Twitter Geolocation and Regional Classification via Sparse Coding

Miriam Cha
Harvard University

Youngjune Gwon
Harvard University

H. T. Kung
Harvard University

Abstract

We present a data-driven approach for Twitter geolocation and regional classification. Our method is based on sparse coding and dictionary learning, an unsupervised method popular in computer vision and pattern recognition. Through a series of optimization steps that integrate information from both feature and raw spaces, and enhancements such as PCA whitening, feature augmentation, and voting-based grid selection, we lower geolocation errors and improve classification accuracy from previously known results on the GEOTEXT dataset.

Introduction

Location information of a user provides crucial context to build useful applications and platforms for social networking. It has been known that one can geolocate a document generated online by identifying geographically varying characteristics of the text. In particular, a variety of linguistic features such as toponyms, geographic and dialectic elements can be used to create a content-based geolocation system.

In recent years, there is a growing interest to leverage social networking and microblogging as a special sensor network to detect natural disasters, terrorism, and social unrests. Social media research has experimented with supervised topical models trained on Twitter messages (known as *tweets* of 140 characters or fewer) for geolocation. However, predicting accurate geocoordinates using only microblog text data is a hard problem. Microblog text can deviate significantly from normal usage of a language. The processing and decomposition of casual tweets often require a complex topical model, whose parameters must be trained meticulously with large labeled dataset. The fact that only 3% of tweets are geotagged indicates the difficulty of supervised learning due to limited availability of labeled training examples. Moreover, learning algorithms that use raw data directly without computing feature representations can severely limit the prediction performance.

In this paper, we present a new approach for the Twitter geolocation problem, where we want to estimate the geocoordinates of a Twitter user solely based on associated tweet text. Our approach is based on unsupervised representational learning via sparse coding (1997). We show that sparse coding trained on unlabeled tweets leads to a powerful representational mapping for text-based geolocation comparable to topical models for lexical variation. Such mapping is particularly effective when it is used in a similarity searching heuristic in location estimation such as k-Nearest Neighbors (k-NNs). When we apply a series of enhancements consisting of PCA whitening, regional dictionary learning, feature augmentation, and voting-based grid selection, we achieve competitive performance standing at 581 km mean and 425 km median location errors for the GEOTEXT dataset. Our classification accuracy for 4-way US Regions is a 9% improvement over the best topical model in prior literature, and the 14% for 48-way US States.

Related work

Eisenstein *et al.* (2010) propose a latent variable model with two sources of lexical variation: topic and region. The model describes a sophisticated generative process from multivariate Gaussian and Latent Dirichlet Allocation (LDA) for predicting an author's geocoordinates from tweet text via variational inference. Hong *et al.* (2012) extends the idea to model regions and topics jointly. Wing & Baldridge (2011) describe supervised models for geolocation. They have experimented with geodesic grids having different resolutions and obtained a lower median location error than Eisenstein *et al.* Roller *et al.* (2012) present an improvement over Wing & Baldridge with adaptive grid schemes based on k-d trees and uniform partitioning.

Data

We use the CMU GEOTEXT dataset (2010). GEOTEXT is a geo-tagged microblog corpus comprising 377,616 tweets by 9,475 users from 48 contiguous US states and Washington D.C. Each document in the dataset is concatenation of all tweets by a single user whose location information is provided as GPS-assigned latitude and longitude values (geocoordinates). The document is a sequence of integer numbers ranging 1 to 5,216, where each number represents the position in `vocab`, the table of uniquely appearing words in the dataset. While recognizing its limitation to US Twitter users only, we choose GEOTEXT for fair comparison with the prior approaches mentioned in related work that use the same dataset.

Approach

Our approach is essentially semi-supervised learning. We concern the shortage of labeled training examples in practice, and hence pay special attention on the unsupervised part consisting of sparse coding and dictionary learning with unlabeled data.

Preliminaries

Notation. Let our vocabulary `vocab` be an array of words with size $V = |\texttt{vocab}|$. A document containing W words can be represented as a binary *bag-of-words* vector $\mathbf{w}_{BW} \in \{0,1\}^V$ embedded onto `vocab`. The ith element in $\mathbf{w}_{BW}$ is 1 if the word `vocab[`i`]` has appeared in the text. We denote a word-counts vector $\mathbf{w}_{WC} \in \{0,1,2,\dots\}^V$, where the ith element in $\mathbf{w}_{WC}$ now is the number of times the word `vocab[`i`]` has appeared. We can also use a word-sequence vector $\mathbf{w}_{WS} \in \{1,\dots,V\}^W$. That is, w_i, the ith element in $\mathbf{w}_{WS}$ represents the ith word in the text, which is `vocab[`w_i`]`. We use patch $\mathbf{x} \in \mathbb{R}^N$, a subvector taken from $\mathbf{w}_{BW}$, $\mathbf{w}_{WC}$, or $\mathbf{w}_{WS}$ for an input to sparse coding. For m-class classification task, we use label $l \in \{c_1, c_2, \dots, c_m\}$, where c_j is the label value for class j. For geolocation task, label $l = (\text{lat}, \text{lon})$ is the geocoordinate information.

Sparse coding. We use sparse coding as the basic means to extract features from text. Given an input patch $\mathbf{x} \in \mathbb{R}^N$, sparse coding solves for a representation $\mathbf{y} \in \mathbb{R}^K$ in the following optimization problem:

$$\min_{\mathbf{y}} \|\mathbf{x} - D\mathbf{y}\|_2^2 + \lambda \cdot \psi(\mathbf{y}) \tag{1}$$

Here, $\mathbf{x}$ is interpreted as a linear combination of basis vectors in an overcomplete dictionary $D \in \mathbb{R}^{N \times K}$ ($K \gg N$). The solution $\mathbf{y}$ contains the linear combination coefficients regularized by the sparsity-inducing second term of Eq. (1) for some $\lambda > 0$.

Sparse coding employs ℓ_0- or ℓ_1-norm of $\mathbf{y}$ as $\psi(.)$. Because ℓ_0-norm of a vector is the number of its nonzero elements, it precisely fits the regularization purpose of sparse coding. Finding the ℓ_0-minimum solution for Eq. (1), however, is known to be NP-hard. The ℓ_1-minimization known as Basis Pursuit (2001) or LASSO (1994) is often preferred. Recently, it is known that the ℓ_0-based greedy algorithms such as Orthogonal Matching Pursuit (OMP) (2007) can run fast. Throughout this paper, our default choice for sparse coding is OMP.

How can we learn a dictionary D of basis vectors for sparse coding? This is done by an unsupervised, data-driven process incorporating two separate optimizations. It first computes sparse code for each training example (unlabeled) using the current dictionary. Then, the reconstruction error from the computed sparse codes is used to update each basis vector in the dictionary. We use K-SVD algorithm (2006) for dictionary learning.

Baseline methods

We consider two localization tasks. The first task is to estimate the geocoordinates given tweets of unknown location. The second task is to perform multiclass classification that predicts a region (Northeast, Midwest, South, West) or state in the US. In the following, we present our baseline methods for these two tasks.

Geolocation. We use the following steps for geolocation.

1. (Text embedding) perform binary, word-counts, or word-sequence embedding of raw tweet text, using `vocab`;
2. (Unsupervised learning) use unlabeled data patches to learn basis vectors in a dictionary D;
3. (Feature extraction) do sparse coding with labeled data patches $\{(\mathbf{x}^{(1)}, l^{(1)}), (\mathbf{x}^{(2)}, l^{(2)}), \dots\}$, using the learned dictionary D and yield sparse codes $\{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots\}$;
4. (Feature pooling) perform max pooling over a group of M sparse codes from a user to obtain pooled sparse code $\mathbf{z}$ such that the kth element in $\mathbf{z}$, $z_k = \max(y_{1,k}, y_{2,k}, \dots, y_{M,k})$ where $y_{i,k}$ is the kth element from $\mathbf{y}_i$, the ith sparse code in the pooling group;
5. (Tabularization of known locations) build a lookup table of reference geocoordinates associated with pooled sparse codes $\mathbf{z}$ (from labeled data patches).

When tweets from unknown geocoordinates arrive, the geolocation pipeline comprises preprocessing, feature extraction via sparse coding (without labels), and max pooling. Using the resulting pooled sparse codes of the tweets, we find the k pooled sparse codes from the lookup table that are closest in cosine similarity. We take the average geocoordinates of the k-NNs.

Region and state classification. We employ the following method for classification.

1. Repeat steps 1 to 4 from the geolocation method;
2. (Supervised classifier training) use pooled sparse codes $\mathbf{z}$ as features to train a classifier such as multiclass SVM and softmax regression.

When tweets from an unknown region or state arrive, the classification pipeline consists of preprocessing, feature extraction via sparse coding (without labels), and max pooling. The resulting pooled sparse codes of the tweets are applied to the trained classifier from step 2 to predict the class label.

Enhancements

PCA whitening. Principal components analysis (PCA) is a dimensionality reduction technique that can speed up unsupervised learning such as sparse coding. A similar procedure called whitening is also helpful by making input less redundant and thus sparse coding more effective in classification. We precondition n input patches taken from $\mathbf{w}_{BW}$, $\mathbf{w}_{WC}$, and $\mathbf{w}_{WS}$ for sparse coding by PCA whitening:

1. Remove patch mean $\mathbf{x}^{(i)} := \mathbf{x}^{(i)} - \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}^{(i)}$;
2. Compute covariance matrix $\mathbf{C} = \frac{1}{n-1}\sum_{i=1}^{n} \mathbf{x}^{(i)}\mathbf{x}^{(i)\top}$;
3. Do PCA `[`$\mathbf{U}$`,` $\mathbf{\Lambda}$`] = eig(`$\mathbf{C}$`)`;
4. Compute $\mathbf{x}_{\text{PCAwhite}} = (\mathbf{\Lambda} + \epsilon\mathbf{I})^{-1/2}\mathbf{U}^\top\mathbf{x}$, where ϵ is a small positive value for regularization.

Regional dictionary learning (regD). In principle, dictionary learning is an unsupervised process. However, label information of training examples can be used beneficially

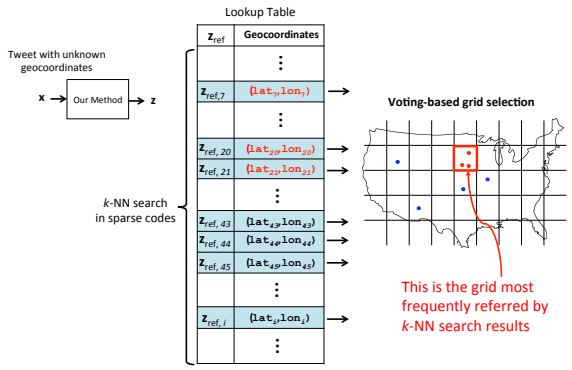

Figure 1: Voting-based grid selection scheme

to train dictionary. We consider the following enhancement. Using regional labels of training examples, we learn separate dictionaries D_1, D_2, D_3, and D_4 for the four US regions, where each D_i is trained using data from the respective region. Simple concatenation forms the final dictionary $D = [D_1 \| D_2 \| D_3 \| D_4]$. Thus, the resulting basis vectors are more representative for their respective regions.

Raw feature augmentation. Sparse code $\mathbf{y}$ serves as the feature for raw patch input $\mathbf{x}$. Information-theoretically speaking, $\mathbf{x}$ and $\mathbf{y}$ are approximately equivalent since one representation is a transform of the other. However, due to reasons such as insufficient samples for dictionary training or presence of noise in $\mathbf{x}$, there is a possibility of many-to-one ambiguity. In other words, raw data belonging to different classes may have very similar sparse code. To mitigate this problem, we consider a simple enhancement by appending the raw patch vector $\mathbf{x}$ to its sparse code (or, pooled sparse code $\mathbf{z}$) and use the combined vector as the augmented feature.

Voting-based grid selection. Figure 1 illustrates our voting-based grid selection scheme. With patches with known geocoordinates, we compute their pooled sparse codes and construct a lookup table with their corresponding geocoordinates. When a patch from unknown geocoordinates arrives, we compute its pooled sparse code, and find the k pooled sparse codes from the lookup table that are closest in cosine similarity. Each k-NN casts a vote to its corresponding grid. We identify the grid which receives the most votes and take the average of geocoordinates in the selected grid.

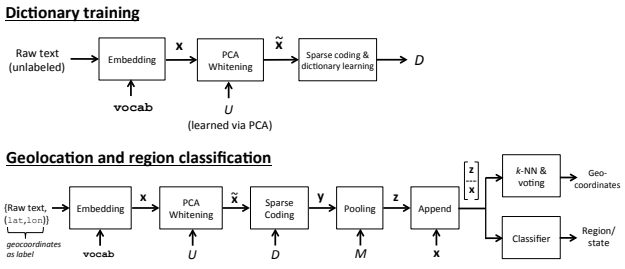

Figure 2: Full system pipeline

Full system pipeline

In Figure 2, we present the full system pipeline for geolocation and classification via sparse coding.

Evaluation

We evaluate our baseline methods and enhancements empirically, using the GEOTEXT dataset. This section will present a comparative performance analysis against other text-based geolocation approaches of significant impact.

Experiments

Dataset processing. For each test instance, we cut the dataset into five folds such that `fold=user_id%5`, following Eisenstein *et al.* (2010). Folds 1–4 are used for training, and fold 5 for testing. We have converted the entire text data for each user to binary (BW), word-counts (WC), and word-sequence (WS) vectors. From these vectors, we sparse code patches of a configurable size N taken from the input. We use patch sizes of $N = 32, 64$ with small or no overlap for feature extraction.

Training. In unsupervised learning, we precondition patches with PCA whitening prior to sparse coding. The parameters for sparse coding are experimentally determined. We have used a dictionary size $K \approx 10N$, sparsity $0.1N \leq T \leq 0.4N$ (T is number of nonzero elements in sparse code $\mathbf{y}$). For max pooling, we use pooling factors M in 10s.

In supervised learning, we have trained linear multiclass SVM and softmax classifiers, using max pooled sparse codes as features. In our results, we report the best average performance of the two. We have exhaustively built the lookup table of reference geocoordinates for k-NN. This results up to 7,500 entries (depending on how much of labeled dataset is allowed for supervised learning). In reducing prediction errors, we find k-NN works well for $20 \leq k \leq 30$. We use higher k ($200 \leq k \leq 250$) with voting-based grid selection scheme. We have varied grid sizes suggested by Roller *et al.* (2012) and decided on 5° for most purposes.

Metrics. For geolocation, we use the mean and median distance errors between the predicted and ground-truth geocoordinates in kilometers. We note that it is required to approximate the great-circle distance between any two locations on the surface of earth. We use the Haversine formula (1984). For classification tasks, we adopt multiclass classification accuracy as the evaluation metric.

Results and discussion

Geolocation. In Table 1, we present the geolocation errors of our baseline method and enhancements. We call our baseline method "SC" (sparse coding). Note that "Raw" means k-NN search based on vectors $\mathbf{w}_{BW}$ and $\mathbf{w}_{WC}$. We omit Raw for word-sequence embedding $\mathbf{w}_{WS}$ because there is no logic in comparing word-sequence vectors of two different documents. For our baseline with each enhancement, we have "SC+PCA" (PCA whitened patches), "SC+regD" (regional dictionary learning for sparse coding), "SC+Raw" (raw feature augmentation), "SC+voting" (voting-based grid selection), and "SC+all" (combining all enhancements).

Table 1: Geolocation errors. The values are mean (median) in km.

	Raw	SC	SC+PCA	SC+regD	SC+Raw	SC+voting	SC+all
Binary	1024 (632)	879 (722)	748 (596)	846 (701)	861 (713)	825 (529)	707 (489)
Word counts	1189 (1087)	1042 (887)	969 (802)	1020 (843)	1022 (863)	998 (511)	926 (497)
Word sequence	–	767 (615)	706 (583)	735 (596)	671 (483)	715 (580)	**581 (425)**

Table 2: Performance comparison summary

	Geolocation error		Classification accuracy	
	Mean	Median	Region	State
Our approach	**581**	**425**	**67%**	**41%**
Eisenstein *et al.*	845	501	58%	27%
W&B	967	479	–	–
Roller *et al.*	897	432	–	–

Overall, our methods applied to word-sequence vectors achieve the best performance. PCA whitening substantially improves the baseline (SC), especially on patches taken from binary embedding. We have appended binary embedding for SC+Raw on patches from word sequence vector and achieved the most impressive performance gain. Each of these enhancements individually has decreased geolocation errors for all embedding schemes. When all enhancements are applied, the combined scheme achieves the best results for each embedding scheme.

Figure 3 depicts the general trend that sparse coding outperforms Raw as a function of the amount of words in the labeled document. We observe the general trend that as more labeled datasets are available, geolocation error decreases for both Raw and sparse coding. When labeled dataset is limited, the result for sparse coding with $N = 64$ is particularly striking, showing decreased geolocation errors by approximately 100 km compared to Raw, throughout all reported amounts of labeled data. Note that SC (w/ $N = 64$) at 2×10^6 labeled data performs about the same as Raw at 3.7×10^6. As the amount of labeled training samples is limited in practice, such advantage of sparse coding is attractive.

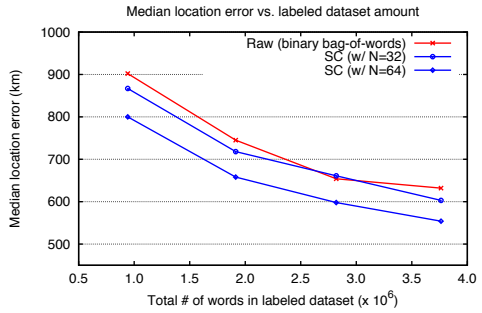

Figure 3: Median location error vs. size of labeled dataset.

Performance comparison. Table 2 presents a summary that compares the geolocation and regional classification performances of our approach and the previous work. We have achieved a 9% gain for region classification over Eisenstein *et al.*, and a 14% gain for state classification.

Conclusion and Future Work

We have shown that twitter geolocation and regional classification can benefit from unsupervised, data-driven approaches such as sparse coding and dictionary learning. Such approaches are particularly suited to microblog scenarios where labeled data can be limited. However, a straightforward application for sparse coding would only produce suboptimal solutions. As we have demonstrated, competitive performance is attainable only when all of our enhancement steps are applied. In particular, we find that raw feature augmentation and an algorithmic enhancement by voting-based grid selection have been significant in reducing errors. To the best of our knowledge, the use of sparse coding in text-based geolocation and the proposed enhancements are novel. Our future work includes a hybrid method that can leverage the strengths of data-driven and model-based approaches and its evaluation in both US and worldwide data.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE1144152 and gifts from the Intel Corporation.

References

Aharon, M.; Elad, M.; and Bruckstein, A. 2006. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Trans. on Signal Processing*.

Chen, S. S.; Donoho, D. L.; and Saunders, M. A. 2001. Atomic Decomposition by Basis Pursuit. *SIAM Rev.*

Eisenstein, J.; O'Connor, B.; Smith, N. A.; and Xing, E. P. 2010. A Latent Variable Model for Geographic Lexical Variation. In *EMNLP*.

GeoText. 2010. CMU Geo-tagged Microblog Corpus. `http://www.ark.cs.cmu.edu/GeoText/`.

Hong, L.; Ahmed, A.; Gurumurthy, S.; Smola, A. J.; and Tsioutsiouliklis, K. 2012. Discovering Geographical Topics in the Twitter Stream. In *WWW*.

Olshausen, B. A., and Field, D. J. 1997. Sparse Coding with an Overcomplete Basis Set: Strategy Employed by V1? *Vision research*.

Roller, S.; Speriosu, M.; Rallapalli, S.; Wing, B.; and Baldridge, J. 2012. Supervised Text-based Geolocation Using Language Models on an Adaptive Grid. In *EMNLP*.

Sinnott, R. W. 1984. Virtues of the Haversine. *Sky and Telescope*.

Tibshirani, R. 1994. Regression Shrinkage and Selection via Lasso. *Journ. of Royal Statistical Society*.

Tropp, J., and Gilbert, A. 2007. Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit. *IEEE Trans. on Information Theory*.

Wing, B. P., and Baldridge, J. 2011. Simple Supervised Document Geolocation with Geodesic Grids. In *ACL*.