Document Classification by Inversion of Distributed Language Representations

Matt Taddy
University of Chicago Booth School of Business
taddy@chicagobooth.edu

Abstract

There have been many recent advances in the structure and measurement of *distributed* language models: those that map from words to a vector-space that is rich in information about word choice and composition. This vector-space is the distributed language representation.

Given the success of such approaches, researchers have proposed models and algorithms to adapt them for use in document classification; e.g., for predicting sentiment. The goal of this note is to point out that any distributed representation can be turned into a classifier through inversion via Bayes rule. The approach is simple and modular, in that it will work with any language representation whose training can be formulated as optimizing a probability model. In our application to 2 million sentences from Yelp reviews, we also find that it performs as well as or better than complex purpose-built algorithms.

1 Introduction

Distributed, or vector-space, language representations consist of a location for every vocabulary *word* in $\mathbb{R}^d$, where d is the dimension of the latent representation space. These locations are learned to optimize (or approximately optimize) an objective function defined on the original text, such as a likelihood for word occurences.

A popular example is the word2vec machinery of Mikolov et al. (2013). This trains the distributed representation to be useful as an input layer for prediction of words from their neighbors in a Skip-gram likelihood; that is, to maximize

$$\prod_{k \neq t,\ k=t-b}^{t+b} \log \mathrm{p}(w_k \mid w_t) \qquad (1)$$

summed across all words w_t, where b is a window (possibly truncated by the end or beginning of the sentence) and the function $\mathrm{p}(w_k \mid w_t)$ is a neural network classifier that depends upon vector-space representations for w_k and w_t.

2 word2vec

The skip-gram probabalistic language model is trained to, for window b, maximize for each word

This probability is calculated for all possible word pairs over a sentence (some defined short chunk of language).

In the word2vec formulation, each probability is represented as

$$\mathrm{p}(w \mid w_I) = \prod_{j=1}^{L(w)-1} \sigma\left(\mathrm{ch}\left[\eta(w, j+1)\right] \mathbf{u}_{\eta(w,j)}^{\top} \mathbf{v}_{w_I}\right)$$

where * $\eta(w,i)$ is the i^{th} node in a binary huffman tree representation (path) for word w, * $\sigma(x) = 1/(1+\exp[-x])$, * and $\mathrm{ch}(\eta) \in \{-1,+1\}$ translates from left/right child to +/- one.

The binary huffman tree represents each word as a series of bits, so that the probability of word w given word w_I can be written as the product of probabilities that each bit is either on or off (represented above through the ch function).

Given a fitted representation, we can score any new sentence (and sum across them for documents) according to the model *implied* by the training process. This gives a document (log) probability under that representation. This can be calculated for language representations trained on each of the corpora associated with some class labels. When combined with priors for each class label and inversion via Bayes rule, the document probabilities provide class probabilities.

The doc2vec tool maps from documents to a vector space of fixed dimension, which can then be used as input to off-the-shelf machine learners. The latter builds the sentiment/meaning into the model itself, and conditions upon this information during the training process.

Advantages of the inversion framework include: * modularity: it works for any model of language that can (or its training can) be interpreted as a probabilistic model. * transparency and replicability: a simple extension of any software for training distributed language models. * performance

3 Example: Yelp Reviews

> "the food is always good i would highly recommend this place"

4 Discussion

Engineering appeal: companies already fit w2v or some other distributed representation. These could be done on a per-class basis instead, with prediction (e.g., of the next word) results averaged when the user class is unknown. The current note makes it clear that such a system could act as the basis for document classification, as well as NLP[1]

References

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

[1]Training separate distributed representations for each class label has the separtate benefit of allowing computation to be partitioned across such labels, and thus fits with recent Big Data schemes such as EBF.

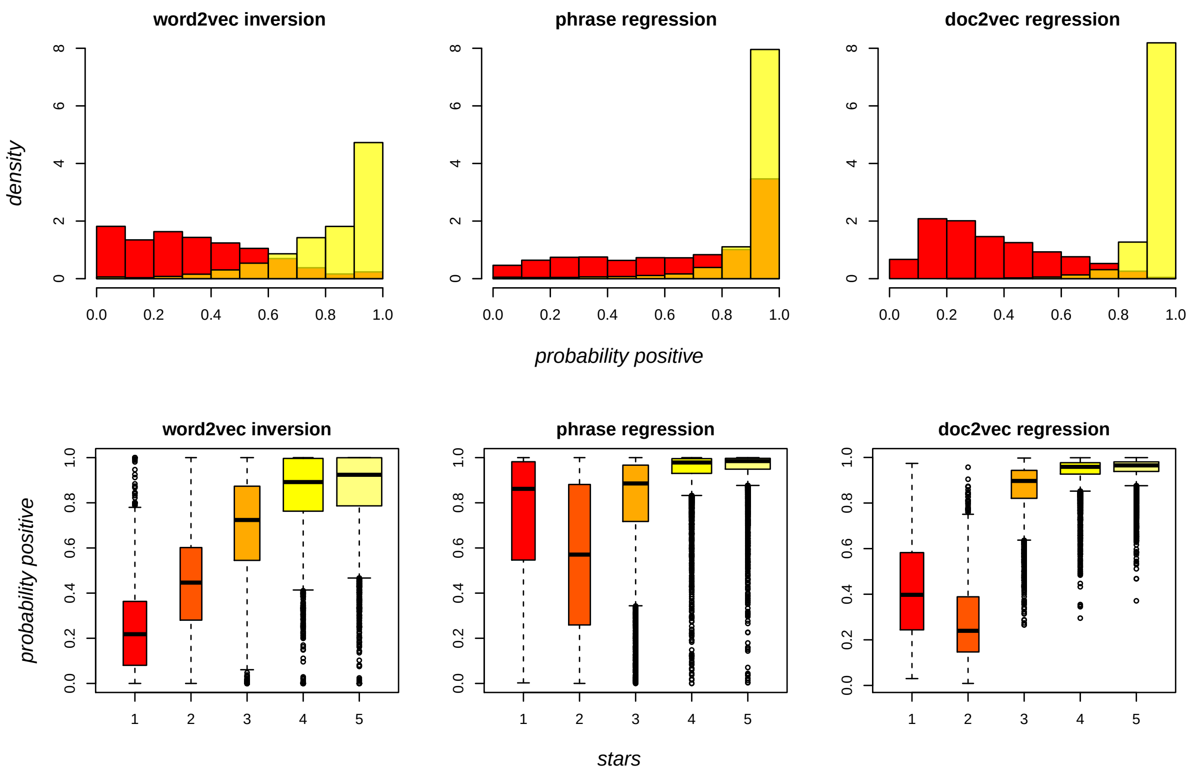

Figure 1: Out-of-Sample fitted probabilities of a review having greater than 2 stars. In the top histograms, red (dark) are the probabilities for true negative reviews and yellow (light) are for true positives.

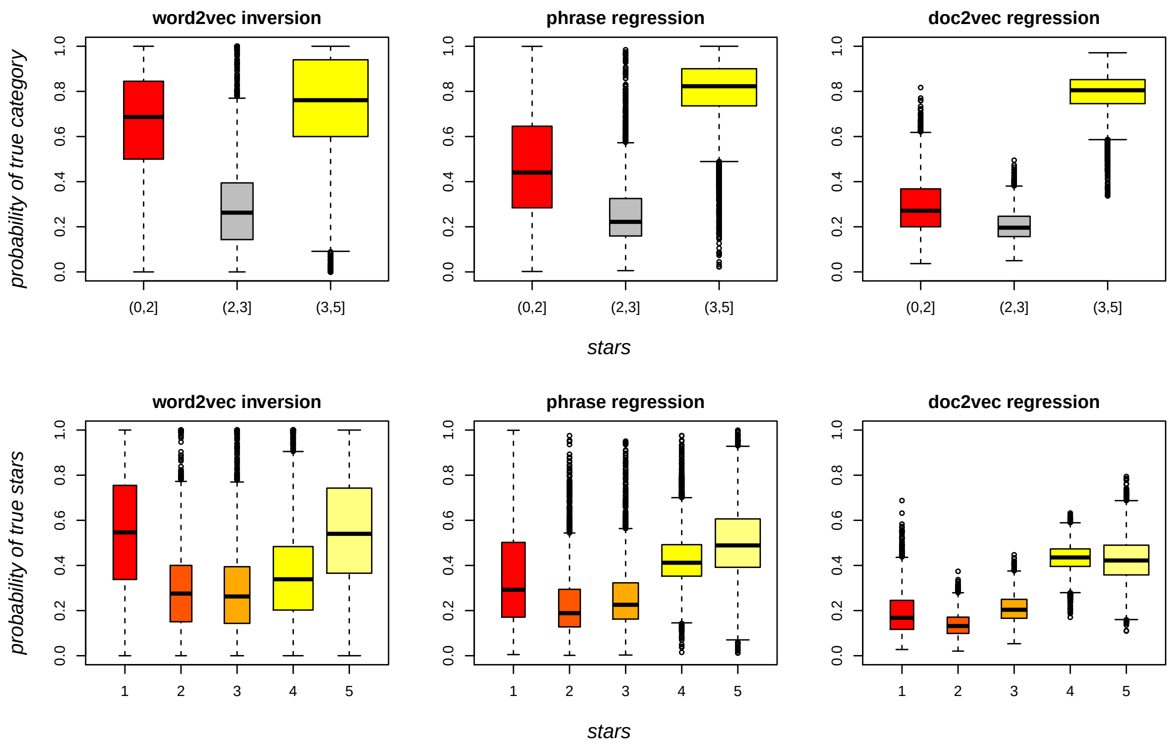

Figure 2: Out-of-Sample fitted probabilities for observed truth. In the top plot, we are predicting Negative (≤ 2), Neutral (3), or Positive (≥ 4). In the bottom, we are predicting each of the separate 5 star ratings.