Discrete Signal Processing on Graphs for Automatic Document Classification

Michael Kellman, Raymond Xia, Matt Bauch
Electrical and Computer Engineering
Carnegie Mellon University

ABSTRACT

Discrete Signal Processing on Graphs (DSP_G) is a new area of research that extends the ideas from classical discrete signal processing to data that is most naturally represented as originating from a graph. In this work, we explore the effectiveness of DSP_G techniques in solving the problem of automatic document classification from natural language processing. Although text in a document exists as a sequence of characters, a complex relationship between words in any human language exists, and therefore the information contained in text is better suited to a graph representation than a time series. We propose a method for classification that uses graph Fourier analysis for feature extraction. Specifically, the coefficients from projecting the bag-of-words representation of a document onto the Fourier basis of a term distance graph generated from training documents are used as features. We show classification accuracies comparable to a classifier architecture commonly used in problems such as spam filtering.

I. INTRODUCTION

Classical signal processing techniques are well-established and highly effective tools for analysis and synthesis in a variety of settings involving regularly sampled, ordered data such as time series and images. However, the massive increase in availability of other types of structured data over the last decade, especially from the internet, social networks, and physical sensor networks, has created demand for a more general signal processing framework. What has become known as *Discrete Signal Processing on Graphs* (DSP_G) [1] satisfies this need by generalizing ideas from classical signal processing such as filtering, Fourier transforms, and wavelets to *graph signals*. Whereas in classical signal processing a signal is typically indexed by position in time or space, a graph signal is indexed by nodes on a graph and is therefore well suited for representing many kinds of structured data where a time series is insufficient.

While there are numerous potential applications for the data analysis and processing techniques offered by DSP_G, the majority of the work done at present has been to develop the prerequisite theories. Much work remains in evaluating the practical effectiveness of DSP_G in various settings. It is our goal in this work to explore an application in natural language processing (NLP), where graphs are frequently employed to represent the semantic relationships between words in a body of text. Example problems include automatic document classification, automatic summarization, keyword extraction, and machine translation. In this work, we focus our attention on the problem of supervised, automatic document classification (ADC).

In the remainder of this paper, we will first provide an overview of the ADC problem followed by a brief overview of relevant concepts from DSP_G. Next, we will propose a novel algorithm for ADC that applies the graph Fourier transform to an existing graph representation for text in the feature extraction step. We will then discuss our implementation of the algorithm in Python and MATLAB. Finally, we will present experimental results and conclude with a brief discussion of the algorithm's effectiveness.

II. BACKGROUND

Automatic Document Classification

In many settings, the ability to automatically sort documents into categories is crucial. Examples include email spam filtering, grouping news articles by category for presentation to users, and routing customer email in a customer service department. Such systems generally entail the design of a scheme for extracting features from text and leverage standard statistical classification techniques (Fig. 1).

Working within this basic framework, the engineer must make appropriate choices for three major components: system input, feature extraction, and classification method. In the case of ADC, the input is the body of text to be classified. There are many possible features that may be extracted, ranging from a list of word counts, known as *bag-of-words*, in the simplest case to semantic models in the most complex. Finally, a method of classification well suited to the type of text and choice

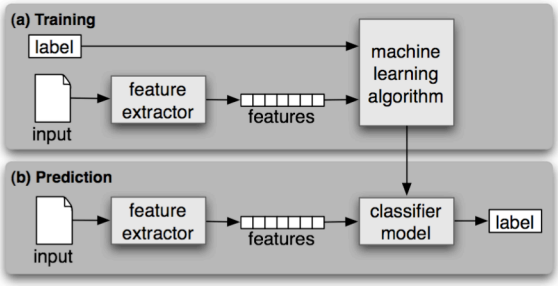

Fig 1. General framework for supervised, automatic document classification. *Source: http://nltk.org*

of features must be chosen. Popular choices include support vector machines (SVM) and naïve Bayes. Statistical models more tailored to the specific problem may also be built. As one example, a common setup for spam classification combines bag-of-words feature extraction with naïve Bayes classification. N-grams feature extraction is also used in many applications [2].

As our goal is to apply DSP_G to an ADC system, we must first find a suitable graph representation for text documents. For many tasks within the field of NLP besides ADC, many graph representations have been formulated to capture the information in the relationships between words in text. In an N-gram language graph, each node represents a contiguous sequence of syllables, phonemes, words, or letters from a body of speech or text, and edges represent transition likelihoods. Another useful representation is based on the concept of eigenvector centrality, in which an adjacency matrix is constructed from intra-sentence cosine similarity [4]. In another representation successfully applied to text novelty detection, the point-wise mutual information (PMI) between the words in a sentence is used to weight the edges between words [3]. A fourth popular framework, *generalized latent semantic analysis* (GLSA) [5], also uses PMI as a measure of semantic association between words and can be modified for use as a graph-based document representation [6]. PMI will be discussed further in section III.

Discrete Signal Processing on Graphs

For a detailed introduction to DSP_G, see [1]. In this subsection we will provide only a brief introduction to relevant concepts employed in our algorithm, discussed in the next section.

The most basic concept from DSP from which we can begin to formulate DSP_G generalizations is the *shift*. Whereas for time series data a shift represents a translation in time, a *graph shift* represents the spreading and mixing of a graph signal's values according to the graph adjacency matrix. The time shift of time series data is a special case of the graph shift. The graph shift is also used in the derivation of the *graph filter*, which reduces to its analog, the FIR filter, in special cases.

In DSP, the spectral decomposition of a signal refers to the identification of a basis that is invariant to filtering. Analogously, spectral decomposition in DSP_G is performed by identifying a basis for which the graph signal is invariant to filtering. In general, this is accomplished through the Jordan decomposition of the graph's adjacency matrix. However for symmetric adjacency matrices (undirected graphs), which we will be working with exclusively in the following section, spectral decomposition is equivalent to the eigendecomposition of the adjacency matrix. The Fourier coefficients of a graph signal, then, are just the coefficients obtained from projecting the signal onto the eigenbasis.

III. PROPOSED CLASSIFICATION METHODS

We formulated and evaluated several document classification schemes. In this section we will focus most of our attention on what became our most successful classifier, pointing out where we departed from the suggestions made in similar work [3] on the problem of text novelty detection.

Adjacency Matrix Construction

In formulating features for document classification, we assume that documents belonging to the same class will likely not only have some terms in common, but that the documents also share some semantic theme. In other words, two documents about the same topic may in fact differ in exact terminology, but will likely construct ideas similarly and contain sentences expressing the same ideas. To capture this information in our specification of an adjacency matrix, we follow [3] in assuming that 1) all words in a sentence are related to all the other words in the sentence within some window of word distance, and 2) the strength of their relationships is positively correlated with their proximity within a sentence. We therefore define the edge weights in an adjacency matrix defined for one sentence as

$$wt_{i,j} = \frac{1 + PMI(i,j)}{d_{i,j}^2} \tag{1}$$

where $d_{i,j}$ is the distance, in words, of word i from word j in the sentence. If this distance exceeds a window size the weight is set to zero for the given word pair. A window size of 6 words seems to perform best. PMI is used as a measure of word co-occurrence and is defined globally—that is, over the entire body of input text—as

$$PMI(i,j) = \log_2 \frac{P(i,j)}{P(i)P(j)} \tag{2}$$

In the definition of PMI, $P(i,j)$ is the prior probability of word i and word j occurring together in a

given sentence and $P(i)$ is the prior probability of word i occurring in a given sentence. In the calculation of edge weights for a body of text consisting of either a single or multiple documents, the priors in the definition of PMI are calculated using all available information. The edge weights in the adjacency matrices for each sentence in the input document are summed to obtain an adjacency matrix for the entire document. This formulation therefore captures both the local information provided by word proximity within a sentence as well as the global information provided by word co-occurrence frequency. Again following [3], we will refer to the resulting graph as the *term distance* (TD) graph. Large edge weights between words signify a strong semantic association between words.

Document Feature Extraction

In our most successful proposed algorithm, we depart from [3] at the suggestion that features for classification can be derived directly from the TD graph. The classifiers we developed and evaluated with TD graph features consistently performed poorly. In one case, a TD graph was calculated for each document class from several training examples. The TD graph was then calculated for each input document, and the coefficients from the projection of the test document's adjacency matrix onto the Fourier basis of each class's adjacency matrix were used as features.

So, rather than deriving features directly from TD graphs, we opt to work with graph signals themselves. Since the TD graph's nodes correspond to words, i.e. our graph signal is word-indexed, we must choose a function to map documents to word-indexed graph signals. For it's simplicity widespread use in other classifiers, we choose to map documents to a bag-of-words, or word histogram, graph signal representation. Intuitively, the amplitude of the graph signal at each word is given simply by the number of times the word occurs.

To calculate the graph Fourier transform of a graph signal, we must first prune words from the signal that are not in the Fourier basis and add zeros to the signal for words that are in the basis but not contained in the test document. Then, the graph signal is projected onto the Fourier basis calculated from the eigendecomposition of the TD graph adjacency matrix. In our algorithm (Fig. 2), an equal sample of documents from each class is used to calculate a single TD graph, and therefore a single Fourier basis.

Note two important qualities of our feature extraction step. First, we have expanded upon and built a framework for extracting semantic information from documents without having done any explicit linguistic modeling. The result is that our feature extraction is extremely simple both in terms of implementation and required computation relative to more complex alternatives that impose linguistic structure, though we expect it to capture some of the same semantic information. Second, the Fourier basis is learned in an unsupervised manor, meaning that it's likely not the most discriminative choice of basis possible in which to perform the classification.

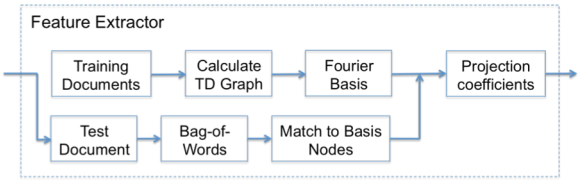

Fig 2. Feature extraction from TD graphs

Classification

As mentioned previously, a common and simple method for ADC is naïve Bayes classification of bag-of-words features, where the word counts are assumed to be multinomial distributed. We modify this approach by instead using the graph Fourier transform of the bag-of-words signal as a feature space, and classify with naïve Bayes as well as SVMs with various kernel types. This approach is analogous to the problem of waveform classification, where Fourier transform coefficients provide a much more informative feature space than raw signal values. In our case, we hypothesize that the Fourier basis represents a more informative feature space than a simple bag-of-words by also encoding some semantic information.

IV. IMPLEMENTATION

In this section we outline the computational methods for generating TD graphs for the purpose of training and testing our proposed hypothesis.

Preprocessing and TD Graph Generation

For text parsing and graph creation, we opted to write scripts in Python. In addition to having a more powerful string library and more flexible data structures for operating on sparse matrix representations, Python is also the language of the Natural Language Toolkit (NLTK), which provides powerful text parsing capabilities out-of-the-box [7].

Before the generation of graphs, the text must be preprocessed to aid in the effectiveness of feature extraction. First, all bodies of text are stripped of some predefined indiscriminative words. While these words provide aid to humans reading the text, they do not contain knowledge that will provide information for classification. These words are known as *stop words*. We prune stop words from all text input. Another important consideration is that words can be written in many different forms—conjugation, tense, plurality—without changing meaning in a significant way. While these characteristics likely often do contain information that could aid in classification, we opt to normalize the text to

promote denser TD graphs. All plural words are made singular and punctuation and capitalization are stripped.

TD graphs are calculated using the weight formula as given in eq. 1 above for a subset of the training articles. They are exported as in CSV format, and will be imported into MATLAB where the rest of algorithm is implemented.

TD Graph Fourier Basis and Classification

Once the text processing has been completed and sparse TD graphs have been generated, we opt to implement the rest of the proposed ADC algorithm in MATLAB, which works naturally with matrix representations. We use the Graph Signal Processing Toolbox, developed by the Signal Processing Laboratory at the Ecole Polytechnique Fédérale de Lausanne [7], to compute the Fourier basis of the TD graph adjacency matrix. We also leveraged MATLAB's built in naïve Bayes classification functions and LIBSVM for SVM classification [9].

V. EXPERIMENTS

NLTK provided several corpora of text articles that are categorically tagged. The Brown corpus provided by NLTK best suited our needs, by providing numerous articles and a variety of categories [7]. To evaluate our algorithm we choose three of these categories. Other ADC applications might choose to evaluate on two classes, in the case of spam filtering, or to evaluate on many classes. Using equal numbers of articles from each class we partitioned our data into three groups: data that the TD graph will be computed with, data that will be used to train the classifier, and data that will be used to test the classifier. To then further evaluate our methods we cross-validated our results by choosing to partition the data for training and testing K different ways (K-fold). This gives an average accuracy and a standard deviation of the accuracy. The presented results are achieved with a partition of 300 articles used for computing the TD graph, 100 from each category, 900 articles used for training and testing, 300 from each category, and with K equal to 30 for cross-validating the results.

VI. RESULTS

Using the proposed three-class evaluation our algorithm, with a kernel SVM as the classifier, performed with an average classification accuracy of 70.11% with a standard deviation of 9.9%. To achieve these results we were required to tune the parameters of our kernel SVM, the results of the tuning are plotted in Fig. 3. Using the same data a *bag-of-words*, with a Naïve Bayes classifier, implementation achieved an average classification accuracy of 71.33% with a standard deviation of 11.09%. The similar performance between these two techniques suggests that our novel method could be used as an alternative to the baseline *bag-of-words* algorithm.

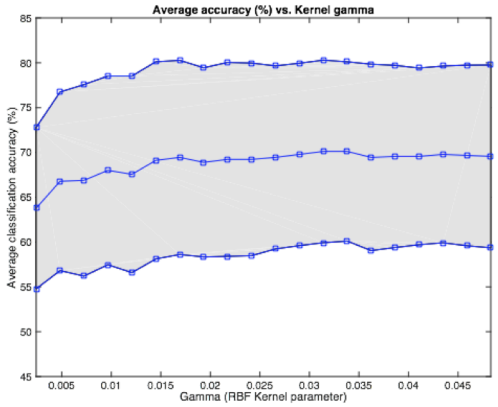

Fig 3. Cross validation of Gaussian Kernel width in SVM document classification using bag-of-words Fourier basis

VII. CONCLUSION AND FUTURE WORK

Given that we have achieved moderate success in matching baseline performance with a relatively unrefined algorithm and implementation, it's plausible that DSP_G feature extraction method proposed in this paper could be improved given some modifications and tuning.

One modification that we hypothesize will result in a higher classification accuracy is the use of multiple Fourier basis. Explicitly, calculating a TD graph and Fourier basis for each possible class and concatenating the Fourier coefficients from projecting test instances onto each basis should result in a more discriminative feature space. We expect that each Fourier basis should provide a more informative representation for documents belonging to its class, and therefore should provide a higher level of certainty in classification.

In future work, we'd also like to test our system on a larger dataset with many more classes, as we suspect the semantic information captured in our features may provide a more significant advantage over naïve Bayes with bag-of-words in this harder problem setting.

REFERENCES

[1] Sandryhaila, A., & Moura, J. M. F. (2012). *Discrete Signal Processing on Graphs*. Retrieved from http://arxiv.org/pdf/1210.4752

[2] Goller, C., Loning J., Will, T., Wolff, W. *Automatic Document Classification: A Thorough Evaluation of Various Methods.* Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.966&rep=rep1&type=pdf

[3] Gamon, M. (2006). *Graph-Based Text Representation for Novelty Detection*. Retrieved from http://www.msr-waypoint.com/pubs/69322/novelty_camera_ready.pdf

[4] Erkan, G., Radev, D. R. (2004). *LexRank: Graph-based Lexical Cetrality as Salience in Text Summarization*. Journal of Artificial Intelligence Research 22, 457-479. Retreived from http://www.jair.org/media/1523/live-1523-2354-jair.pdf

[5] Matveeva, I. Levow, G.-A., Farhat, A. Royer, C. (2005). *Term Representation with Generalized Latent Semantic Analysis*. Retrieved from http://faculty.washington.edu/levow/papers/SynGLSA_ranlp_final.pdf

[6] Matveeva, I., Levow, G.-A. (2006). *Graph-based generalized latent semantic analysis for document representation*. Retrieved from http://people.cs.uchicago.edu/~matveeva/HLTworkshop.pdf

[7] Bird, S., Loper, E., Klein, E., (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.

[8] Nathanaël, P., Paratte, J.,Shuman, D., Kalofolias, V., Vandergheynst, P., Hammond, D. K., *GSPBOX: A toolbox for signal processing on graphs*. Arxiv e-print, 08-2014.

[9] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011.