Language Pyramid and Multi-Scale Text Analysis

Shuang Hong Yang
College of Computing
Georgia Institute of Technology
266 Ferst Dr., Atlanta, GA 30318
shy@gatech.edu

Hongyuan Zha
College of Computing
Georgia Institute of Technology
266 Ferst Dr., Atlanta, GA 30318
zha@cc.gatech.edu

ABSTRACT

The classical *Bag-of-Word* (BOW) model represents a document as a histogram of word occurrence, losing the spatial information that is invaluable for many text analysis tasks. In this paper, we present the *Language Pyramid* (LaP) model, which casts a document as a probabilistic distribution over the joint semantic-spatial space and motivates a multi-scale 2D local smoothing framework for nonparametric text coding. LaP efficiently encodes both semantic and spatial contents of a document into a pyramid of matrices that are smoothed both semantically and spatially at a sequence of resolutions, providing a convenient multi-scale imagic view for natural language understanding. The LaP representation can be used in text analysis in a variety of ways, among which we investigate two instantiations in the current paper: (1) multi-scale text kernels for document categorization, and (2) multi-scale language models for ad hoc text retrieval. Experimental results illustrate that: for classification, LaP outperforms BOW by (up to) 4% on moderate-length texts (RCV1 text benchmark) and 15% on short texts (Yahoo! queries); and for retrieval, LaP gains 12% MAP improvement over uni-gram language models on the OHSUMED data set.

Categories and Subject Descriptors

I.5.4 [**Pattern Recognition**]: Applications—*Text processing*; H.3.3 [**Information Search and Retrieval**]: Retrieval Models; I.2.4 [**Artificial Intelligence**]: Learning

General Terms

Theory, Algorithms, Performance

Keywords

Text Spatial Contents Modeling, Multi-Scale Text Analysis, Bag of Word, Language Pyramid, Multi-Scale Language Models, Multi-Scale Text Kernel

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

1. INTRODUCTION

Today's text analysis systems (e.g., search engines, text miners, social medias) rely almost unanimously on the *Bag-of-Word* (BOW) model for text representation, which views a document as a loose collection of independent words and represents it as a histogram of word occurrences. Although simple, efficient, sometimes effective, and usually convenient for popular vector-space analyzing algorithms, the BOW model loses too much information during its balls-into-bins process – with the same set of word occurrences, one could compose tens of documents with diversified meanings. Particularly, the spatial information (e.g., word correlation, proximity, ordering and long distance dependence) is totally lost by BOW, which, however, has been proved invaluable for many text analysis tasks [21, 10, 23, 15], especially for short text (e.g., web search queries, online ads, Twitter mini-posts) processing [11, 28, 3, 37, 29].

Substantial efforts have been made to develop techniques to (partially) encode spatial information in text representations. Perhaps the first attempt is the n-gram BOW model [21], where every n adjacent words are bound together for occurrence counting. This model, however, has been plagued by the curse of dimensionality [8] since the exponential number of word combinations could easily exceed any possibly available corpus size, not to mention the spatial correlations among n-grams are still missing. In information retrieval, various techniques have been developed to segment short texts (e.g., web search queries) into segments so as to enforce word proximity and ordering constraints [11, 28]. Nonetheless, to get the right segments remains a challenge – any segmentation system could occasionally either split segments or miss segments or both; yet, the spatial correlations among segments are still at large. Similar problems exist for text representation methods based on: noun phrase (or named entity) extraction [7, 29], hidden Markov models [24], mixture of n-grams [10], subsequence string kernel [19], etc.

In this paper, we present the *Language Pyramid* (or LaP) model, a new text representation method. Without sacrifice of any spatial information, LaP characterizes a document as a binary 2D matrix (i.e., a black-and-white image), with semantic axis along rows and spatial axis along columns. This 2D code inspires a new perspective of text representations as probabilistic distributions over the joint semantic-spatial space, and motivates 2D nonparametric density estimation using locally weighted smooth kernels, which, in the spatial-axis, coincides the spatial filter transforms in image processing [4] and, in the semantic direction, resembles the smoothing techniques used in language models [35]. We formulate

both the semantic and spatial smoothing as graph-based optimizations and efficiently solve them by simply convolving the binary text codes with well-defined local kernels. To enhance scale robustness of this 2D local smoothing framework, we borrow the idea of image pyramids [4] to iteratively apply the 2D smoothers at a sequence of scales, resulting in (for each document) a pyramid of matrices that are smoothed both semantically and spatially at progressive resolutions. This multiple-scale representation includes both the original document (at the finest scale) and its BOW representation (at the coarsest scale) as spacial cases.

We believe the LaP model creates a new dimension for text modeling:

- Firstly, it efficiently encodes both semantic and spatial contents of documents, making possible more accurate modeling of texts, especially short texts such as web search queries, social community questions, online ads, Twitter mini-posts and so on;
- Secondly, it provides multi-scale representations for texts, allowing natural language to be analyzed in a principled scale-invariant fashion;
- Thirdly, it provides an imagic view for document, dramatically facilitating natural language understanding and text visualization;
- Last but not least, it enables well-established image analysis tools to be applied to text processing, e.g., matching, segmentation, description, interests points detection, classification.

The LaP representation could be used in text analysis tasks in a variety of ways. In this paper, we demonstrate two instantiations of LaP applications:

1. *Multi-Scale Text Kernels.* In the scenario of text classification, we show that the matrices in LaP can be combined using *multiple kernel learning* (MKL), leading to *multi-scale text kernels* (MSK) that allow kernel-based learning machines to be applied on.
2. *Multi-Scale Language Models.* The LaP representation defines multi-scale, spatially sensitive, 2D nonparametric language models that could be exploited for ad hoc text retrieval. We show that based on LaP, queries and documents can be matched either globally or locally at multiple resolutions, opening up an elegant way for multi-scale ad hoc text retrieval.

Extensive experiments were conducted to test LaP against BOW counterparts, on RCV1 benchmark as well as Yahoo! query corpus for classification evaluation, and on OHSUMED data set for retrieval evaluation. Experimental results indicate that LaP consistently outperforms BOW in all the three tasks. Particularly, for classification, it improves the classification accuracy by (up to) 15% on short texts (Yahoo! queries) and 4% on moderate length texts (RCV1); and for retrieval, it achieves up to 12% MAP improvement over the uni-gram language models.

The remaining sections of this paper is organized as follows. We first present the 2D local smoothing framework in Section 2, and extend it in Section 3 to the multi-scale version, i.e., the LaP model. Section 4 instantiates two applications of LaP, i.e., multi-scale kernels for text classification and multi-scale language models for ad hoc retrieval. Section 5 then conducts extensive evaluations for the proposed models. Finally, related topics are briefly discussed in Section 6, and conclusions are addressed in Section 7.

2. 2D LOCAL SMOOTHING

Let $\mathcal{W}$ be a document comprised of a finite sequence of words: $\{\mathcal{W} = w_1 w_2 \dots w_N,\ w_j \in \mathcal{V}\}$, where N is the document length and $\mathcal{V} = \{v_1, v_2, \dots, v_M\}$ the vocabulary of size M. Alternatively, without any information loss, we can characterize $\mathcal{W}$ as a 2D $M \times N$-sized binary matrix X, the (i,j)-th entry x_{ij} of which indicates whether or not the i-th word of $\mathcal{V}$ is observed at the j-th position of $\mathcal{W}$, i.e.: $x_{ij} = I(v_i, w_j)$, where $I(a,b) = 1$ if $a = b$ and 0 otherwise. Hereafter, we will refer to the i and j directions of X as the *semantic axis* (or v-axis) and the *spatial axis* (or w-axis) respectively. Accordingly, $\mathcal{W}$ and $\mathcal{V}$ are also referred to as *spatial space* and *semantic space* respectively. Under this 2D view, a document X is equivalent to a black-and-white image except that for images both i and j directions are spatial axes.

Usually, the document length is much smaller than the vocabulary size, $N \ll M$, making X too sparse for statistical analyzing. Therefore, the popular BOW model represents a document as a vector of word frequencies, which is equivalent to project the 2D matrix X to a 1D vector:

$$\mathbf{d}^{bow} = \sum\nolimits_{j=1}^{N} x_{ij} = X \times \mathbf{1}, \tag{1}$$

where $\mathbf{1}$ is an N-vector with all 1 entries. As the spatial axis is wiped out, the rich spatial information contained in X is totally lost by the BOW representation.

In this paper, we are motivated for text representation that encodes both semantic and spatial contents of a document. Generalizing the binary matrix representation, we have the following definition:

DEFINITION 1. *The 2D text model $X \in \mathbb{R}^{M \times N}$ is a probabilistic distribution over the joint semantic-spatial space*: $\mathcal{V} \times \mathcal{W} \to \mathbb{R}^+$, $0 \leqslant x_{ij} \leqslant 1$, $\sum_i \sum_j x_{ij} = 1$.

This 2D text model defines a generative model for texts. Intuitively, we can interpret x_{ij} as the probability of observing the semantic word v_i at the spatial position w_j. From this point of view, the binary matrix representation $\hat{X}$ (after normalization) can be understood as a nonparametric estimation of X. To see this, we have:

$$\hat{\mathbf{x}}_{\cdot j} = \frac{1}{N} \sum\nolimits_{n=1}^{N} \delta(w_j - w_n) \mathbf{I}_{\cdot j}, \tag{2}$$

$$\hat{\mathbf{x}}_{i\cdot} = \frac{1}{M} \sum\nolimits_{m=1}^{M} \delta(v_i - v_m) \mathbf{I}_{i\cdot}, \tag{3}$$

$$\hat{x}_{ij} = \frac{1}{MN} \sum\nolimits_{n=1}^{N} \sum\nolimits_{m=1}^{M} \delta(v_i - v_m) \delta(w_j - w_n). \tag{4}$$

where $\hat{\mathbf{x}}_{\cdot j}$ denotes the j-th column vector of $\hat{X}$, and $\hat{\mathbf{x}}_{i\cdot}$ the i-th row vector. Eqn.(2) is equivalent to a Parzen estimator in the (spatial) w-direction, that is, the density at position w_j is expressed as an average of kernel functions that are centered at each sample w_n. Here the Dirac kernels δ is used, i.e., only samples with zero distance [1] ($j = n$) contributes to the density at w_j. Similarly, Eqn.(3) and (4) correspond to Dirac kernel based Parzen estimators in (semantic) v-axis

[1]Since w, v are indices for ordered sets $\mathcal{W}$ and $\mathcal{V}$ respectively, $(w_i = w_j) \Leftrightarrow (i = j)$; similarly, $(v_i = v_j) \Leftrightarrow (i = j)$.

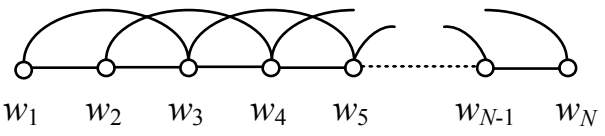

Figure 1: Chain graph used in spatial smoothing.

and 2D (w, v)-axes respectively. In practice, Dirac kernels lead to estimates that are too erratic and noisy. This motivates us to use more well-defined kernels for reliable text coding.

A fundamental characteristic of documents is that neighboring words are usually highly correlated, which is true both semantically [22] and spatially [15]. For instance, observing the word "New" indicates a high likelihood of seeing the other word "York" at the next position. This phenomenon suggests more reliable estimating of X by using locally weighted smooth kernels [4], which is, in effect, equivalent to smoothing $\hat{X}$ from binary to grayscale matrix. The following sections derive local smoothing for the spatial and the semantic dimensions respectively, both of which can be formulated as graph-based learning problems and reduced to simple convolutions with smooth kernels.

2.1 Spatial Smoothing

Spatial smoothing has long been popularized in signal processing and image analysis [4], and was quite recently explored in text modeling by [15]. Particularly for our task, spatial smoothing attempts to smooth the matrix X along the w-axis, based on the intuitions that: (i) the semantic distribution (i.e., distribution over the semantic space $\mathcal{V}$) should be similar between two neighboring positions, i.e., $\mathbf{x}_{\cdot i} \sim \mathbf{x}_{\cdot j}$ if $(i-j) \sim 0$; and (ii) the degree of similarity should be consistent with the distance of the locations. By doing this, we are actually borrowing the presence of a word at w_i to its neighboring location w_j with a discount depending on the distances $|i-j|$. Formally, this implies an undirected chain graph $\mathcal{G}_w$ (Figure 1), where two neighboring positions are connected with a weight depending on their distance. And spatial smoothing can be formulated as graph-based learning that seeks a balance between the fidelity to the original code X^0 and the consistency with the the graph $\mathcal{G}_w$:

$$\begin{aligned} \min \quad & (1-\lambda)\sum\nolimits_{i=1}^{N}(\mathbf{x}_{\cdot i}-\mathbf{x}^0_{\cdot i})^2 \\ & +\lambda\sum\nolimits_{i=1}^{N}\sum\nolimits_{j=1}^{N}\omega_{ij}(\mathbf{x}_{\cdot i}-\mathbf{x}_{\cdot j})^2, \end{aligned} \tag{5}$$

where the superscript '0' specifies the Dirichlet-smoothed version of the original binary code: $x^0_{ij} = (\frac{\eta}{MN}+\hat{x}_{ij})/(1+\eta)$, $\lambda \geqslant 0$ is the tradeoff parameter between the two objectives, and ω_{ij} the connection wight between w_i and w_j.

It is quite interesting that the solution to Eqn.(5) turns out to be as simple as the convolution of the original code with a specific kernel [2]:

$$\mathbf{x}_{\cdot i} = \sum\nolimits_t \mathbf{x}^0_{\cdot i-t}k_t \tag{6}$$

where k is a kernel function defined over a T-width window $t \in \{\lceil -\frac{T}{2} \rceil, \ldots, \lfloor \frac{T}{2} \rfloor\}$, the value of k depends on λ and ω's, and the summation is over the T-window. This simple result inspires us that, instead of carefully picking the right values for ω's and λ, we can directly work with the kernel k, e.g., by using well defined kernel functions such as the Gaussian kernel $k_t = \frac{1}{2\pi\sigma}\exp(t^2/2\sigma)$, and convolving it with the original binary coding to obtain the spatially smoothed codes. This is equivalent to applying a Gaussian spatial filter to the original code [4].

[2]Eqn.(6) could be arrived at by taking the first-order derivative of the objective in Eqn.(5) w.r.t. $\mathbf{x}$ and making it zero.

2.2 Semantic Smoothing

In contrast to the w-direction spatial smoothing, semantic smoothing attempts to smooth X along the semantic v-axis, ensuring that the distributions over the spatial space $\mathcal{W}$ are similar between two semantically related words, i.e., if $v_i \sim v_j$, then $\mathbf{x}_{i\cdot} \sim \mathbf{x}_{j\cdot}$. Intuitively, we are borrowing the presence of the word v_i to a relevant word v_j with a discount according to the degree of their relevance.

Semantic smoothing has been extensively explored in natural language processing, e.g., for tackling the sparseness of BOW [21] or the wellness of language models (LMs) [35]. Classical smoothing methods, e.g., Laplacian, Dirichlet, and Jelinek-Mercer smoother, usually shrink the original distributions with a predefined reference distribution, regardless of the correlations among words. In contrast, we consider locally weighted smoothing, similar to what we used in spatial smoothing, which is preferable because of their capability to capture word correlations [22].

Assume the knowledge on word correlations has been fully conveyed into an undirected graph $\mathcal{G}_v$, where two semantically correlated words v_i and v_j are connected with a weight μ_{ij} according to the degree of their dissimilarity d_{ij}. Semantic smoothing is formulated as the following graph-based learning task:

$$\begin{aligned} \min \quad & (1-\eta)\sum\nolimits_{i=1}^{M}\mu_i(\mathbf{x}_{i\cdot}-\mathbf{x}^0_{i\cdot})^2 \\ & +\eta\sum\nolimits_{i=1}^{M}\sum\nolimits_{j=1}^{M}\mu_{ij}(\mathbf{x}_{i\cdot}-\mathbf{x}_{j\cdot})^2, \end{aligned} \tag{7}$$

where $\eta \geqslant 0$ defines the tradeoff, μ_i weights the importance of the node v_i. Again, the solutions could be expressed as convolutions with locally weighted smooth kernels:

$$\mathbf{x}_{i\cdot} = \sum\nolimits_t \mathbf{x}^0_{(i-t)\cdot}\kappa_{it}. \tag{8}$$

Compared with spatial smoothing, semantic smoothing is, however, more challenging to work with. In Eqn.(6), because ω_{ij} is spatially homogeneous (i.e., value depends solely on $i-j$), there exists a kernel function k that is defined over a small window and invariant to the spatial coordinates i. This no longer holds for the case of semantic smoothing. In Eqn.(8), as μ_{ij} is dependent on d_{ij} rather than $i-j$, the kernel κ is now coupled with both t and i. Although it is possible to make μ_{ij} spatially homogeneous, for example, by embedding $\mathcal{G}_v$ to one dimension (e.g., spectral embedding [2], structure preserving embedding [27]) and reordering the nodes according to the 1D coordinates, this will compromise the accuracy. To handle this problem, we define the following Gaussian kernel: $\kappa(i,t) = \frac{1}{2\pi\sigma}\exp(-d^2_{it}/2\sigma)$. Accordingly, the summation in Eqn.(8) is over all the neighboring nodes of v_i, i.e., $\forall t$, v_t and v_i are connected in $\mathcal{G}_v$.

Eqn.(8) is also equivalent to applying a Gaussian filter to the original code, except that this time the filter is specifically defined according to the semantic graph $\mathcal{G}_v$. Special care needs to be taken to the construction of $\mathcal{G}_v$ though,

as the results are typically very sensitive to it. Previous works usually build semantic graphs based on the training data. For example, in [22], the mutuel information scores estimated on training corpus were used to define connection weights. Although simple and efficient, such methods is destined to overfit the available training corpus. We recommend to construct semantic graphs by using more reliable knowledge base (e.g., WordNet, Wikipedia), or by estimating the weights on much larger universal corpora, similar to the transfer learning framework proposed by [6].

3. LANGUAGE PYRAMID

An essential property of natural language phrases is that they exist as meaningful concepts only at the right resolutions. Therefore, the choice of the scale parameter σ plays a fundamental role for the success of the 2D local smoothing framework presented in the previous section. Taken to the limit, the finest scale ($\sigma = 0$) of our 2D-smoothed representation corresponds to the original document binary code X^0 that is too erratic, whereas the coarsest scale ($\sigma \to \infty$) approaches to an overly-shrunk code, i.e., a matrix with constant entries that loses all the details. For the purpose of automatic modeling, there is no way to decide a priori which scale fits the best. More pessimistically, different concepts usually stands at different scales, making it impossible to capture all the right meanings with a single scale. For example, named entities usually range from unigram (e.g., "new") to bigram (e.g., "New York"), to multigram (e.g., "New York Times"), and even to a whole long sequence (e.g., a song name " Another Lonely Night In New York").

To this end, we borrow the idea of pyramid representation from image processing [4, 5] to textual area and present the *Language Pyramid* (LaP) model. The key insight is, as the best scale is unknown a prior and no single scale works best, the only reasonable model is to simultaneously represent the document at multiple scales.

There are two main types of pyramids: the Gaussian (or lowpass) pyramid (GP), and the Laplacian (or bandpass) pyramid (LP). In our scenario, the original code X^0 serves as the 0-level GP X_g^0, corresponding to scale 0. The whole GP is obtained by iteratively repeating the 2D local smoothing process several times, where each time the current level is smoothed to get the next level GP. That is, the i-level GP X_g^i is a smoothed version of X_g^{i-1} at scale σ_i. As the scales increases level by level, the results are a pyramid of gradually more smoothed matrices. In contrast to GP, LP encodes the representation error of GP, i.e., the l-th level LP X_l^i is formed by the difference between GP codes at two successive levels: $X_l^i = X_g^i - X_g^{i+1}$. Although Laplacian pyramid offers more compact representation (relatively lower encoding entropy [4]), it also requires more effort for intuitive interpretation (difference between two distributions is not generally comprehensible). Therefore, in the current paper, we mainly focus on Gaussian pyramid.

As an intuitive validation of LaP, Figure 2 demonstrates the visualization results for a synthetic short text "*New York Times offers free iPhone 3G as gifts for new customers in New York*", which is selected because of its containing of entities of multiple resolutions (e.g., "new", "New York", "New York Times"). For this task, we use a vocabulary containing twelve words: "new", "york", "time", "gift", "free", "iPhone", "customer", "apple", "egg", "city", "service" and "coupon", where the last four words are chosen because of their strong

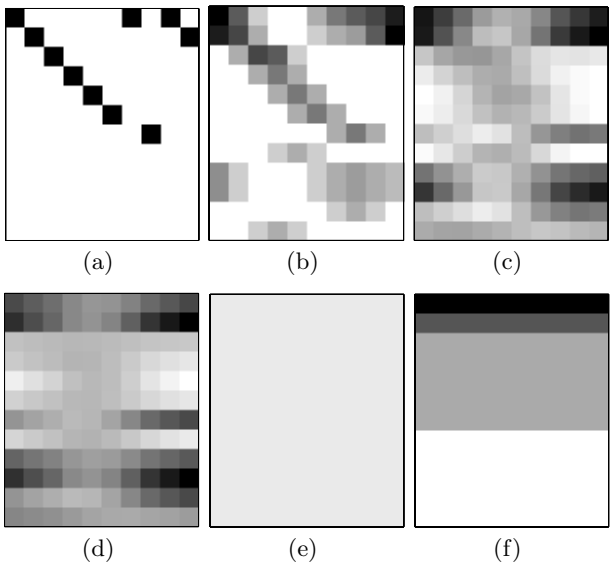

Figure 2: LaP multi-scale visualization of the text "New York Times offers free iPhone 3G as gifts for new customers in New York". (a) Original binary code ($\sigma = 0$). (b-e) 2D smoothed codes at scales $\sigma =$ 1, 2, 4 and ∞ respectively. (f) Bag-of-Word code i.e., spatial-only smoothing at ($\sigma_w = \infty$, $\sigma_v = 0$). (a-e) forms a 5-level Gaussian pyramid. Refer to text for more details.

correlations with the words appearing in the text. The semantic graph is constructed based on pairwise mutual information scores estimated on the RCV1 corpus [17] as well as a large scale repository of web search queries. After removing stop-words (i.e., words appearing in the text but not the vocabulary) and stemming, the text is represented as a 12×10 binary code, which is a sharp and erratic representation encoding precisely which word appears at which position, as shown the white-and-black image in Figure 2(a). The 2D smoothed LaP [3] codes are shown in Figure 2(b-e) at scales $\sigma = 1$, 2, 4 and ∞ respectively, which, along with (a), form a 5-level Gaussian pyramid at a sequence of progressively coarser resolutions. As a reference, (f) shows the BOW representation, which is equivalent to a 2D smoothed code at scales ($\sigma_w = \infty, \sigma_v = 0$). Compared with BOW, LaP not only captures the spatial correlations at multiple resolutions (e.g., the named entities "New York" and "New York Times"), but also boosts the probability of semantical topics underlying the text, for example, "iPhone" → "apple", "customer" → "service", "free" and "gift" → "coupon", "new" and "iPhone" → "egg" (probably due to the online electronics store `newegg.com`), and so on .

Complexity At the first glance, the representation costs (e.g., computational time and storage spaces) of LaP seemingly discount its advantages. However, the overhead, compared with BOW, is actually not as much as one imagines. The computational complexity for each layer of LaP

[3]In all our implementations, to allow efficient unified treatment, both w and v axes are smoothed with the same scale parameter σ and window width T, which are achieved by normalizing the v-axis distances to the same order as the w-axis distances.

is $\mathcal{O}(TMN)$; and overall $\mathcal{O}(LTMN)$ for an L-layer pyramid. Compared with BOW, which is $\mathcal{O}(MN)$, the relative overhead, $\mathcal{O}(LT)$, is reasonably small because in practice, the smoothing width T and the number of layers L are both small numbers, e.g., in our simulation, T=5, L=5. The storage complexity of LaP is $\mathcal{O}(LMN)$ overall. Compared with BOW, which is $\mathcal{O}(M)$, LaP requires $\mathcal{O}(LN)$ times spaces. However, for short texts (e.g., web search queries, social community questions and Twitter mini-posts), which span very small spatial spaces, only trivial amount of storage spaces are increased by LaP.

To further reduce the overhead, for popular text analysis tasks such as document classification and ad hoc retrieval, we show in the next section that by using kernel methods or similarity-based relevance models, the high-dimensionality of LaP is no more a concern.

4. MULTI-SCALE TEXT ANALYSIS

LaP opens a new dimension for text modeling: (*i*) it captures both semantic and spatial contents of documents, enabling more accurate text modeling; (*ii*) it provides multi-scale representations for texts, allowing texts to be analyzed in a scale-invariant fashion; (*iii*), it provides an imagic view for document, not only facilitating natural language understanding and visualization, but also enabling well-established image analysis tools to be applied to text processing, e.g., matching, segmentation, description, interests points detection, classification. As such, LaP could be used in text mining tasks in a variety of ways. In this section, we explore two instantiations of LaP as it applied to multi-scale text analysis. Specifically, in the scenario of supervised text mining, we describe a method to combine the matrices in LaP model to form a *multi-scale kernel* (MSK), upon which popular kernel based learning methods could be applied. In addition, we generalize the language models to a multi-scale version, and apply it in the task of document retrieval.

4.1 Multi-Scale Text Kernels

Suppose we are learning classification rules from a training corpus $\mathcal{D} = \{d_k, y_k\}_{k=1}^{K}$, where d is a document, and y its label. Let the LaP representation for d_k be $\{X_k^l\}_{l=1}^{L}$, a pyramid contains totally L levels. The documents are in advance normalized to the same length so that the matrices in LaP are of the same size $M \times N$, $\forall k$. Each level l defines a single-scale kernel $\mathbb{R}^{M\times N} \rightarrow \mathbb{R}$: $\mathsf{k}_l(d_k, d_j) = \langle X_k^l, X_j^l \rangle$, where $\langle \cdot, \cdot \rangle$ denotes a matrix-variate inner product $\mathbb{R}^{M\times N} \times \mathbb{R}^{M\times N} \rightarrow \mathbb{R}^+$, e.g., Frobenius product, Gaussian RBF similarity, Jensen-Shannon divergence [16]. The task can thus be naturally formulated as a *multiple kernel learning* (MKL) problem, i.e.: to learn an optimal combination kernel $\mathsf{k} = \sum_{l=1}^{L} \alpha_l \mathsf{k}_l$, where $0 \leqslant \alpha_l \leqslant 1$ and $\sum_{l=1}^{L} \alpha_l^2 = 1$.

From a methodological point of view, MKL approaches can be divided into two categories: the *Wrapper* methods that learn α's to minimize the training error of a specified type of classifiers (e.g., SVMs), and the *Filter* methods that learn α's independent of any classifier. Despite the facts that Wrapper MKL provide less generic results than Filter, and that they are usually solved by very time-consuming optimizers (e.g., quadratic-constrained quadratic programming [1], semi-definite programming [14]), almost all the existing MKL techniques are essentially Wrapper methods as they all stick to SVM classifiers.

In this paper, both Wrapper and Filter MKL are implemented to build MSKs. For Wrapper, we employ the SimpleMKL MKL toolbox [26]. For Filter, we present BEM (Bayes-Error-Minimization) MKL, a new MKL method that is orthogonal to any classifier and extremely efficient to solve. BEM-MKL is inspired by the Relief [12] feature-weighting algorithm, and is formulated as convex optimization:

$$\begin{aligned} &\max \sum\nolimits_{k=1}^{K} \sum\nolimits_{l=1}^{L} \alpha_l m_{kl} \\ &\text{subject to : } \alpha_l \geqslant 0, \sum\nolimits_{l=1}^{L} \alpha_l^2 = 1 \end{aligned} \tag{9}$$

where $m_{kl} = \triangle_l(d_k, m_k) - \triangle_l(d_k, h_k)$ is a heuristic margin; h_k, called "nearest-hit", is the nearest neighbor of d_k within the same class, whereas m_k, the "nearest-miss", is the nearest neighbor of d_k from different classes; $\triangle_l(d, d') = \sqrt{\mathsf{k}_l(d,d) + \mathsf{k}_l(d',d') - 2\mathsf{k}_l(d,d')}$ is the l-level kernel distance. This convex formulation is theoretically sound as it has been shown to minimize the nonparametric Bayes error on the training data [31]. In addition, it has a close-form solution that is extremely efficient to obtain:

$$\boldsymbol{\alpha} = (\bar{\mathbf{m}})^+ / ||(\bar{\mathbf{m}})^+|| \tag{10}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_L]^\top$, the margin vector $\bar{\mathbf{m}} = [\bar{m}_1, \ldots, \bar{m}_L]^\top$ with entry $\bar{m}_l = \frac{1}{K}\sum_{k=1}^{K} m_{kl}$, and $(\cdot)^+$ denotes the positive-part operator, i.e., $(a)^+ = \max(a, 0)$.

4.2 Multi-Scale Language Models

Extracting users' information needs from their input queries is fundamental to the success of information retrieval systems, which is tremendously challenging because queries contain far poorer information than normal texts, e.g., they are usually much shorter (3 words on average), noisier (e.g., no grammatical structure) and less well-formed (e.g., typos, misspellings). Worse still, the standard BOW query model further abandons all the spatial information, making it impossible to capture the precise search intents from queries. For higher retrieval accuracy, it is therefore necessary to go beyond bag-of-word and base search on text representation methods that are able to encode richer knowledge of user submitted queries. The LaP model seems to be a good choice for this purpose.

In ad hoc text retrieval, we are concerned with ranking a document d with response to a query q based on a relevance score function $r(q, d)$. A standard approach to learning $r(q, d)$ is the (unigram) language models [25], which define the probability of observing a piece of text as the products of observing each word. In its raw form, it coincides with the BOW representation. Since LaP is motivated as nonparametric estimation of a 2D generative model, each level of the LaP code could be seen as a single-scale 2D nonparametric language model, defining the probability of observing each word at each text position. Accordingly, the LaP matrices define a set of multi-scale language models, i.e., language models capturing semantic and spatial correlations at different resolutions. In this section, we exploit LaP for ad hoc retrieval.

Let $X_q^l \in \mathbb{R}^{M\times N_q}$ and $X_d^l \in \mathbb{R}^{M\times N_d}$ be the l-th level LaP representations for query q and document d respectively. The l-th level relevance score $r_l(q, d)$ is a function $\mathbb{R}^{M\times N_q} \times \mathbb{R}^{M\times N_d} \rightarrow \mathbb{R}^+$. As q and d are of different dimensions, we consider the following two approaches to matching them:

Table 1: Text classification corpora statistics

corpus	#texts	avg len	#classes	#text/class
RCV1	161311	134.2	10	16131
Yahoo!	205280	9.5	101	2144.2

1. *Global Matching*: Documents and queries are all normalized to the same dimension, and matched globally by a similarity function (these defined in Section 4.1.). Here, we consider the Jensen-Shannon similarity:

$$s(X_q^l, X_d^l) = \frac{1}{2}KL(X_d^l||\bar{X}^l) + \frac{1}{2}KL(X_q^l||\bar{X}^l) \quad (11)$$

where $KL(X^1||X^2) = \sum_i \sum_j X_{ij}^1(\log X_{ij}^1 - \log X_{ij}^2)$ is the Kullback-Leibler (KL) divergence between two 2D distributions, and $\bar{X}^l = (X_q^l + X_d^l)/2$.

2. *Local Matching*. Instead of normalizing and matching X_q^l with X_d^l globally, we can locally match the query X_q^l with part of the document X_d^l. Particularly, we consider two matching methods:

   - *Matching with a sliding window*. Match X_q^l exhaustively with every same-sized sub-matrix of X_d^l, i.e., $X_{d[i:i+N_q-1]}^l$, and combine the matching scores as the relevance score: $r_l(q,d) = \sum_i s(X_q^l, X_{d[i:i+N_q-1]}^l)$, or $r_l(q,d) = \max_i s(X_q^l, X_{d[i:i+N_q-1]}^l)$;
   - *Matching around interests points*. This approach first apply interest point detection algorithms [20] to both X_q^l and X_d^l, then matches them locally between region pairs centered at each interest point. We will leave this investigation for future works.

The overall relevance function could be obtained by aggregating all the single-scale relevance functions. For simplicity, assume a linear aggregation: $r(q,d) = \sum_l^L a_l r_l(q,d)$, where $a_l \geqslant 0$, $\sum_l a_l^2 = 1$, is the creditability weight for the l-th level. Consider a supervised aggregation setting, where we are given a training corpus containing a set of query q, and for each q a set of documents $\{d_i^q\}$ along with their relevance judgements. We use the following convex optimization to learn a's:

$$\begin{aligned} &\max \sum_q \sum_{d_i^q \succ d_j^q} \sum_{l=1}^L a_l(r_l(q,d_i^q) - r_l(q,d_j^q)) \\ &\text{subject to}: a_l \geqslant 0, \sum_{l=1}^L a_l^2 = 1 \end{aligned} \quad (12)$$

where $d_i^q \succ d_j^q$ means d_i^q is more relevant to q than d_j^q. This formulation also has an efficient close-form solution:

$$\mathbf{a} = (\bar{\mathbf{h}})^+ / ||(\bar{\mathbf{h}})^+|| \quad (13)$$

where $\mathbf{a} = [a_1, \ldots, a_L]^\top$, $\bar{\mathbf{h}} = [\bar{h}_1, \ldots, \bar{h}_L]^\top$, and $\bar{h}_l = \sum_q \sum_{d_i^q \succ d_j^q}(r_l(q,d_i^q) - r_l(q,d_j^q))$, the inner summation is over all relevant-irrelevant document pairs. This algorithm is refereed to as *Bayes Error Minimization based Multi-scale Language Model*, or BEM-MLM.

5. EXPERIMENTS

In this section, we test LaP by experimenting with the two instantiations. Specifically, we evaluate the effectiveness of the multi-scale text kernels in text classification tasks, and the multi-scale language models in the scenario of ad hoc text retrieval. We aim at comparing text representation models only. To rule out other factors, LaP is compared with BOW counterparts, which are popular in existing text analysis systems as the standard or the state-of-the-art methods.

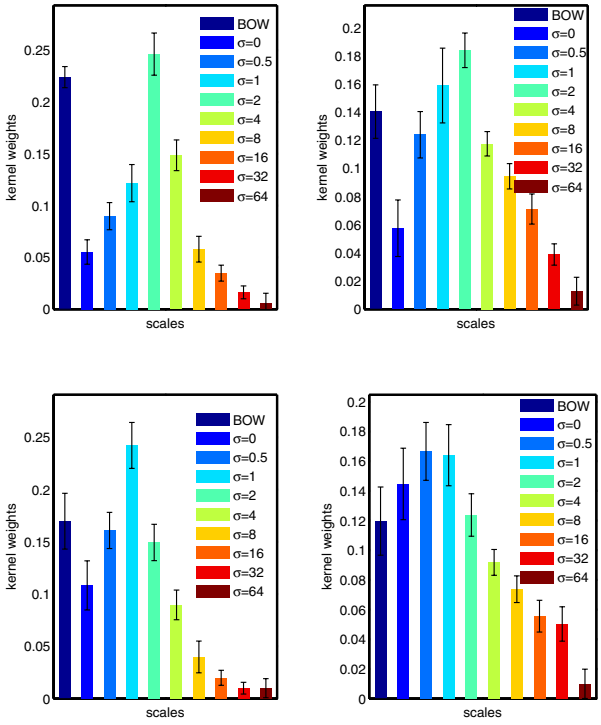

Figure 3: Kernel weights learned by (left) SimpleMKL and (right) BEM-MKL for BOW kernel as well as single-scale LaP kernels. Shown results on (top) moderate-length texts (RCV1) and (bottom) short texts (Yahoo! queries).

5.1 Text Classification

5.1.1 Setup

We evaluate multi-scale text kernels (MSKs) on two real-world text sets (Table 1). The RCV1 [17] version-2 corpus is a benchmark for text classification, which contains over 804K documents from 103 classes. Our experiments focused on 161,311 documents from the following ten leaf-node topics: `C11, C24, C42, E211, E512, GJOB, GPRO, M12, M131` and `M142`. The documents in this corpus are of moderate lengths (134 words on average, min: 8, max: 3441). The other data set is a snapshot of `answers.yahoo.com` crawled in early 2008, containing 205,280 queries from 101 classes. Texts (i.e., queries) in this set are of relatively short lengths, ranging from 3 to 27 words (9.5 on average). The basic characteristics of the data sets are summarized in Table 1. For both corpora, the texts are first stop-worded and stemmed. For the purpose of computational efficiency, the top 20% (approximately 10K) words with the highest DFs (document frequencies) are selected as the final vocabulary [4]; all

[4]According to [33] and many others, words with high DFs are usually most informative for text classification.

Table 2: Testing set accuracy. Compared models: BOW with TF or TFIDF attributes, single-scale LaP code with optimal scale σ^*, and LaP multi-scale codes based kernel combined using SimpleMKL and BEM-MKL. Used kernel forms: linear (Frobernius) kernel, RBF Gaussian kernel and Jesen-Shannon kernel. Results where LaP are significantly better (p-value less than 0.05) than BOW are marked with ★.

Model\Kernel	Linear	RBF	J-S
RCV1			
BOW-TF	0.8796	0.9137	0.8893
BOW-TFIDF	0.8851	0.9196	0.9028
$\mathbf{LaP}_{\sigma*}$	0.8870★	0.9219★	0.8943★
$\mathbf{LaP}_{\mathrm{Sim}}$	0.9350★	0.9514★	0.9374★
$\mathbf{LaP}_{\mathrm{Bem}}$	0.9245★	0.9378★	0.9193★
Yahoo!			
BOW-TF	0.5027	0.5231	0.5127
BOW-TFIDF	0.4982	0.5309	0.5244
$\mathbf{LaP}_{\sigma*}$	0.4996★	0.5571★	0.5519★
$\mathbf{LaP}_{\mathrm{Sim}}$	0.5426★	0.5993★	0.5725★
$\mathbf{LaP}_{\mathrm{Bem}}$	0.5394★	0.5711★	0.5676★

the out-of-vocabulary words are discarded. Both corpora use the same semantic graph, which is constructed based on pairwise mutual information scores estimated on the whole RCV1 corpus as well as a large scale web search query logs, and is further sparsified using *k-nearest neighbor* method with a sufficiently large k [5]. The documents are normalized to the length of the longest one in corpus. Document normalization is done via interpolation, in our experiment, we use bi-linear interpolation, an efficient and standard method in image processing.

5.1.2 Kernel Weights

As MKL algorithms optimize the overall goodness of the combined kernels, the kernel weights learned by MKL directly reflect the relative effectiveness of each component kernel. Figure 3 plots the kernel weights learned by SimpleMKL and BEM-MKL respectively, where the kernel pool consists of the BOW kernel as well as each LaP kernel. The experiments were conducted on all the 10 one-vs-all splits of the RCV1 corpus and the one-vs-all splits of the 20 most frequent classes in the Yahoo! query corpus, reported were results (average weights and standard deviations) of 20 random runs using Gaussian RBF kernels. The results are quite similar when using other kernel forms.

Although the two MKL methods weigh different kernels slightly differently, they agree on the following observations:

1. BOW is significantly less effective than LaP. BOW kernel only occupies 12-23% of the overall effectiveness, compared with in total 77-88% contributions of the LaP kernels;

2. Scale matters. The weights of effectiveness vary among different scales. The optimal scale is achieved at approximately $\sigma = 2$ for the moderate-length text corpus RCV1, and at $\sigma = 1$ for short texts in Yahoo! query logs. But even the optimal-scale kernel only achieves 17-26% of the total effectiveness. Whereas LaP integrates all the scales and accumulatively achieves up to 88% effectiveness.

Both observations intuitively validate the effectiveness of the LaP text representation model.

5.1.3 Classification Performance

We tested the LaP MSKs vs. BOW baselines (BOW representations with TF or TFIDF features) on text classification tasks. For both RCV1 and Yahoo! query corpora, we examined the classification performances of the SVM classifiers. Note that SVMs with TFIDF features are the state-of-art for this task [17, 36]. We train the SVMs on the one-vs-all splits of the training subsets. Three types of kernels (i.e., linear (Frobenius), RBF Gaussian and Jensen-Shannon kernels) were considered in our experiments. Reported in Table 2 are the averaged results (i.e., Micro-averaged F1 measures) over the 20 random repeats of 10-fold cross validation evaluations (i.e., 10% data as training).

There are some interesting observations from Table 2. Firstly, LaP outperforms BOW counterparts significantly for all the entries (i.e., on both corpora and for all the three kernel forms). LaP MSKs achieves amazingly up to 14.57% improvements of accuracy over BOW on short texts (the Yahoo! queries corpus). For the RCV1 corpus that contains mainly moderate-length texts, LaP MSKs outperforms BOW by up to 4.13%, which is quite encouraging because the accuracies of BOW are already very high and thus not much room for further improvements. Based on Student t-test at level 0.05, all the improvements are significant. We also note that for both corpus, MSKs learned by SimpleMKL seem to perform the best, especially when Gaussian RBF kernels are used. Yet, the performances of BEM-MKL learned MSKs are also quite competitive, considering its overwhelming computational efficiency over SimpleMKL.

The performance of single-scale LaP code relies largely on the choice of scales. If appropriate scales are set, single-scale LaP can perform significantly better than BOW. From Table 2, we observe that single-scale LaP with optimal scale parameters outperforms BOW (both TF and TFIDF) significantly in all the 6 entries. It is worth noting that scale selection turns to be more fundamental to short texts than normal texts. As an empirical validation, in Figure 4, for each single-scale LaP kernel computed on the Yahoo! query corpus, we plot the testing accuracy as a function of scales. As references, the performances of BOW and LaP kernels are shown as constant lines. Longer texts in general are more robust to scales, although the tendency is quite similar. Due to space limitation, we did not report the results for the RCV1 corpus. From Figure 4, we see that the performances of single scale LaP codes vary vastly with different scale setting. At scales that are too fine or too coarse, single-scale LaP codes even perform substantially worse than BOW. By considering multiple scales simultaneously, the multi-scale LaP codes, on the other hand, is not only robust to scale selection but also able to achieve consistently better performances.

5.2 Document Retrieval

In this section, we evaluate the multi-scale language models on an ad hoc text retrieval task. Our simulations were conducted on the OHSUMED [9] test collection, which is a set of 348,566 medical documents, 106 queries and 16,140

[5] We use k=50, attaining 0.0025% sparsity.

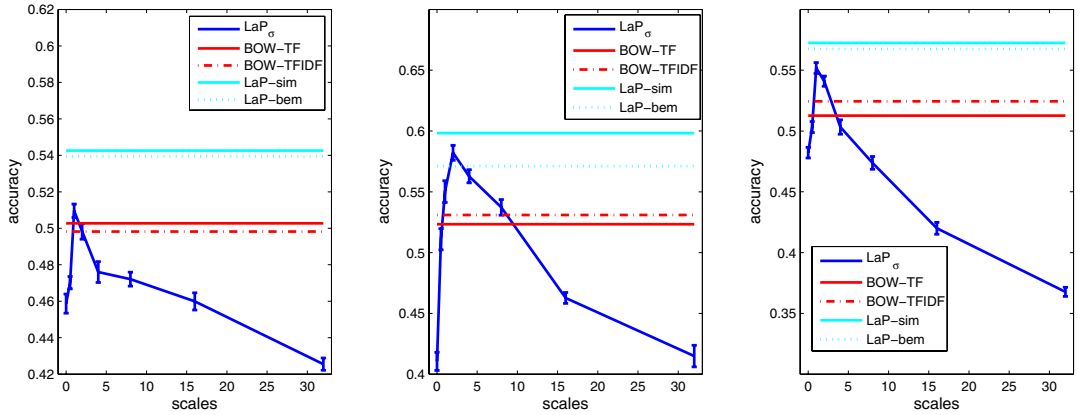

Figure 4: Testing accuracy vs. scales for LaP. Used kernels: (left) linear, (middle) Gaussian RBF, and (right) Jensen-Shannon.

Table 3: Retrieval performance comparison. Compared models: Unigram LMs (BOW), Mutli-scale LMs using global matching ($\mathbf{LaP}_{\text{Global}}$), local matching with sum operator ($\mathbf{LaP}_{\text{Local.Sum}}$) and max operator ($\mathbf{LaP}_{\text{Local.Max}}$).

Model\Measure	MAP	P@5	P@10
BOW	0.2699	0.4812	0.4659
$\mathbf{LaP}_{\text{Global}}$	0.2713	0.5033★	0.4766
$\mathbf{LaP}_{\text{Local.Sum}}$	0.3028★	0.5405★	0.5049★
$\mathbf{LaP}_{\text{Local.Max}}$	0.2885★	0.5264★	0.4727

relevance judgements for document-query pairs on three scales: either "definitely relevant", "possibly relevant" or "irrelevant".

We tested three retrieval models described in Section 4.2: the multi-scale LMs using global matching, local matching with sliding window and sum operator, local matching with sliding widow and max operator. The unigram LMs (i.e., BOW) were used as baselines. For all the models, the standard Kullback-Leibler divergence were used as relevance functions. The weights for multi-scale LMs are learned from Eqn.(12) on document pairs ("partially relevant" documents were treated the same as "relevant" ones) from 5 randomly sampled queries. Figure 5 shows the empirical distributions of the weights obtained on 20 random repeats. BOW weights contribute higher than in classification tasks, probably because the texts in the OHSUMED corpus are mainly technical words with low-frequency. However, even for such corpus, BOW only occupies at most 24% of the overall effectiveness, much lower than LaP.

Table 3 reports the overall retrieval performances of the three retrieval models. We use three standard IR evaluation measures: the Mean-Average-Precision (MAP), Precision at N with N=5 and 10 (P@5, P@10). We observe that, in terms of MAP, LaP multi-scale LMs outperform BOW uni-gram LMs by (up to) 12.19%. Among the three retrieval models, local matching in general outperforms global matching, while local matching with a sliding window and sum operator performs the best. Based on Wilcoxon test with 0.05 significance threshold, we found, out of the 9 entries in Table 2, LaP significantly improve BOW in 6 entries.

6. RELATED WORKS

Text representation is fundamental to almost all the text analysis tasks. Although classical BOW or unigram LMs capture semantic information quite well and have gained great success in existing systems, they lose the invaluable spatial contents and thus cannot meet the requirement for more accurate text modeling, especially for short texts. Lots of efforts have been devoted to developing advanced spatial-sensitive models, most representatively, the n-gram BOW [21], n-gram LMs [35], cashed LMs [13, 10], named entity detection [7], query segmentation [11], and string kernel [19]. These models, however, achieve no significant improvement at high prices of computational resources [34, 8].

Perhaps most related to our model is the recently proposed *Locally Weighted Bag-of-Word* (LOWBOW) framework [15]. Using also locally weighted smooth kernels, LOWBOW embeds a document as a spatially smoothed curve in the high-dimensional multinomial simplex. Our LaP model can be seen as an advanced extension to LOWBOW: (*i*) LaP is locally smoothed both spatially and semantically; (*ii*) LaP represents text at multiple resolutions; (*iii*) LaP outputs matrices, much simpler than the functional outputs of LOWBOW.

LaP applies 2D smoothing, which, in semantic direction, resembles the LMs smoothing [35]. Classical smoothing methods (e.g., Laplacian, Jelineke-Mercer, and Dirichlet smoothers [34]) essentially amount to shrinking with reference distributions, neglecting the correlation relationships among words. LaP, however, uses local smoothing which conveys semantic correlation into graphs and employs graph-based learning. Similar approaches were also exploited in [22]. Our approach, however, is evidently different: (*i*)Besides semantic smoothing, LaP applies also spatial smoothing; (*ii*) Instead of solving the graph-based optimization directly as in [22], LaP reduces it to kernel convolutions, which is more efficient to obtain and also allows both semantic and spatial directions to be handled in a unified way; (*iii*) Instead of tuning parameters by cross-validation, LaP is parameter-free, i.e., it directly uses well-defined kernels; (*iv*) We use more reliable semantic graphs, more robust to training corpus noises.

Our model is built on the pyramid representations in image processing [4, 5, 18], which establish a pyramid of images

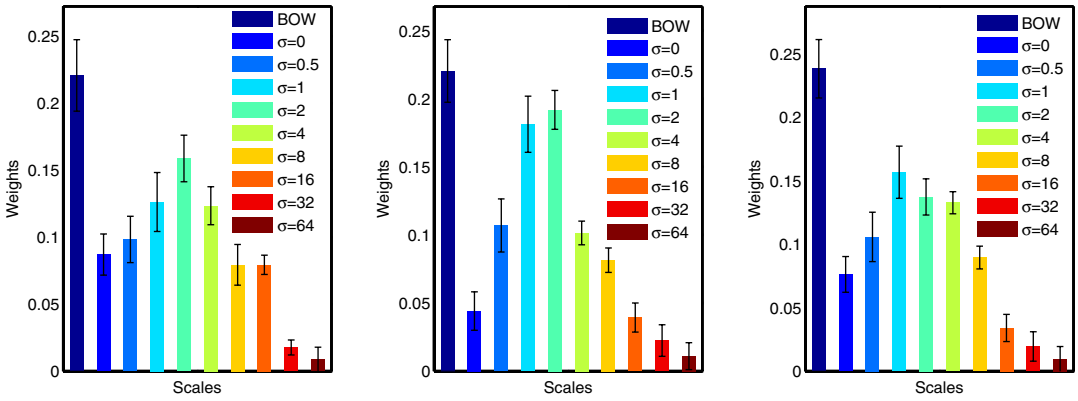

Figure 5: Weights for unigram LMs as well as single-scale LaP LMs, learned from Eqn.(12) on retrieval models: global matching (left), sliding-window based local matching using sum (middle) or max (right) operators.

smoothed at progressive resolutions to capture the multi-scale characteristics of an image. LaP for the first time presents the pyramid representations for natural language texts, and opens up the possibilities to apply well-established image processing tools for text analysis.

7. CONCLUDING REMARKS

In this paper, we have presented the Language Pyramid model for multi-scale text analysis. LaP encodes both semantic and spatial contents of a document as a probabilistic distribution over the joint semantic-spatial space. It employs nonparametric estimation with locally weighted kernels, resulting in text codes smoothed both semantically and spatially at multiple resolutions. We have instantiated two frameworks for applying LaP, i.e., multi-scale text kernels, and multi-scale language models. Evaluations on benchmark corpora indicate that the proposed model and algorithms are useful and effective for text modeling.

In principal, the more scales used in LaP, the better. Taken to the limit, we can consider an infinite number of scales, leading to a functional text representation framework, similar to the scale-space theory in image processing [18]. We will leave such investigation for future works.

This paper employed traditional classifiers (e.g., SVMs) for text classification. Advanced classifiers, such as the probabilistic models presented in [32, 30] that are capable to handle data ambiguity (e.g., multi-label classification, paragraph-wise classification), are also applicable.

We also plan to develop appropriate local descriptors (similar to the SIFT descriptor for image [20]) as well as interest point detectors on LaP, and empirically study the interest points based local-matching retrieval model proposed in Section 4.2.

Acknowledgements

We thank James M. Rehg, whose computer vision lecture inspired the idea of textual pyramid. Part of this work is supported by NSF #DMS-0736328, grants from Microsoft and Hewlett-Packard, and the financial support of the 111-Project (B07022) and NSFC.

8. REFERENCES

[1] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML '04: Proceedings of the 21st international conference on Machine learning*, page 6, New York, NY, USA, 2004. ACM.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2002.

[3] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *SIGIR '08: Proceedings of the 31th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 491–498, 2008.

[4] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31,4:532–540, 1983.

[5] J. Crowley and A. C. Parker. A representation for shap based on peaks and ridges in the difference of low-pass transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):156–170, 1984.

[6] C. B. Do and A. Y. Ng. Transfer learning for text classification. In *NIPS '05: Advances in Neural Information Processing Systems 18*, 2005.

[7] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134, June 2005.

[8] K. Hall and T. Hofmann. Learning curved multinomial subfamilies for natural language processing and information retrieval. In *ICML'00: Proceedings of the 17th International Conference on Machine Learning*, pages 351–358, 2000.

[9] W. Hersh, C. Buckley, T. J. Leone, and D. Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 192–201, 1994.

[10] R. Iyer and M. Ostendorf. Modeling long distance dependence in language: Topic mixtures vs. dynamic cache models. *IEEE Transactions on Speech and Audio Processing*, pages 236–239, 1996.

[11] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW'06: Proceedings of the 15th international conference on World Wide Web*, pages 387–396, New York, NY, USA, 2006. ACM.

[12] K. Kira and L. A. Rendell. A practical approach to feature selection. In *ICML '92: Proceedings of the 9th international conference on Machine learning*, pages 249–256, 1992.

[13] R. Kuhn and R. De Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583, 1992.

[14] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[15] G. Lebanon, Y. Mao, and J. Dillon. The locally weighted bag of words framework for document representation. *Journal of Machine Learning Research*, 8:2405–2441, 2007.

[16] L. Lee. Measures of distributional similarity. In *ACL '99: Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 25–32, 1999.

[17] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[18] T. Lindeberg. Scale-space for discrete signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:234–254, 1990.

[19] H. Lodhi, C. Saunders, J. S. Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[21] C. D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. The MIT Press, June 1999.

[22] Q. Mei, D. Zhang, and C. Zhai. A general optimization framework for smoothing language models on graph structures. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 611–618, 2008.

[23] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *SIGIR '05: Proceedings of the 28th ACM SIGIR conference on informaion retrieval*, pages 472–479, 2005.

[24] D. R. H. Miller, T. Leek, and R. M. Schwartz. A hidden markov model information retrieval system. In *SIGIR '99: Proceedings of the 22th ACM SIGIR conference on informaion retrieval*, pages 214–221, 1999.

[25] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 275–281, New York, NY, USA, 1998. ACM.

[26] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, November 2008.

[27] B. Shaw and T. Jebara. Structure preserving embedding. In *ICML '09: Proceedings of the 26th International Conference on Machine Learning*, pages 937–944, 2009.

[28] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and wikipedia. In *WWW '08: Proceedings of the 17th international conference on World Wide Web*, pages 347–356, New York, NY, USA, 2008. ACM.

[29] G. Xu, S.-H. Yang, and H. Li. Named entity mining from click-through data using weakly supervised latent dirichlet allocation. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1365–1374, New York, NY, USA, 2009. ACM.

[30] S. H. Yang, J. Bian, and H. Zha. Hybrid generative / discriminative learning for automatic image annotation. In *UAI '10: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 2010.

[31] S. H. Yang and B. G. Hu. Feature selection by nonparametric bayes error minimization. In *PAKDD '08: Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining*, pages 417–428, 2008.

[32] S. H. Yang, H. Zha, and B.-G. Hu. Dirichlet-bernoulli alignment: A generative model for multi-class multi-label multi-instance corpora. In *NIPS '09: Advances in Neural Information Processing Systems 22*, pages 2143–2150, 2009.

[33] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, 1997.

[34] C. Zhai. *Statistical Language Models for Information Retrieval*. Now Publishers Inc., Hanover, MA, USA, 2008.

[35] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.

[36] D. Zhang and W. S. Lee. Question classification using support vector machines. In *SIGIR '03: Proceedings of the 26th ACM SIGIR conference on informaion retrieval*, pages 26–32, 2003.

[37] J. Zhao and Y. Yun. A proximity language model for information retrieval. In *SIGIR '09: Proceedings of the 32th ACM SIGIR conference on informaion retrieval*, pages 291–298, 2009.