dense low-level image descriptors into sparse high-dimensional feature vectors. The improvements in classification accuracy are largely due to new quantization approaches that better capture appearance, shape and texture information from raw image data. The BoF model maps images into a high-dimensional space where a good separation (i.e., hyperplane with a large margin) can be found using a linear SVM classifier. As such, a batch [173] or an online [174] learning method can be used to train this classifier in linear time. On the other hand, complexity of the quantization procedure in BoF can still impact the scalability of these image classification systems.

A typical BoF pipeline consists of *three main stages*: extraction, coding, and pooling of the low-level image descriptors (LID). In the first stage of the pipeline the low-level image descriptors are extracted from a dense lattice over an input image $\mathbf{x}$. Popular choices of LID include HoG [175], SURF [176], SIFT [177],

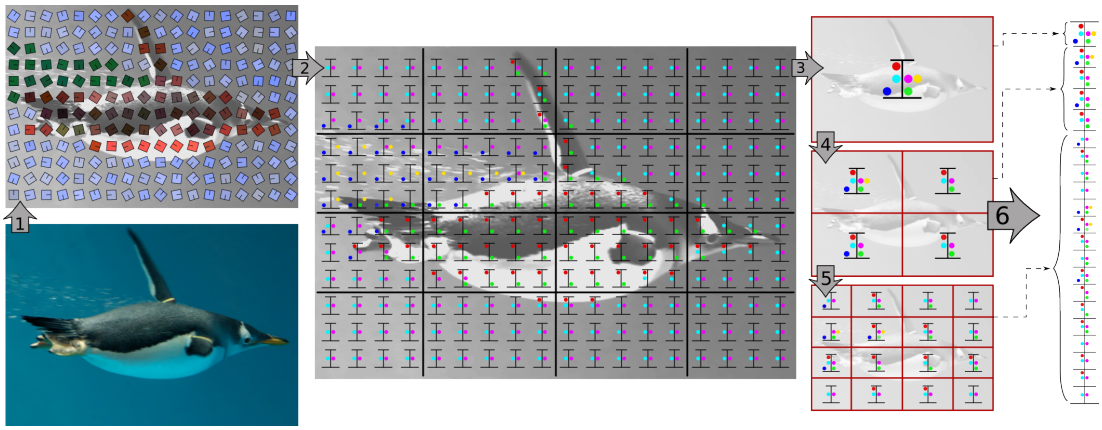


Figure 5.1: The BoF pipeline with SPM: 1) given an input image $\mathbf{x}$, low-level image descriptors (LID) are extracted from a dense lattice over $\mathbf{x}$; 2) each LID is coded with a sparse vector in $\mathbb{R}^{|\mathcal{D}|}$, which is induced by the codebook $\mathcal{D}$; 3–5) SPM partitions the image $\mathbf{x}$ into 1×1, 2×2, and 4×4 segments, and the coded LIDs are pooled to form high-dimensional vectors in $\mathbb{R}^{|\mathcal{D}|}$ that describe these segments; 6) high-dimensional vectors for image segments are concatenated to obtain a representation for the whole image $\mathbf{x}$ in $\mathbb{R}^{21\times|\mathcal{D}|}$. BEST VIEWED IN COLOR.

normalized color histograms [178] and local binary pattern (LBP) [174]. The SIFT or HoG descriptors can be viewed as features that retain shape information [174], while the LBP descriptors extract texture-specific features [174], and color histograms encode color and overall appearance information [178]. We note that multiple types of LID are often used by the same BoF model to enhance visual information retained in the representation domain. The first stage of the BoF pipeline that corresponds to step 1 in Figure 5.1 will be referred to as the *LID extraction*. The second stage of the BoF model will be identified as the *LID coding* stage. The LID coding stage, which is illustrated in step 2 of Figure 5.1, maps the extracted LIDs from an image $\mathbf{x}$ into a high-dimensional space $\mathbb{R}^{|\mathcal{D}|}$. This space is induced by a *codebook* $\mathcal{D}$ of visual-word features or *codewords* (see Figure 5.2). In the third stage, called the *LID pooling*, the results from the coding stage are aggregated to form a single vector that describes the whole image or a segment in $\mathbf{x}$. Steps 3–5 in Figure 5.1 correspond to the LID pooling stage performed using three partitions of the input image $\mathbf{x}$ into 1×1, 2×2, and 4×4 segments. A number of different prior art approaches have been considered at each stage of this BoF pipeline. This section will examine several notable ones.

The LID coding stage involves a codebook $\mathcal{D}$, which is a set of codeword vectors that amount to exemplars for the corresponding LID model. The construction of $\mathcal{D}$ generally involves clustering of LIDs, extracted from

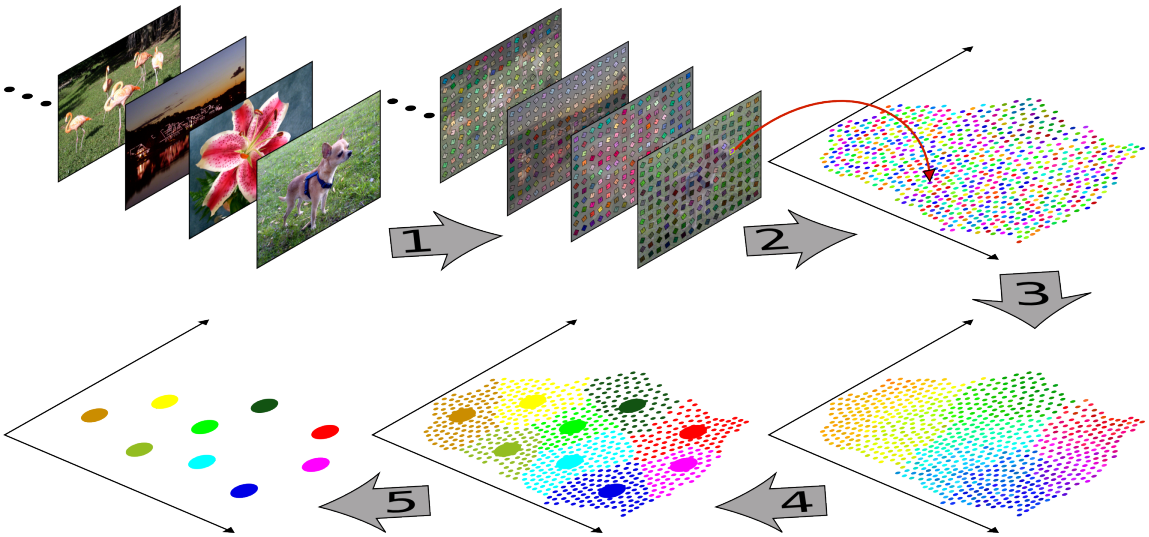


Figure 5.2: Codebook construction for the BoF representation: 1) given a collection of training images, the low-level image descriptors (LID) are extracted from a dense lattice over each image; 2) each LID vector can be seen as a point in a high-dimensional space (e.g., 128 dimensions for SIFT); 3) LIDs extracted from the training images are clustered into a fixed set of clusters (e.g., using k-means); 4–5) centroids of these clusters are selected as codewords and placed into the codebook. BEST VIEWED IN COLOR.

the training samples in a given image corpus. Once the codebook is obtained, every LID can be encoded in terms of its similarity to the codewords. Figure 5.2 provides an illustration of a typical procedure for the codebook construction. The codebook $\mathcal{D}$ is often computed using k-means algorithm applied to a set of randomly sampled LIDs [62, 173, 174, 179]. However, alternative procedures, such as Gaussian mixture models [180–182] or mean-shift clustering [183], can be used to obtain the codebook. In addition, kernel k-means [184] has also been considered for codebook construction in the BoF pipeline [185]. All of the aforementioned procedures for codebook construction can be viewed as adaptive models, since the training samples are involved in the construction of $\mathcal{D}$. It is important to note that similar to the unsupervised learning of the filter bank weights in CNN, these models do not consider class labels. On the other hand, the CNN representation transforms raw pixel intensities into the representation domain, while the BoF pipeline projects LID vectors extracted from images.

Once the codebook $\mathcal{D}$ is constructed, a coding stage is used to transform each LID vector from $\mathbf{x}$ into a high-dimensional (sparse) vector in $\mathbb{R}^{|\mathcal{D}|}$. The coding is then coupled with a pooling stage that combines vectors in $\mathbb{R}^{|\mathcal{D}|}$ to form a representation vector for the whole image $\mathbf{x}$ or one of its segment. The following

three LID coding models have been recently proposed that result in the state of the art image classification: *Fisher vector* [180, 186], *locality-constrained linear coding* [62] and *supervector coding* [187]. These models were proposed as extensions to the standard vector quantization (VQ) approach [63], which identifies several codewords $\mathbf{u}_j \in \mathcal{D}$ with the smallest L_2 distance to the given LID vector $\mathbf{v}$ (i.e., $\|\mathbf{u}_j - \mathbf{v}\|_2$). The original VQ involves a nearest-neighbor datastructure that selects the closest vectors $\mathbf{u}_j \in \mathcal{D}$ for a given LID $\mathbf{v}$. The super-vector coding can be seen expanding VQ with local tangent directions that results in a smoother coding scheme [174]. Fisher vector extends VQ, that uses zero-order statistics or counting, by considering up to the second order statistics when coding LID vectors with codewords. Locality-constrained linear coding [62] (LLC) is an example of a fast approximation for sparse coding [173, 188], which attempt to encode each LID vector $\mathbf{v}$ with as few codewords as possible by solving the optimization:

$$\arg\min_{\mathbf{u}} \|\mathbf{v} - \mathbf{B}\mathbf{u}\|_2^2 + \|\mathbf{u}\|_1,$$

where the codewords from $\mathcal{D}$ are arranged as columns of matrix $\mathbf{B}$, $\|\cdot\|_2$ and $\|\cdot\|_1$ denote L_2 (i.e., Euclidean) and L_1 norms, respectively. The Fisher vector coding is known to outperform the other two [189], albeit the image classification performance of LLC is usually pretty close [62, 180, 185].

The pooling stage usually involves a naive aggregation function that combines the results from the coding stage to form a high-dimensional representation for a segment of the image $\mathbf{x}$. The original vector quantization procedure [63] used average pooling function – i.e., a representation for a segment was defined as the mean of the coded LID vectors in $\mathbb{R}^{|\mathcal{D}|}$. However, it was later established that max-pooling function (i.e., applying $\max(\cdot)$ along each dimension in $\mathbb{R}^{|\mathcal{D}|}$) improves image classification for many coding approaches [173]. The max-pooling is also motivated by the biological evidence obtained from the mammalian visual cortex [190]. Procedures based on sparse coding, including LLC, rely on the max-pooling step in their BoF quantization pipelines. It is also worth noting, that Fisher vector coding estimates the distribution of the LID vectors from a segment in $\mathbf{x}$, which is modeled as a mixture of Gaussians. Thus, this procedure does not require explicit pooling step. Furthermore, when super-vector coding is used, the pooling step is defined by smoothing the Bhattacharyya kernel [174, 191].

The aforementioned three-stage quantization procedure maps LID vectors into a high-dimensional space,

while ignoring relative spatial distribution of the extracted LID vectors. One of the earlier techniques to mitigate this issue, called Spatial Pyramid Matching (SPM) was proposed by Lazebnik *et al.* [63]. SPM partitions an image $\mathbf{x}$ at multiple scales into fixed number of *image segments*. Steps 3–5 in Figure 5.1 illustrate a construction of SPM using three scales with 1×1, 2×2, and 4×4 segments. The BoF vectors in $\mathbb{R}^{|\mathcal{D}|}$ are computed for each image segment and the resulting vectors are concatenated to form a high-dimensional representation in $\mathbb{R}^{k \cdot |\mathcal{D}|}$ for the whole image $\mathbf{x}$, where k denotes the total number of partitions. As a consequence, SPM does not suffer from exponential explosion of features, and does not require explicit feature selection pre-processing as the bag-of-n-grams model for text. The appropriate number of scales is usually determined empirically, with popular choice being 1×1, 2×2, and 4×4 partitions that results in $21 \cdot |\mathcal{D}|$-dimensional image representation (i.e., $k = 1 + 4 + 16 = 21$). The use of multiple partitions of the input image allows SPM to deal with the unknown scale of observations in images, which is akin to the scale-space framework discussed in Chapter 2. In contrast to SPM, the proposed SSE-based representation for images requires partitioning of each image only at a single scale.

The BoF framework gives rise to a variety of representations for images that can be seen mapping color, shape and texture features into a space with dimensionality in tens of thousands to several millions. This allows one to efficiently tackle image classification problems with linear classifiers (e.g., SVM) discussed in Section 3.3.2, since they are known to result in similar performance as their non-linear counterparts when such high-dimensional representation is used [110]. We chose to use LLC [62] in our implementation of the BoF pipeline, since this method offers (close to) the state of the art performance that comes at a very low computational cost. The details of the BoF pipeline used in our experiments will be provided in Section 5.3.

5.2 Image Classification with SSE

Transforming visual information from raw image data into a computationally suitable format, such as the high-dimensional vector-space obtained with the BoF model, is an inherently imprecise procedure. Small variations of the physical environment, such as changes in illumination and viewpoint, occlusions, shape deformations, or variations in (intrinsic) camera parameters inhibit these computational models to extract features that accurately capture appearance and shape information. In this, the BoF framework differs from the bag-of-words model, where every unigram feature has an exact semantic interpretation. As such, a conflux

of image features is used to describe a segment in the BoF representation. From another perspective, the bag-of-n-grams extension to BoW can be seen encoding relative configuration of unigram features locally, while global spatial distribution of features is more important when encoding visual information for image classification [192]. Indeed, the aforementioned SPM procedure captures global configuration of features in partitioned images, where quantized LIDs encode color, shape, and texture information in each segment.

The proposed SSE-based image representation can be seen as an adaptive alternative to the SPM model. Our model follows the compositionality principal from Chapter 3, and encodes visual information in the whole scene as a function of its individual segments. The proposed model computes representation for an image using two latent projection steps from a single partition of the image. In the first step, denoted by map $\varphi_{\text{SSE}}(\cdot)$, SSE encodes the local configuration of image features by projecting all sliding windows of the segments into a low-dimensional latent space. The global structure of the input image is captured in the second step $\phi_{\text{LIN}}(\cdot)$ with a linear projection of the latent patches from $\varphi_{\text{SSE}}(\cdot)$. Let us consider a 3×3 partition of an image as illustrated in Figure 5.3. Let $\mathbf{e}_{w_{i,j}} \in \mathbb{R}^{|\mathcal{D}|}$ denote a BoF vector for segment $w_{i,j}$. Then spatial configuration of features in a 2×2 window of image segments (e.g., $w_{1,2}$, $w_{1,3}$, $w_{2,2}$, $w_{2,3}$) is encoded by concatenating BoF vectors for these segments into a single vector (e.g., $\widehat{\mathbf{e}}_2$ in Figure 5.3). In this, the procedure is similar to the one-dimensional case, described in Chapter 3, where $\mathbf{e}_{w_i}$ in (3.1) denoted a word selector vector with a single non-zero entry at w_i-th index. Here, $\mathbf{e}_{w_{i,j}} \in \mathbb{R}^{|\mathcal{D}|}$ denotes a sparse vector with s non-zero entries, where $\mathcal{D}$ denotes a codebook and $|\mathcal{D}| \gg s$. As such, the SSE layer can be used in the same manner to project all sliding 2D windows into a low-dimensional space.

Given an image $\mathbf{x}$, the SSE-based model first partitions $\mathbf{x}$ into $N \times N$ regions, and the BoF vectors are obtained for these segments. Next, SSE projects all sliding windows of $n \times n$ regions into an $\widehat{M}$-dimensional latent space. Figure 5.4 provides an overview of the proposed model with $N = 3$ and $n = 2$. For a j-th sliding window γ_j consisting of $k = n^2$ regions $\gamma_j = \{w_{j_1}, w_{j_2}, \dots w_{j_k}\}$, vector $\widehat{\mathbf{e}}_{\gamma_j}$ denotes a concatenation of the BoF vectors:

$$\widehat{\mathbf{e}}_{\gamma_j} = \text{vec}\Big(\mathbf{e}_{w_{j_1}}, \mathbf{e}_{w_{j_2}}, \dots \mathbf{e}_{w_{j_k}}\Big). \tag{5.1}$$

where $\mathbf{e}_{w_i} \in \mathbb{R}^{|\mathcal{D}|}$ is the BoF representation of the region w_i. The $\widehat{M}$-dimensional latent embedding $\varphi_{\text{SSE}}(\gamma_j)$

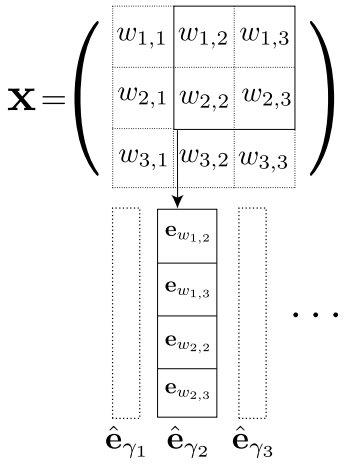


Figure 5.3: Encoding spatial configuration of features in images. The image is partitioned at a single scale into 3×3 segments. A sparse high-dimensional BoF vector for a segment w_i is denoted with $\mathbf{e}_{w_i}$. The spatial configuration of the BoF vectors is encoded with $\widehat{\mathbf{e}}_j$ by concatenating the BoF vectors for the respective image segments. For example, $\widehat{\mathbf{e}}_2$ encodes the segments $w_{1,2}$, $w_{1,3}$, $w_{2,2}$, $w_{2,3}$.

of the j-th window is obtained using

$$\varphi_{\text{SSE}}(\gamma_j) \equiv \mathbf{p}_{\gamma_j} = \mathbf{G}\,\widehat{\mathbf{e}}_{\gamma_j}, \tag{5.2}$$

where $\mathbf{G} \in \mathbb{R}^{\widehat{M} \times n^2 \cdot |\mathcal{D}|}$. We note that the projection (5.2) maintains n^2 independent embedding parameters for each region w_i, based on its position within the j-th sliding window γ_j.

The encoding of all sliding windows with the SSE layer is illustrated in step 2 of Figure 5.4. Step 3 in Figure 5.4 concatenates the latent embedding of all sliding windows into a single vector $\widehat{\mathbf{p}}_{\mathbf{x}}$ defined as:

$$\widehat{\mathbf{p}}_{\mathbf{x}} = \tanh\big(\,\text{vec}\big(\mathbf{p}_{\gamma_1}, \mathbf{p}_{\gamma_2}, \ldots, \mathbf{p}_{\gamma_{N^2}}\big)\,\big), \tag{5.3}$$

where $\tanh(\cdot)$ is the element-wise hyperbolic tangent. The final image-level latent representation $\mathbf{d}_{\mathbf{x}} \in \mathbb{R}^M$ that encodes the global structure of the image $\mathbf{x}$ is the result of the projection step:

$$\phi_{\text{LIN}}(\mathbf{x}) \equiv \mathbf{d}_{\mathbf{x}} = \mathbf{F}\,\widehat{\mathbf{p}}_{\mathbf{x}}, \tag{5.4}$$

where $\mathbf{F} \in \mathbb{R}^{M \times N^2 \cdot \widehat{M}}$. As before, the bias terms were dropped in the definitions of the latent projections (5.2)

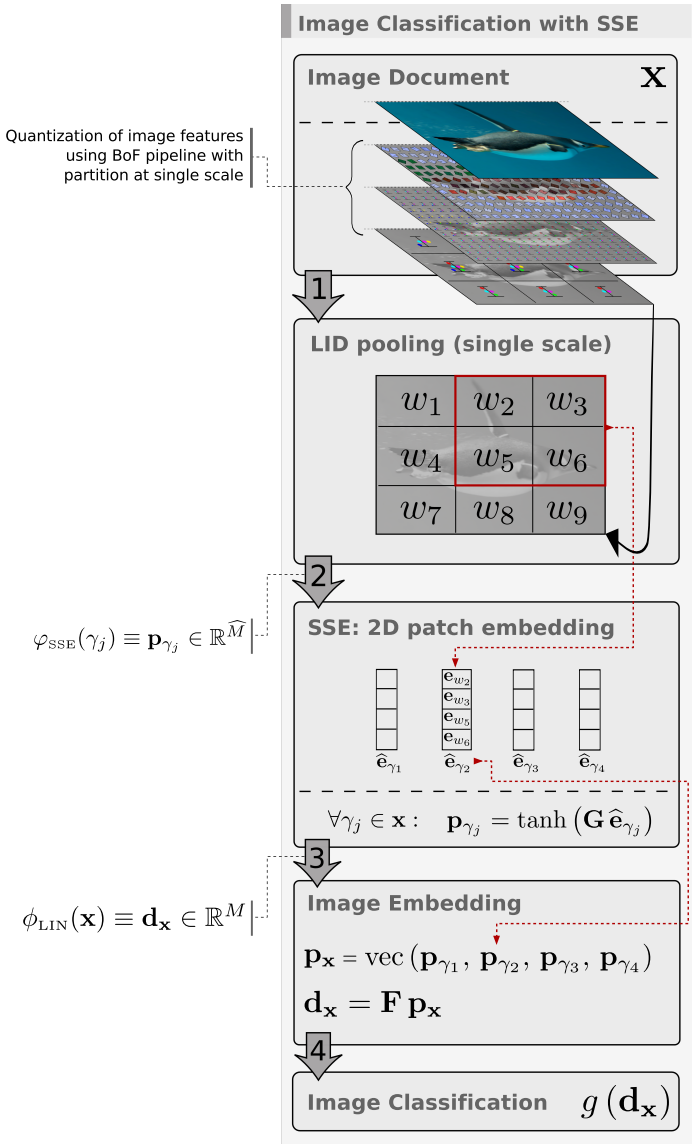


Figure 5.4: Image classification using SSE: 1) input image $\mathbf{x}$ is partitioned into 3×3 segments and the BoF vector $\mathbf{e}_{w_i}$ is obtained for each segment $w_i \in \mathbf{x}$; 2) SSE projects all sliding windows of 2×2 segments into an $\widehat{M}$-dimensional space; 3) linear projection $\phi_{\text{LIN}} : \mathbb{R}^{4 \cdot \widehat{M}} \rightarrow \mathbb{R}^{M}$ is used to obtain an M-dimensional representation for $\mathbf{x}$; 4) image labeling is carried out with a neural network classifier, such as NLL from (2.13).

and (5.4).

In summary, SSE projects all sliding windows each containing n^2 regions into a latent space using projection (5.2). The latent representations of these windows are then concatenated to form the vector $\widehat{\mathbf{p}}_{\mathbf{x}}$ in (5.3). A linear projection layer is used to obtain the image-level latent representation $\mathbf{d}_{\mathbf{x}}$ in (5.4). From one per-

spective, the proposed method resembles the construction of SPM in a latent space. On the other hand, the SPM procedure constructs image-level representation in an unsupervised fashion, while $\varphi_{\text{SSE}}(\cdot)$ performs a supervised feature extraction at an intermediate level of the spatial pyramid (see step 2 in Figure 5.4), and $\phi_{\text{LIN}}(\cdot)$ computes the image-level representation for $\mathbf{x}$ in a supervised latent space (see step 3 in Figure 5.4). It is this latter encoding, $\mathbf{d}_{\mathbf{x}}$, of the image $\mathbf{x}$ that will be used as the input to the classifier. It was discussed in Section 2.2 that SSE is closely related to the convolutional neural networks (CNN). However, to the best of our knowledge image classification methods with CNNs are generally restricted to small or medium-scale datasets e.g., Caltech-101/256 [193, 194], PASCAL07 [195]. In contrast, our image classification system explores a single image partition to encode the spatial distribution of the sparse BoF vectors. To be fair, recent advancements in the CNN implementation, such as distributed parallelization [196], utilization of GPUs [97, 197] and FPGAs [198], will eventually enable these CNN-based labeling systems to handle large-scale image datasets.

5.3 Baseline

The SSE-based representation that was described in Section 5.2 is an adaptive alternative to the SPM procedure, that encodes spatial configuration of quantized image features. Both the SSE and SPM models were empirically evaluated using the identical BoF pipeline, whose details will be presented in Section 5.3.1. The proposed representation model, as well as the baseline SPM, can be coupled with a vector-space classifier to predict image labels. In the experimental results, that will be presented in Section 5.4, the NLL classifier from (2.13) was considered. In addition, image classification using so-called WARP loss was considered in our empirical evaluations. The WARP loss optimizes the ranking of category labels and query images that share a low-dimensional latent space. The benchmark dataset, ILSVRC2011 [60], contains over 1.2 million training images which are organized into 1,000 categories. The WARP loss, that will be discussed in Section 5.3.2, is a preferable approach to categorization when the number of classes is large [199]. It is also worth noting that the large size of the training set in ILSVRC2011 required distributed training of the classifiers, thus only perceptron systems were considered in these experiments. The details of the training procedure will be presented in Section 5.3.3.

5.3.1 Bag-of-Features Pipeline

The BoF pipeline (see steps 1–6 in Figure 5.1) was implemented using the VLFeat[1] computer vision library. For the LID extraction, images were rescaled with the largest dimension set to 300 pixels. We used grayscale SIFT [177] descriptors extracted at every 8 pixels using square regions with scales 8, 16, and 24 pixels. 30 million SIFT descriptors randomly chosen from the training images were clustered into $|\mathcal{D}| = 4,096$ leaf clusters (i.e., codewords) using the hierarchical k-means. The coding and pooling layers of the BoF pipeline closely followed the implementation of the locality-constrained linear coding (LLC) [62]. Let $\mathbf{B} \in \mathbb{R}^{Q \times |\mathcal{D}|}$ denote a matrix, whose columns correspond to the Q-dimensional codewords from the codebook $\mathcal{D}$. In addition, let $\mathbf{b}_i \in \mathbb{R}^Q$ denote i-th column in $\mathbf{B}$. Then, given an LID vector $\mathbf{v} \in \mathbb{R}^Q$ extracted from image $\mathbf{x}$, LLC encodes $\mathbf{v}$ with a sparse vector $\mathbf{c} \in \mathbb{R}^{|\mathcal{D}|}$ that minimizes:

$$\begin{aligned} \min_{\mathbf{c}} \quad & \|\mathbf{v} - \mathbf{B}\mathbf{c}\|_2^2 + \lambda \|\mathbf{q} \odot \mathbf{c}\|_2^2 \\ \text{s.t.} \quad & \mathbf{1}^\top \mathbf{c} = 1, \end{aligned} \tag{5.5}$$

where $\odot$ denotes element-wise multiplication, $\mathbf{1}$ is a vector of ones, and $\mathbf{q} \in \mathbb{R}^{|\mathcal{D}|}$ is the *locality adaptor* that quantifies the similarity of $\mathbf{v}$ to the codewords in $\mathcal{D}$. Specifically, the locality adaptor is defined:

$$\mathbf{q} = \exp\left(\frac{\left[\|\mathbf{v} - \mathbf{b}_1\|, \, \|\mathbf{v} - \mathbf{b}_2\|, \, \ldots, \, \|\mathbf{v} - \mathbf{b}_{|\mathcal{D}|}\| \right]}{\sigma} \right), \tag{5.6}$$

where $\|\mathbf{v} - \mathbf{b}_i\|$ denotes the Euclidean distance between the LID vector $\mathbf{v}$ and the codeword $\mathbf{b}_i$, and σ controls the similarity decay speed for the locality adaptor. Intuitively, the regularization with the locality adaptor in (5.5) weights each codeword proportional to its similarity with $\mathbf{v}$. In other words, the locality adaptor "promotes" codewords that closely match the LID vector $\mathbf{v}$.

For every LID vector $\mathbf{v}$, LLC limits the number of codewords in (5.5) to the k closest vectors from the codebook using approximate nearest neighbor datastructure. Thus, every LID is encoded with a vector $\mathbf{c}$ that contains k non-zero entries (i.e., $\|\mathbf{c}\|_0 = k$). The optimization (5.5) admits an analytical solution [62]. The LLC procedure can be seen as fast approximation of sparse coding approaches [173, 188] that consider entire

[1] `http://www.vlfeat.org/`

codebook when solving their coding optimizations iteratively. As a consequence, LLC enables one to handle large-scale image datasets, while providing (close to) the state of the art classification performance.

In our implementation of the BoF pipeline, each LID vector $\mathbf{v}$ was coded with $k = 5$ closest codewords from $\mathcal{D}$. Similar to the original LLC procedure [62], the max-pooling step was used in our BoF pipeline. In our experiments, the SPM baseline was computed using three scales: 1×1, 2×2 and 4×4. The latent embedding of the SPM representation ($\mathbf{d}_{\mathbf{x}} \in \mathbb{R}^M$) was obtained using a single sparse linear projection $\phi_{\text{BoF}} : \mathbb{R}^{21 \cdot |\mathcal{D}|} \rightarrow \mathbb{R}^M$.

5.3.2 WARP Loss Classifier

In addition to the NLL classifier described in Section 2.3, the ranking-based classifier was also considered in our evaluations. This classifier was trained to minimize the WARP loss, which was proposed by Weston *et al.* [163] in order to reduce the training time and storage complexity for classification with many category labels. In the case of a supervised embedding based on the WARP loss [163], information relevant to a specific task (e.g., text or image annotations) is preserved in the low-dimensional latent space where the dot product between a document and a label captures their ground-truth relevance ranking. In other words, the WARP loss learns an embedding vector $\mathbf{v}_{y_k}$ for a category label $y_k \in \mathcal{Y}$ so that a relevant training document

$$\mathbf{x}^+ \in \{\mathbf{x}_j \mid \exists j \in [1, |\mathcal{X}|],\, (x_j, y_k) \in \mathcal{X}\}$$

is ranked higher than any training document

$$\mathbf{x}^- \in \{\mathbf{x}_j \mid \forall j \in [1, |\mathcal{X}|],\, (x_j, y_k) \notin \mathcal{X}\},$$

which is not assigned to the topic y_k. Hence, we have that

$$\mathbf{v}_{y_k}^\top \mathbf{d}_{\mathbf{x}^+} \;>\; \mathbf{v}_{y_k}^\top \mathbf{d}_{\mathbf{x}^-},$$

where $\mathbf{d}_{\mathbf{x}^+}$ and $\mathbf{d}_{\mathbf{x}^-}$ denote the embedding vectors for the respective documents.

The WARP loss optimizes margin-penalized ranking of pairwise similarity values between training im-

ages and their labels. In addition to learning latent representation $\phi(\cdot)$ for images, the WARP classifier computes an M-dimensional latent representation for each label $y_i \in \mathcal{Y}$. Let $\mathbf{V} \in \mathbb{R}^{M \times C}$ denote the parameters for a latent embedding of C labels, where $\mathbf{v}_{y_i}$ are arranged into the columns of $\mathbf{V}$. The classifier is then computed by minimizing the WARP loss:

$$\mathcal{L}_{\text{WARP}}(\mathcal{X}) = \sum_{i \in \{1..|\mathcal{X}|\}} \Lambda \left(\sum_{y_j \neq y_i} I \left[\xi_{\text{WARP}}(\mathbf{x}_i, y_j) > \xi_{\text{WARP}}(\mathbf{x}_i, y_i) \right] \right), \tag{5.7}$$

where $\Lambda(k) = \sum_{j=1}^{k} \frac{1}{j}$, and $I(\cdot)$ is the indicator function. The likelihood measure $\xi_{\text{WARP}}(\mathbf{x}, y)$ in (5.7) for an image $\mathbf{x}$ and a label $\mathbf{y}$ is defined as

$$\xi_{\text{WARP}}(\mathbf{x}, y) = \mathbf{v}_y^\top \, \mathbf{d}_{\mathbf{x}}. \tag{5.8}$$

In practice, in order to make the WARP loss (5.7) continuous, the indicator function $I(\cdot)$ is approximated with a hinge loss (2.10), that adds a fixed margin μ to the ranking formulation. The prediction for an image $\mathbf{x}$ using the classifier optimized with the WARP loss is obtained using:

$$g_{\text{WARP}}(\mathbf{d}_{\mathbf{x}}) = \underset{i \in \{1..C\}}{\arg\max} \; \xi_{\text{WARP}}(\mathbf{x}, y_i). \tag{5.9}$$

5.3.3 Parameters and Training

In the case of the SSE method, every image was partitioned into 4×4 segments and all sliding windows of 3×3 segments were projected into a latent space with dimensionality $\widehat{M} = 100$. The latent dimensionality of the SSE and SPM representations were set to $M = C = 1000$ when the NLL classifier was used. In the case of the WARP classifier, the dimensionality of the latent space was set to $M = 300$. The margin parameter of the hinge loss (2.10) used to approximate $I(\cdot)$ in (5.7), was set to $\mu = 0.1$. The hyper-parameters of LLC were set to $\lambda = 500$ and $\sigma = 100$ in (5.6) and (5.5), respectively.

The 1.2 million training images from ILSVRC2011 were split into 30 slices so that the samples for each synset were divided evenly among the slices. The training proceeded in a distributed fashion with ten work nodes, each updating the parameters of a *work model* using 35,000 samples from a randomly selected slice. The computed work models were then combined into the latest *master model* by averaging the respective pa-

rameters from the work models. The work nodes proceeded with the training by initializing their parameters with the latest master model. The master models were regularly evaluated using validation images and the *best performing model* was retained for each method. After the classification performance stopped improving, we used the best performing model to obtain label predictions for the test images and submitted these predictions to the ILSVRC2011 evaluation server [2]. For both SSE and SPM methods, the training took less than a week to complete.

5.4 Experimental Results

We evaluate the proposed method using the dataset from the Image-Net Large Scale Visual Recognition Challenge 2011 (ILSVRC2011) [60]. The dataset contains images of 1,000 categories of objects where each category corresponds to a synset (i.e., a set of synonymous nouns) in WordNet. The categories are organized as leaf nodes into a hierarchy that corresponds to a subset of the WordNet synset hierarchy. The competing methods in ILSVRC2011 were evaluated using two cost measures. The first one, called the *flat cost* (FL), measured classification hit rate among the top five label predictions produced by each method. Let y_{l_j}, $j \in [1,5]$ denote the predicted labels for an image $\mathbf{x}$, and y_t denote the ground-truth label for $\mathbf{x}$. The FL prediction error for $\mathbf{x}$ computed using the flat cost is defined as:

$$\min_{j \in [1,5]} \delta(y_{l_j}, y_t), \tag{5.10}$$

where $\delta(y_i, y_k) = 0$, if $y_i = y_k$ and $\delta(y_i, y_k) = 1$ otherwise. The *hierarchical cost* (HI) is the second cost measure used in ILSVRC2011. HI used the minimum height of the lowest common ancestor of the ground-truth and predicted labels. Namely, the HI prediction error for an image $\mathbf{x}$ was computed using (5.10), where $\delta(y_i, y_k) = s$, and s is the height of the lowest common ancestor of $\mathbf{y}_i$ and $\mathbf{y}_k$ in the ILSVRC2011 category hierarchy divided by the maximum height in this hierarchy (i.e., a subset of WordNet). Intuitively, the HI cost is the equivalent of predicting a path along the hierarchy and evaluating where the ground-truth path and the predicted path diverge. For both FL and HI cost measures, the overall error was defined as the average error (5.10) over all test images. The classification errors computed with five label predictions for

[2] `http://www.image-net.org/challenges/LSVRC/2011/test_server`

Table 5.1: Image classification error. FL and HI denote flat and hierarchical cost, respectively. The FL numbers marked with † are statistically significantly better than the SPM baseline with $p < 10^{-4}$. Classification results are obtained using one (Top-1) and five (Top-5) label predictions for each test image.

		NLL		**WARP**	
Cost	# Lbls	**SPM**	**SSE**	**SPM**	**SSE**
FL	Top-5	60.7	56.5†	60.9	59.6†
FL	Top-1	77.5	75.5†	80.3	79.4†
HI	Top-5	29.0	26.6	28.8	28.3
HI	Top-1	46.2	43.7	46.3	45.3

each image will be denoted with **Top-5**. In addition, **Top-1** will be used to denote the FL and HI errors that were obtained using a single predicted label for each test image. The FL and HI classification errors can be found in Table 5.1.

The SSE-based method resulted in statistically significant improvement over SPM with $p < 10^{-4}$ for both NLL and WARP classifiers. It should be noted that our method relied on construction of the BoF vectors for an image partition at a single scale, and performed the low-dimensional embedding via a two-stage latent projection. The baseline SPM method on the other hand, required extraction of the BoF vectors at three scales. The empirical evidence suggests that irrespective of the classifier, the SSE-based method outperformed the baseline that relied on the SPM procedure to capture the spatial distribution of image features.

We also note that the classification performance of both SPM and SSE methods corresponds to the tail of the ranked list of participants in ILSVRC2011. Moderately chosen parameters of the BoF pipeline allowed us to train the models in a timely fashion but significantly degraded the classification performance. For example, a variant of LLC [174] resulted in the state of the art performance on ILSVRC2010 [200]. The method scaled images to at most 500 pixels, used 20,480 codewords, two types of image descriptors (HoG and LBP), and 20 closest codewords to describe each LID. Our BoF pipeline scaled images to 300 pixels, constructed the codebook with $|\mathcal{D}| = 4,096$ from a single type of LID (SIFT), and coded each LID vector with only five closest codewords.

Table 5.2: Image classification using three types of the low-level image descriptors: SIFT, normalized color histograms, and local binary pattern. FL and HI denote flat and hierarchical cost, respectively. The FL numbers marked with † are statistically significantly better than the SPM baseline with $p < 10^{-4}$. Classification results are obtained using one (Top-1) and five (Top-5) label predictions per image.

		NLL	
Cost	# Lbls	**SPM**	**SSE**
FL	Top-5	54.6	$39.2^{\dagger}$
FL	Top-1	72.3	$61.5^{\dagger}$
HI	Top-5	26.3	17.2
HI	Top-1	41.5	32.8

5.5 Discussion

The BoF pipeline that was used in our empirical evaluations presented in Section 5.4 employed only one type of LID, namely the SIFT descriptors [177]. SIFT can be seen as capturing shape information in images [174]. As such, we performed an additional set of experiments, where in addition to SIFT normalized color histograms (NCH) [178] and local binary patterns (LBP) [174] were also used to capture color and texture information in the BoF model, respectively. For each LID type, the BoF vectors were computed using the same setup as described in Section 5.3. Only NLL classifier was trained for this experiment.

In the case of the baseline classification system with SPM, the BoF vectors for each LID were concatenated and the label predictions were obtained using the map

$$\phi_{\text{BoF}} : \mathbb{R}^{63 \cdot |\mathcal{D}|} \rightarrow \mathbb{R}^{M},$$

where $|\mathcal{D}| = 4,096$, and $M = C = 1000$. For the proposed image representation with SSE, different types of LID were combined in the second latent projection step (see step 3 in Figure 5.4). Let $\widehat{\mathbf{p}}_{\text{SIFT}}$, $\widehat{\mathbf{p}}_{\text{NCH}}$, and $\widehat{\mathbf{p}}_{\text{LBP}}$ denote the concatenation of the latent windows from an image $\mathbf{x}$, obtained with (5.3) using the respective types of LID. Then, the map $\phi_{\text{LIN}}(\mathbf{x})$ that computes M-dimensional image embedding takes the form:

$$\phi_{\text{LIN}}(\mathbf{x}) = \mathbf{F}\, \text{vec}(\widehat{\mathbf{p}}_{\text{SIFT}}, \widehat{\mathbf{p}}_{\text{NCH}}, \widehat{\mathbf{p}}_{\text{LBP}}),$$

where $\mathbf{F} \in \mathbb{R}^{M \times 3 \cdot \widehat{M} \cdot N^2}$, $M = C = 1000$, $\widehat{M} = 100$, and $N = 3$. The HI and FL classification errors

are provided in Table 5.2. The results indicate that combining different descriptors in the second projection step benefited the SSE-based system, since its improvements in the classification is more significant when compared with the results in Table 5.1. In addition, the improvements of the SSE-based method for Top-5 and Top-1 are statistically significant with $p < 10^{-4}$. Finally, it is worth noting that the Top-5 FL error of 39.2% rivals the systems that participated in the ILSVRC2011. For instance, the best performing method in ILSVRC2011 that used LLC in their BoF pipeline reached the Top-5 FL classification error of 35.9%.

Chapter 6: Discussion and Conclusion

Supervised sequence embedding (SSE) that was proposed in this dissertation is an adaptive representation that efficiently encodes spatial distribution of sparse features. In Chapter 3 the SSE model was first introduced to encode text phrases in a low dimensional latent space. This SSE model can be seen as an adaptive alternative to the BoN representation. The SSE and BoN models were benchmarked on the sentiment classification and binary topic categorization tasks. The dimensionality of BoN suffers from exponential explosion of features when encoding long n-grams ($n \geq 3$). As such, the BoN model requires feature selection pre-processing in order to keep the dimensionality of the BoN vectors tractable when dealing with large-scale text labeling tasks. In contrast to the BoN model, SSE avoids feature selection pre-processing since its parameter space grows linearly with the length of n-grams that SSE can encode. The experimental results presented in Chapter 3 validated our hypothesis that the SSE model is a viable alternative to the BoN representation. These results showed that text classification accuracy using the SSE representation rivals the standard BoN baselines. From a limited set of experiments that were reported in Section 4.6.1, we concluded that BoN with trigrams performs statistically significantly better than BoN with bigrams for binary sentiment classification on the Amazon-v2 dataset. Further evaluations of the BoN baseline will be performed in our future work – e.g., the effect of using different feature selection heuristics with longer n-grams on the multi-class sentiment classification performance.

The properties of feature extraction in the SSE representation for text were further investigated in Chapter 4. The ability of SSE to encode phrase semantics such as the pseudo-subjectivity of n-grams was illustrated in that chapter. In addition, a weighted extension to SSE was proposed that captured global structure of text documents in the representation domain. In our future work, we intent to evaluate these SSE models on additional text labeling tasks. For instance, the multi-class multi-label setup of the RCV1 dataset [3]. A text document can be seen as a 1D sequence of symbolic features (i.e., unigrams). Hence, the SSE model can be applied to other sequence labeling tasks such as protein classification. In addition, the same SSE model can be applied to encode spatial distribution of sparse features in higher-dimensional sequences. This hypothesis

was evaluated in Chapter 5 where SSE was used to capture spatial configurations of image features, where each image was first transformed into a 2D sequence of sparse features through the quantization of low-level image descriptors. In this, the SSE-based image representation is akin to the spatial pyramid matching (SPM) [63] that encodes spatial distribution of image features in an unsupervised fashion. In the large-scale image classification experiments presented in Chapter 5 the SSE model outperformed the SPM baseline. In addition, it was shown that image classification with the SSE-based representation can rival the performance of the state of the art systems that use the same image quantization method.

In general, the SSE layer implements a sparse variant of a convolutional operator that projects sparse multi-dimensional arrays into a low-dimensional latent space. A text document corresponds to a 1D sequence of sparse features, while images can be transformed into 2D sequences of quantized features. The same SSE model can be applied to encode spatial configurations of features in higher-dimensional sequences. For example, a video can be transformed into a 3D sparse sequence using a quantization procedure for image descriptors. Then, the SSE layer can be used to encode spatial as well as temporal distribution of the quantized features in videos. In addition, the SSE-based representation for images that was considered in Chapter 5, partitioned an image at a single scale to obtain 2D sequence of quantized features for this image. As such, the scale of the partition can be varied in order to transform each image into a 3D sequence. In this case, SSE will encode spatial configuration of image features at multiple scales. This setup may enable the SSE model to better encode visual information in the representation domain, which in turn could improve image classification accuracy. We intend to test this hypothesis in our future work.

Another prominent direction of our future work is the development of an auto-encoder for SSE. The SSE model was proposed to tackle large-scale document labeling tasks. Large size of the parameter space in SSE may hinder its applications to small- or medium-scale datasets. These datasets contain significantly fewer samples that can be used for supervised training of the embedding weights in the SSE model. An appropriately defined auto-encoder may be used to initialize the SSE parameters in an unsupervised fashion using unlabeled documents. The initialized embedding parameters can then be fine-tuned with a few labeled samples that are available. Below, we will present one possible formulation of an auto-encoder for SSE. This auto-encoder allows one to initialize the phrase-level embedding parameters in the SSE layer (step

1 in Figure 4.2). However, the current formulation does not account for the document-level embedding layer (step 3 in Figure 4.2). As such, the proposed auto-encoder constitutes only our preliminary attempt at developing a procedure for an unsupervised pre-training of the parameters in SSE, biased for the document-level embedding (see steps 1–3 in Figure 4.2).

Auto-encoder for the Supervised Sequence Embedding Supervised learning that minimizes a loss function using stochastic gradient descent only guarantees to find a locally optimal solution. Consequently, a proper initialization of weights in the SSE model may significantly affect its classification performance. A common practice that is used to initialize weights in deep neural network (NN) models is the formulation of an appropriate auto-encoder. In general, an auto-encoder is defined as an inverse projection operator for the activation function at every layer of a neural network. The parameters of each activation function as well as its inverse operator are then estimated using an unsupervised pre-training that minimizes the error between unlabeled samples and their reconstructions. The use of auto-encoders in deep architectures can be beneficial when dealing with small datasets that contain only a few labeled samples. However, even for large-scale datasets a proper initialization of the NN parameters may result in a more stable solution to the optimization for the supervised learning, that better generalizes to the unseen samples. This section will present one possible formulation of an auto-encoder for the SSE model.

The SSE layer that was defined in (3.2), projects all sliding windows of n words into an M-dimensional space. This projection can be seen as a discrete 1D convolution applied to sparse vectors that encode words from a document $\mathbf{x} = (w_1, \ldots, w_N)$. Let $\mathbf{E}_\mathbf{x}$ denote the concatenation of word selector vectors for words in $\mathbf{x}$, so we have:

$$\mathbf{E}_\mathbf{x} = [\mathbf{e}_{w_1}, \ldots, \mathbf{e}_{w_N}],$$

where $\mathbf{E}_\mathbf{x} \in \mathbb{R}^{|\mathcal{D}| \times N}$, and each $\mathbf{e}_{w_i} \in \mathbb{R}^{|\mathcal{D}|}$ is the word selector vector, defined in (2.1), that contains a single non-zero entry at w_i-th index. Then the SSE projection in (3.2) can be re-written using 1D discrete convolution:

$$\varphi_{\text{SSE}}(\mathbf{x}) = \tanh\left(\mathbf{G} * \mathbf{E}_\mathbf{x}\right),$$

where $*$ denotes 1D discrete convolution operator with window size n, $\varphi_{\text{SSE}}(\mathbf{x}) \in \mathbb{R}^{M \times N}$, and $\tanh(\cdot)$ is

Table 6.1: Average classification accuracy for the movie review dataset.

Method	Classification Accuracy
Voting with two lexica [86]	63.1
Rule-based reversal on trees [86]	62.9
BoW with reversal [86]	76.4
Tree-CRF [201]	77.3
RAE (random init.) [86]	76.8
RAE (auto-encoder) [86]	77.7
SVM BoW-1g	76.17
SVM BoW-2g	**78.10**
SVM BoW-3g	77.57
SVM BoW-5g	76.84
SSE (random init.)	75.77
SSE (auto-encoder)	76.55

the hyperbolic tangent. In addition, let $\rho_{\text{SSE}}(\mathbf{x})$ denote an operator that returns unigram embedding vectors from $\mathbf{G}$ that make up each latent n-gram from $\mathbf{x}$. In other words, $\rho_{\text{SSE}}(\mathbf{x}) \in \mathbb{R}^{n \cdot M \times N}$ is a matrix whose j-th column contains the embedding vectors from $\mathbf{G}$ for n unigrams that form the n-gram γ_j.

An auto-encoder for SSE, denoted with $\varphi_{\text{inv}}(\mathbf{x})$, can now be defined:

$$\varphi_{\text{inv}}(\mathbf{x}) = \widehat{\mathbf{G}} * \varphi_{\text{SSE}}(\mathbf{x}),$$

where $*$ is a 1D discrete convolutional operator with window size of $\widehat{n}$, and $\widehat{\mathbf{G}} \in \mathbb{R}^{n \cdot M \times \widehat{n} \cdot M}$. The unsupervised pre-training of the parameters in $\varphi_{\text{SSE}}(\cdot)$ and $\varphi_{\text{inv}}(\cdot)$ minimizes the reconstruction error for the unigram embedding vectors. The reconstruction loss for one sample $\mathbf{x}$ is defined as

$$\mathcal{L}_{\text{SSE-ENC}}(\mathbf{x}_i) = \frac{1}{N} \cdot \|\varphi_{\text{inv}}(\mathbf{x}) - \rho_{\text{inv}}(\mathbf{x})\|_2^2. \tag{6.1}$$

The aforementioned formulation of the auto-encoder for SSE was evaluated using movie review dataset [202] for binary sentiment classification. The dataset contains short movie reviews with 5331 positive and 5331 negative samples. The auto-encoder for SSE was pre-trained using $n = 5$ and $\widehat{n} = 3$. Table 6.1 provides classification results on the movie review dataset for SSE, as well as several previously reported results for this dataset. In order to match the experimental setup from prior art, 10-fold cross validation was performed

and the average classification accuracy is reported. In order to obtain the validating sets, we removed 250 positive and 250 negative samples from the training sets. These validating sets were used to select the best-performing SSE model, and to set the penalty parameter C for our SVM baseline. We did not perform feature selection for the BoN baseline representation and used all unique phrases from the training sets. Surprisingly our BoN baseline outperformed all prior-art methods, such as the BoW model with bigrams and populated with hand-crafted features [86]. We attribute the improvement in the classification for our BoN baseline to the grid search that set the penalty parameter C.

The use of the auto-encoder to initialize the SSE weights improved the classification accuracy from 75.77% to 76.55%. The change in classification accuracy is comparable to the results reported in [86] that also initialized their model with an auto-encoder (see RAE numbers in Table 6.1). However, the SSE-based method achieved lower accuracy than other competing systems. From one perspective, this suggests that the SSE model may require a large number of supervised samples in order to achieve document labeling performance that rivals BoN. From a different perspective, the parameters in SSE are tuned for the phrase-level embedding, while text labeling is carried out at the document level. Consequently, the performance of the SSE-based document labeling may benefit from a formulation of an auto-encoder that takes into account the three-layer neural network for document-level embedding (see steps 1–3 in Figure 4.2). This issue will be addressed in our future work.

Bibliography

[1] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *In ACL*, pages 187–205, 2007.

[2] Hongning Wang, Yue Lu, and Chengxiang Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 783–792, New York, NY, USA, 2010. ACM.

[3] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, December 2004.

[4] Koray Kavukcuoglu. Hierarchical models for object recognition. `http://koray.kavukcuoglu.org/research.html`, accessed (Aug 2012).

[5] Committee on the Fundamentals of Computer Science: Challenges and National Research Council Opportunities. *Computer Science: Reflections on the Field, Reflections from the Field*. The National Academies Press, 2004.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[7] Tony Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[8] Jan Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[9] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, November 1998.

[10] S. Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *Information Theory, IEEE Transactions on*, 18(1):14 – 20, jan 1972.

[11] R. Blahut. Computation of channel capacity and rate-distortion functions. *Information Theory, IEEE Transactions on*, 18(4):460 – 473, jul 1972.

[12] P.O. Vontobel, A. Kavcic, D.M. Arnold, and H.-A. Loeliger. A generalization of the blahut-arimoto algorithm to finite-state channels. *Information Theory, IEEE Transactions on*, 54(5):1887 –1918, may 2008.

[13] Ankit Gupta and Sergio Verdú. Nonlinear sparse-graph codes for lossy compression. *IEEE Trans. Inf. Theor.*, 55(5):1961–1975, May 2009.

[14] A. Gupta, S. Verdu, and T. Weissman. Rate-distortion in near-linear time. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 847 –851, july 2008.

[15] Kai Wang and J.W. Woods. Mpeg motion picture coding with long-term constraint on distortion variation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(3):294 –304, march 2008.

[16] N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, October 1993.

[17] G.J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *Signal Processing Magazine, IEEE*, 15(6):74 –90, nov 1998.

[18] George Erne, Markus; Moschytz. Audio coding based on rate-distortion and perceptual optimization techniques. In *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*, 8 1999.

[19] David L Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6):797–829, 2006.

[20] David L. Donoho and Xiaoming Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 2001.

[21] S.G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397–3415, December 1993.

[22] Y C Pati, R Rezaiifar, and P S Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *Proceedings of 27th Asilomar Conference on Signals Systems and Computers*, 1:40–44, 1993.

[23] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43(1):129–159, January 2001.

[24] Iddo Drori and David L. Donoho. Solution of l1 minimization problems by lars/homotopy methods. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2006.

[25] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996.

[26] Michael S. Lewicki. Efficient coding of natural sounds. *Nature Neuroscience*, 5:356–363, 2002.

[27] B. Olshausen and D. Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14(4): 481–487, August 2004.

[28] Yuanqing Li, Andrzej Cichocki, and Shun-ichi Amari. Analysis of sparse representation and blind source separation. *Neural Comput.*, 16(6):1193–1234, June 2004.

[29] J.-L. Starck, M. Elad, and D.L. Donoho. Image decomposition via the combination of sparse representations and a variational approach. *Image Processing, IEEE Transactions on*, 14(10):1570 –1582, oct. 2005.

[30] Michael Elad and Michal Aharon. Image denoising via learned dictionaries and sparse representation. *Computer Engineering*, 43(7):895–900, 2006.

[31] Michal Aharon, Michael Elad, and Alfred Bruckstein. K -svd : An algorithm for designing overcomplete dictionaries for sparse representation. *Structure*, 54(11):4311–4322, 2006.

[32] T. Takahashi, K. Konishi, and T. Furukawa. Reweighted l2 norm minimization approach to image inpainting based on rank minimization. In *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, pages 1 –4, aug. 2011.

[33] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1994.

[34] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.*, 52(3):471–501, August 2010.

[35] W.B. Cavnar and J.M. Trenkle. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175, 1994.

[36] Jun Yan, Ning Liu, Benyu Zhang, Shuicheng Yan, Zheng Chen, Qiansheng Cheng, Weiguo Fan, and Wei-Ying Ma. Ocfs: optimal orthogonal centroid feature selection for text categorization. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 122–129, New York, NY, USA, 2005. ACM.

[37] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[38] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[39] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, March 1990.

[40] Hongfang Jing, Bin Wang, Yahui Yang, and Yan Xu. A general framework of feature selection for text categorization. In *Proceedings of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition*, MLDM '09, pages 647–662, Berlin, Heidelberg, 2009. Springer-Verlag.

[41] Duc-Son Pham and S. Venkatesh. Joint learning and dictionary construction for pattern recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[42] Qiang Zhang and Baoxin Li. Discriminative k-svd for dictionary learning in face recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2691 –2698, june 2010.

[43] Julien Mairal, Francis Bach, and Jean Ponce. Task-driven dictionary learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):791–804, 2012.

[44] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Supervised dictionary learning. *CoRR*, abs/0809.3083, 2008.

[45] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of The American Society for Information Science*, 41(6):391–407, 1990.

[46] David M. Blei and Jon D. Mcauliffe. Supervised topic models. In *Neural Information Processing Systems*. MIT Press, 2007.

[47] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM Press New York, NY, USA, 1999.

[48] Mark Girolami and Ata Kabán. On an equivalence between plsi and lda. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 433–434, New York, NY, USA, 2003. ACM.

[49] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[50] Peter V. Gehler, Alex D. Holub, and Max Welling. The rate adapting poisson model for information retrieval and object recognition. In *In Proceedings of 23rd International Conference on Machine Learning (ICML06*, page 2006. ACM Press, 2006.

[51] Krzysztof C. Kiwiel. Convergence and efficiency of subgradient methods for quasiconvex minimization. *Mathematical Programming*, 90:1–25, 2001.

[52] Léon Bottou. Stochastic learning. In Olivier Bousquet and Ulrike von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004.

[53] Yoshua Bengio. *Learning Deep Architectures for AI*. Now Publishers Inc., Hanover, MA, USA, 2009.

[54] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, Bellevue, USAl, June 2011. Omnipress.

[55] Richard Socher, Cliff Chiung-Yu Lin, Andrew Ng, and Chris Manning. Parsing natural scenes and natural language with recursive neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 129–136, New York, NY, USA, June 2011. ACM.

[56] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Departement D'informatique Et Recherche Operationnelle. A neural probabilistic language model. *Journal of Machine Learning Research*, 3: 1137–1155, 2000.

[57] Frederic Morin. Hierarchical probabilistic neural network language model. aistats'05. In *In AISTATS*, pages 246–252, 2005.

[58] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[59] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[60] Alex Berg, Jia Deng, Sanjeev Satheesh, Hao Su, and Fei-Fei Li. Image-net large scale visual recognition challenge 2011 (ilsvrc). `www.image-net.org/challenges/LSVRC/2011`.

[61] Christiane Fellbaum, editor. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May 1998.

[62] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, T. Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360 –3367, june 2010.

[63] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006*, volume 2, pages 2169–2178. IEEE, 2006.

[64] Wai Lam and Chao Yang Ho. Using a generalized instance set for automatic text categorization. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 81–89, New York, NY, USA, 1998. ACM.

[65] Brij Masand, Gordon Linoff, and David Waltz. Classifying news stories using memory based reasoning. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '92, pages 59–65, New York, NY, USA, 1992. ACM.

[66] Piotr Mirowski, Marc'Aurelio Ranzato, and Yann LeCun. Dynamic auto-encoders for semantic indexing. In *Proceedings of the NIPS 2010 Workshop on Deep Learning*, 2010.

[67] Georgios Paltoglou and Mike Thelwall. A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1386–1395, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[68] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48, 1998.

[69] Kamal Nigam. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

[70] Kwan Yi and Jamshid Beheshti. A hidden markov model-based text classification of medical documents. *J. Inf. Sci.*, 35:67–81, February 2009.

[71] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34:1–47, March 2002.

[72] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[73] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*, 2008.

[74] David Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In Claire Ndellec and Cline Rouveirol, editors, *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, pages 4–15. Springer Berlin / Heidelberg, 1998.

[75] Hugo Zaragoza, Nick Craswell, Michael Taylor, Suchi Saria, and Stephen Robertson. Microsoft cambridge at trec-13: Web and hard tracks. In *PROCEEDINGS OF TREC 2004*, 2004.

[76] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389, April 2009.

[77] Christina S. Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch string kernels for SVM protein classification. In *NIPS*, pages 1417–1424, 2002.

[78] Jason Weston, Christina Leslie, Eugene Ie, Dengyong Zhou, Andre Elisseeff, and William Stafford Noble. Semi-supervised protein classification using cluster kernels. *Bioinformatics*, 21(15):3241–3247, 2005.

[79] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, March 2002.

[80] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

[81] M.A. Ranzato, Fu Jie Huang, Y.-L. Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1 –8, june 2007.

[82] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[83] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

[84] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[85] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50:969–978, July 2009.

[86] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[87] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Advances in neural information processing systems 2. chapter Handwritten digit recognition with a back-propagation network, pages 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[88] Yann LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253 –256, 30 2010-june 2 2010.

[89] Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michal Mathieu, and Yann LeCun. Learning convolutional feature hierarchies for visual recognition. In *NIPS'10*, pages 1090–1098, 2010.

[90] E. B. Goldstein. *Sensation and Perception*. Wadsworth Thomson, sixth edition edition, 2002.

[91] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, December 1989.

[92] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 958 – 963, aug. 2003.

[93] Kumar Chellapilla, Sidd Puri, and Patrice Simard. High Performance Convolutional Neural Networks for Document Processing. In Guy Lorette, editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France), October 2006. Université de Rennes 1, Suvisoft.

[94] Ahmad AbdulKader. A two-tier arabic offline handwriting recognition based on conditional joining rules. In *Proceedings of the 2006 conference on Arabic and Chinese handwriting recognition*, SACH'06, pages 70–81, Berlin, Heidelberg, 2008. Springer-Verlag.

[95] Kumar Chellapilla and Patrice Simard. A New Radical Based Approach to Offline Handwritten East-Asian Character Recognition. In Guy Lorette, editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France), October 2006. Université de Rennes 1, Suvisoft.

[96] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *J. Field Robot.*, 26(2): 120–144, February 2009.

[97] Fabian Nasse, Christian Thurau, and Gernot A. Fink. Face detection using gpu-based convolutional neural networks. In *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns*, CAIP '09, pages 83–90, Berlin, Heidelberg, 2009. Springer-Verlag.

[98] M. Delakis and C. Garcia. Text detection with convolutional neural networks. In *International Conference on Computer Vision Theory and Applications*, 2008.

[99] M. E. Maron. Automatic indexing: An experimental inquiry. *J. ACM*, 8(3):404–417, July 1961.

[100] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. Some effective techniques for naive bayes text classification. *Knowledge and Data Engineering, IEEE Transactions on*, 18(11):1457 –1466, nov. 2006.

[101] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, CIKM '98, pages 148–155, New York, NY, USA, 1998. ACM.

[102] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Ndellec and Cline Rouveirol, editors, *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer Berlin / Heidelberg, 1998.

[103] Christina Sauper, Aria Haghighi, and Regina Barzilay. Incorporating content structure into text analysis applications. In *EMNLP'10*, pages 377–387, 2010.

[104] Radford M. Neal and Geoffrey E. Hinton. Learning in graphical models. chapter A view of the EM algorithm that justifies incremental, sparse, and other variants, pages 355–368. MIT Press, Cambridge, MA, USA, 1999.

[105] O. Yakhnenko, A. Silvescu, and V. Honavar. Discriminatively trained markov model for sequence classification. In *Data Mining, Fifth IEEE International Conference on*, page 8 pp., nov. 2005.

[106] M.K. Kozlov, S.P. Tarasov, and L.G. Khachiyan. The polynomial solvability of convex quadratic programming. *USSR Computational Mathematics and Mathematical Physics*, 20(5):223 – 228, 1980.

[107] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[108] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415 –425, mar 2002.

[109] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, March 2002.

[110] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors, *Optimization for Machine Learning*, pages 351–368. MIT Press, 2011.

[111] Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale l2-loss linear support vector machines. *J. Mach. Learn. Res.*, 9:1369–1398, June 2008.

[112] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 408–415, New York, NY, USA, 2008. ACM.

[113] Y. H. Li and A. K. Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

[114] Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

[115] Paolo Frasconi, Giovanni Soda, and Alessandro Vullo. Text categorization for multi-page documents: a hybrid naive bayes hmm approach. In *Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, JCDL '01, pages 11–20, New York, NY, USA, 2001. ACM.

[116] T.K. Moon. The expectation-maximization algorithm. *Signal Processing Magazine, IEEE*, 13(6):47 –60, nov 1996.

[117] Ludovic Denoyer, Patrick Gallinari, and Hugo Zaragoza. Hmm-based passage models for document classification and ranking. In *In ECIR01*, pages 126–135, 2001.

[118] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

[119] M. Negnevitsky and M. Ringrose. Accelerated learning in multi-layer neural networks. In *Neural Information Processing, 1999. Proceedings. ICONIP '99. 6th International Conference on*, volume 3, pages 1167 –1171 vol.3, 1999.

[120] D. N. Lawley. A generalization of fisher's z test. *Biometrika*, 30(1/2):pp. 180–187, 1938.

[121] Graham Neubig, Yuichiroh Matsubayashi, Masato Hagiwara, and Koji Murakami. Safety information mining – what can NLP do in a disaster. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 965–973, Chiang Mai, Thailand, 11 2011.

[122] James Pustejovsky. The generative lexicon. *Comput. Linguist.*, 17(4):409–441, December 1991.

[123] Wlodek Zadrozny. On compositional semantics. In *Proceedings of the 14th conference on Computational linguistics - Volume 1*, COLING '92, pages 260–266, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[124] Zoltn Gendler Szab. Compositionality. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2008 edition, 2008.

[125] Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 415–463. Springer, 2012.

[126] Bing Liu. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012.

[127] Bing Liu. Sentiment analysis and subjectivity. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group, Boca Raton, FL, 2010.

[128] Eric Breck, Yejin Choi, and Claire Cardie. Identifying expressions of opinion in context. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2683–2688, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[129] Alistair Kennedy and Diana Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22:2006, 2006.

[130] Xiaochuan Ni, Gui-Rong Xue, Xiao Ling, Yong Yu, and Qiang Yang. Exploring in the weblog space by detecting informative and affective articles. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 281–290, New York, NY, USA, 2007. ACM.

[131] Ann Devitt and Khurshid Ahmad. Sentiment polarity identification in financial news: A cohesion-based approach. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

[132] Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of EMNLP*, pages 327–335, 2006.

[133] Vasileios Hatzivassiloglou and Janyce M. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th conference on Computational linguistics - Volume 1*, COLING '00, pages 299–305, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[134] Ellen Riloff, Siddharth Patwardhan, and Janyce Wiebe. Feature subsumption for opinion analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 440–448, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[135] Janyce Wiebe and Rada Mihalcea. Word sense and subjectivity. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 1065–1072, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[136] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[137] Daisuke Ikeda, Hiroya Takamura, Lev arie Ratinov, and Manabu Okumura. Learning to shift the polarity of words for sentiment classification. In *In Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP*, 2008.

[138] Benjamin Snyder and Regina Barzilay. Multiple aspect ranking using the good grief algorithm. In *In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL*, pages 300–307, 2007.

[139] Fangtao Li, Minlie Huang, and Xiaoyan Zhu. Sentiment analysis with global topics and local dependency. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[140] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 375–384, New York, NY, USA, 2009. ACM.

[141] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[142] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 342–351, New York, NY, USA, 2005. ACM.

[143] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[144] Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 339–346, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[145] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Expanding domain sentiment lexicon through double propagation. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 1199–1204, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[146] Murthy Ganapathibhotla and Bing Liu. Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 241–248, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[147] Nitin Jindal and Bing Liu. Identifying comparative sentences in text documents. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 244–251, New York, NY, USA, 2006. ACM.

[148] Nitin Jindal and Bing Liu. Mining comparative sentences and relations. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, pages 1331–1336. AAAI Press, 2006.

[149] Koji Eguchi and Victor Lavrenko. Sentiment retrieval using generative models. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 345–354, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[150] Michael Gamon. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[151] Tetsuya Nasukawa and Jeonghee Yi. Sentiment analysis: capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, K-CAP '03, pages 70–77, New York, NY, USA, 2003. ACM.

[152] Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. Learning to identify review spam. In Toby Walsh, editor, *IJCAI*, pages 2488–2493. IJCAI/AAAI, 2011.

[153] Nitin Jindal and Bing Liu. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1189–1190, New York, NY, USA, 2007. ACM.

[154] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the international conference on Web search and web data mining*, WSDM '08, pages 219–230, New York, NY, USA, 2008. ACM.

[155] Justin Martineau and Tim Finin. Delta tfidf: An improved feature space for sentiment analysis. In Eytan Adar, Matthew Hurst, Tim Finin, Natalie S. Glance, Nicolas Nicolov, and Belle L. Tseng, editors, *ICWSM*. The AAAI Press, 2009.

[156] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *ACL*, 2011.

[157] Dmitriy Bespalov, Bing Bai, Yanjun Qi, and Ali Shokoufandeh. Sentiment classification based on supervised latent n-gram analysis. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 375–382, New York, NY, USA, 2011. ACM.

[158] George E. Dahl, Ryan Prescott Adams, and Hugo Larochelle. Training restricted boltzmann machines on word observations. *CoRR*, abs/1202.5695, 2012.

[159] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2012.

[160] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[161] Elad Hazan, Tomer Koren, and Nati Srebro. Beating sgd: Learning svms in sublinear time. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NIPS*, pages 1233–1241, 2011.

[162] Sida Wang and Christopher Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[163] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, pages 2764–2770, 2011.

[164] Gao Yang. Matlab gabor filter implementation for image processing toolbox. `http://www.mathworks.com/matlabcentral/fileexchange/23253-gabor-filter`, accessed (Aug 2012).

[165] Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. Fast inference in sparse coding algorithms with applications to object recognition. *CoRR*, abs/1010.3467, 2010.

[166] Koray Kavukcuoglu. Learning feature extractors using sparse coding. `http://koray.kavukcuoglu.org/research.html`, accessed (Aug 2012).

[167] Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In *NIPS'07*, 2007.

[168] Marc'Aurelio Ranzato. Learning deep hierarchies of features. `http://www.cs.nyu.edu/~ranzato/research/figs/sesm-filters.png`, accessed (Aug 2012).

[169] John G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J. Opt. Soc. Am. A*, 2(7):1160–1169, Jul 1985.

[170] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[171] Guy Lebanon, Yi Mao, and Joshua Dillon. The locally weighted bag of words framework for document representation. *J. Mach. Learn. Res.*, 8:2405–2441, December 2007.

[172] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2012.

[173] Jianchao Yang, Kai Yu, Yihong Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794 –1801, june 2009.

[174] Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, T. Cour, Kai Yu, Liangliang Cao, and T. Huang. Large-scale image classification: Fast feature extraction and svm training. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1689 –1696, june 2011.

[175] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[176] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417, 2006.

[177] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[178] Koen van de Sande, Theo Gevers, and Cees Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1582–1596, September 2010.

[179] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cdric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[180] Florent Perronnin, Jorge Snchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *IN: ECCV*, 2010.

[181] Florent Perronnin, Christopher Dance, Gabriela Csurka, and Marco Bressan. Adapted vocabularies for generic visual categorization. In *In ECCV*, pages 464–475, 2006.

[182] J. Farquhar, S. Szedmak, H. Meng, and J. Shawe-Taylor. Improving bag-of-keypoints image categorisation. 2005.

[183] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 604 – 610 Vol. 1, oct. 2005.

[184] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 551–556, New York, NY, USA, 2004. ACM.

[185] Yao-Jen Chang and Tsuhan Chen. Semi-supervised learning with kernel locality-constrained linear coding. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 2977 –2980, sept. 2011.

[186] Florent Perronnin and Christopher R. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR'07*, pages –1–1, 2007.

[187] Xi Zhou, Kai Yu, Tong Zhang, and Thomas S. Huang. Image classification using super-vector coding of local image descriptors. In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, pages 141–154, Berlin, Heidelberg, 2010. Springer-Verlag.

[188] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2223–2231. 2009.

[189] Andrea Vedaldi Ken Chatfield, Victor Lempitsky and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *Proceedings of the British Machine Vision Conference*, pages 76.1–76.12. BMVA Press, 2011.

[190] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 994 – 1000 vol. 2, june 2005.

[191] J. Sanchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1665 –1672, june 2011.

[192] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Spatial pyramid matching. In B. Schiele S. Dickinson, A. Leonardis and M. Tarr, editors, *Object Categorization: Computer and Human Vision Perspectives*, chapter 21, pages 401–415. Cambridge University Press, 2009.

[193] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Comput. Vis. Image Underst.*, 106(1):59–70, April 2007.

[194] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[195] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[196] Quoc V. Le, Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, and Andrew Y. Ng. On optimization methods for deep learning. In *ICML'11*, pages 265–272, 2011.

[197] Dan C. Cireşan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1237–1242. AAAI Press, 2011.

[198] C. Farabet, C. Poulet, J.Y. Han, and Y. LeCun. Cnp: An fpga-based processor for convolutional networks. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pages 32 –37, 31 2009-sept. 2 2009.

[199] Florent Perronnin, Zeynep Akata, Zaid Harchaoui, and Cordelia Schmid. Towards Good Practice in Large-Scale Learning for Image Classification. In *Computer Vision and Pattern Recognition*, Providence, Rhode Island, United States, June 2012.

[200] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[201] Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 786–794, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[202] Bo Pang and Lillian Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.