Comparing Feature Vectors for Sentiment Analysis of Short Texts

Mitch Kinney

University of Minnesota

December 5, 2017

Analyzing non-numerical data

When dealing with non-numerical data such as text or images a common method to do classification is:

- Create a numerical representation of the data
- Use existing methods to build a model

Much focus of today's research is on finding good combinations.

In text:

- Word2vec with a Convolutional Neural Network (Santos and Gatti, 2014)
- Bag-of-words with Naive Bayes (Go et al., 2009)

In images:

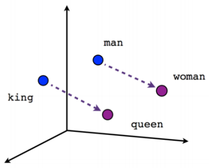
- Pixel brightness with Fast Fourier Transform (Rodriguez et al., 2008)
- Spatial Pyramid with Label Consistent Kernel-SVD (Jiang et al., 2011, Lazebnik et al., 2006)

Analyzing non-numerical data

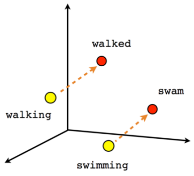
I am interested in a simpler combination as well as trying out doc2vec which is an extension of word2vec

- My project will use doc2vec and bag-of-words to create a numerical representation of short text data
- Then use support vector machines (SVM) with linear and rbf kernel to do classification

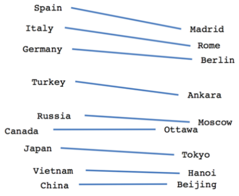
Explaining doc2vec



Male-Female



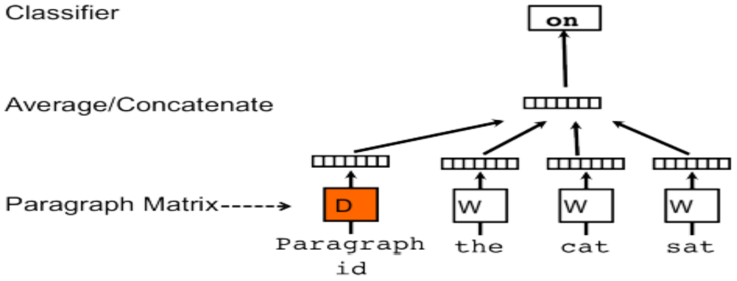
Verb tense



Country-Capital

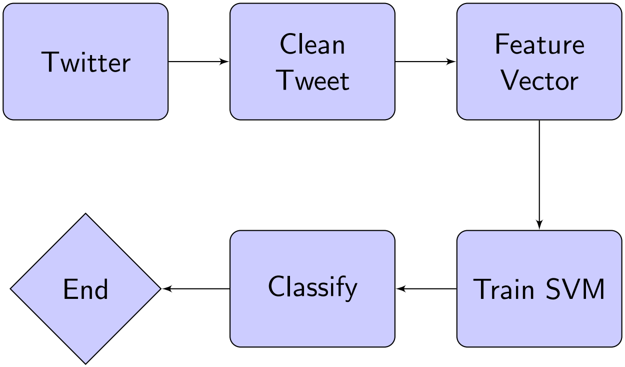
- Based on word2vec which finds semantic relationships between words
- Word2vec uses a neural network to build representations by attempting to predict words in a sentence given all other words
- Word2vec reference: (Mikolov et al., 2013)

Explaining doc2vec



- Shifting window to update both word and paragraph vector, but paragraph vector is kept through all windows to add context
- When inferring doc2vec vectors, word vectors are held constant while new paragraph vectors are added and trained
- Doc2vec reference: (Le et al., 2014)

Model flow chart



Dataset and libraries

STS (Stanford Twitter Sentiment corpus)

- 1.6 million tweets gathered from searching ":)" and ":(".
- Automatically labeled positive or negative based on whether a happy or sad face is attached to the tweet.
- I am able to use about 200,000 tweets

I will use Python as my coding language. And the main packages I will use is gensim which implements doc2vec and liblinear which implements linear SVM.

- liblinear refernce: Fan et al., 2008

Results

Initial results using 200,000 tweets with 70% training, 20% testing and 10% validation

Method	Accuracy
Doc2vec	76.01%
Bag-of-words	79.12%
SVD	65.89%

Recall state of the art is about $\sim$85%

Difficulties

I would like to try a form of Kernel PCA using the rbf kernel

- Need to solve an eigenvalue/eigenvector problem for a square matrix with dimension equal to sample size
- For short text data this needs to be large ($>$10,000 samples)

Why is bag-of-words beating doc2vec despite claims of superiority?

Left to do

Possibly can try an over complicated Kernel PCA. For each element in the kernel matrix

$$k_{i,j} = \phi(x_i)^T \phi(x_j)$$

where $x_i,\ x_j$ are the sample feature vectors.

- Kernel trick is never having to compute $\phi(x_i)$
- Possibly can manually compute $\phi(x_i)$

For instance if $x = (x_1, x_2)$ and $y = (y_1, y_2)$ and I'm using the rbf kernel (from TenaliRaman on Stack Exchange)...

$$\begin{aligned} k(x,y) &= \exp(||x-y||_2^2) \\ &= \exp(||x||_2^2)\exp(||y||_2^2)\exp(2x^T y) \\ &= \exp(||x||_2^2)\exp(||y||_2^2)\sum_{n=0}^{\infty}\frac{(2x^T y)^n}{n} \end{aligned}$$

Left to do

Another interesting thing to try is to see when doc2vec "tops out" for length of feature vector

- For instance keeping fixed all other tuning parameters, what is the minimum length that doc2vec will still do well
- When can we stop adding parameters?

Will doc2vec work better than bag-of-words for documents with longer text?

- Are tweets too short for doc2vec to capture any relationship among the words?
- In a previous talk authors used the news20 dataset which were very long texts so I would like to compare the two again for this dataset

Thank you!