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.

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 $\mathcal{V}$ consist of a location for every vocabulary *word* in $\mathbb{R}^K$, where K 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 occurrences.

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

$$\sum_{k\neq t,\ j=t-b}^{t+b} \log \mathrm{p}_{\mathcal{V}}(w_{sj} \mid w_{st}) \tag{1}$$

summed across all words w_{st} in all sentences $\mathbf{w}_s$, where b is the skip-gram window (truncated by the ends of the sentence) and $\mathrm{p}_{\mathcal{V}}(w_{sj}|w_{st})$ is a neural network classifier that takes vector representations for w_{st} and w_{sj} as input (see Section 2).

Distributed language representations have been studied since the early work on neural networks (Rumelhart et al., 1986) and have long been applied in natural language processing (Morin and Bengio, 2005). The models are generating much recent interest due to the large performance gains from the newer systems, including Wor2Vec and the Glove model of Pennington et al. (2014), observed in tasks such as word prediction, word analogy identification, and named entity recognition.

Given the success of these new models, researchers have begun searching for ways to adapt the representations for use in document classification tasks such as sentiment prediction or author identification. One naive approach is be to use aggregated word vectors across a document (e.g., a document's average word-vector location) as input to a standard classifier (e.g., logistic regression). However, a document is actually an *ordered* path of locations through $\mathbb{R}^K$, and simple averaging destroys much of the available information.

More sophisticated aggregation is proposed in Socher et al. (2011; 2013), where recursive neural networks are used to combine the word vectors through the estimated parse tree for each sentence. Alternatively, Le and Mikolov's Doc2Vec (2014) adds document labels to the conditioning set in (1) and has them influence the skip-gram likelihood through a latent input vector location in $\mathcal{V}$. In each case, the end product is a distributed representation for every sentence (or document for Doc2Vec) that can be used as input to a generic classifier.

1.1 Bayesian Inversion

These approaches all add considerable model and estimation complexity to the original underlying distributed representation. We are proposing a simple alternative that turns fitted distributed language representations into document classifiers without any additional modeling or estimation.

Write the probability model that the representation $\mathcal{V}$ has been trained to optimize (likelihood maximize) as $\mathrm{p}_{\mathcal{V}}(d)$, where document $d = \{\mathbf{w}_1, ...\mathbf{w}_S\}$ is a set of sentences – ordered vectors of word identities. For example, in Word2Vec the skip-gram likelihood in (1) yields

$$\log \mathrm{p}_{\mathcal{V}}(d) = \sum_s \sum_t \sum_{k \neq t,\, j=t-b}^{t+b} \log \mathrm{p}_{\mathcal{V}_y}(w_{sj} \mid w_{st}). \tag{2}$$

Even when such a likelihood is not explicit it will be implied by the objective function that is optimized during training.

Now suppose that your training documents are grouped by class label, $y \in \{1 \ldots C\}$. We can train *separate* distributed language representations for each set of documents as partitioned by y; for example, fit Word2Vec independently on each sub-corpus $D_c = \{d_i : y_i = c\}$ and obtain the labeled distributed representation map $\mathcal{V}_c$. A new document d has probability $\mathrm{p}_{\mathcal{V}_c}(d)$ if we treat it as a member of class c, and Bayes rule implies

$$\mathrm{p}(y|d) = \frac{\mathrm{p}_{\mathcal{V}_y}(d)\pi_y}{\sum_c \mathrm{p}_{\mathcal{V}_c}(d)\pi_c} \tag{3}$$

where π_c is our prior probability on class label c.

Thus distributed language representations trained separately for each class label yield directly a document classification rule via (3). This approach has a number of attractive qualities.

Simplicity: The inversion strategy works for any model of language that can (or its training can) be interpreted as a probabilistic model. This makes for easy implementation in systems that are already engineered to fit such language representations. The strategy is also interpretable: whatever intuition one has about the distributed language model can be applied directly to the inversion-based classification rule. Finally, inversion adds a plausible model for reader understanding on top of any given language representation.

Scalability: when working with massive corpora it is often useful to split the data into blocks as part of distributed computing strategies. Our model of classification via inversion provides a convenient top-level partitioning of the data. An efficient system could fit separate by-class language representations, which will provide for document classification as in this article as well as class-specific answers for NLP tasks such as word prediction or analogy. When one wishes to treat a document as unlabeled, NLP tasks can be answered through ensemble aggregation of the class-specific answers.

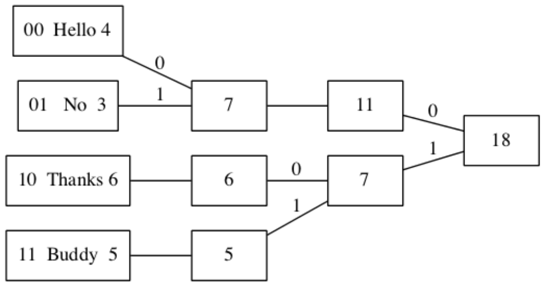

Figure 1: Binary Huffman encoding of a 4 word vocabulary, based upon 18 total utterances.

Performance: We find that, in our examples, inversion of Word2Vec yields lower misclassification rates than both Doc2Vec-based classification and the multinomial inverse regression (MNIR) of Taddy (2013a). We did not anticipate such outright performance gain. Moreover, we expect that with careful calibration of the *many* various tuning parameters available when fitting both Word and Doc 2Vec the performance results will change. Indeed, we find that all methods are often outperformed by phrase-count logistic regression with rare-feature up-weighting and carefully chosen regularization. However, the completely untuned out-of-the-box performance of Word2Vec inversion argues for its consideration as a simple default in document classification.

In the remainder, we outline classification through inversion of a specific Word2Vec model and illustrate of the ideas in classification of Yelp reviews. The implementation requires only a small extension of the popular `gensim` python library (Řehůřek and Sojka, 2010); the extended library as well as code to reproduce all of the results in this paper are available on `github`. In addition, the yelp data is publicly available as part of the correspond data mining contest at `kaggle.com`. See `github.com/taddylab/deepir` for detail.

2 Implementation

Word2Vec trains $\mathcal{V}$ to maximize the skip-gram likelihood based on (1). We work with the Huffman softmax specification (Mikolov et al., 2013), which includes a pre-processing step to encode each vocabulary word in its representation via a binary Huffman tree (see Figure 1).

Each individual probability is then

$$\mathrm{p}_{\mathcal{V}}(w|w_t) = \prod_{j=1}^{L(w)-1} \sigma\Big(\mathrm{ch}\left[\eta(w,j+1)\right] \mathbf{u}_{\eta(w,j)}^{\top} \mathbf{v}_{w_t}\Big) \tag{4}$$

where $\eta(w,i)$ is the i^{th} node in the Huffman tree path, of length $L(w)$, for word w; $\sigma(x) = 1/(1+\exp[-x])$; and $\mathrm{ch}(\eta) \in \{-1,+1\}$ translates from whether η is a left or right child to +/- 1. Every word thus has both input and output vector coordinates, $\mathbf{v}_w$ and $[\mathbf{u}_{\eta(w,1)} \cdots \mathbf{u}_{\eta(w,L(w))}]$. Typically, only the input space $\mathbf{V} = [\mathbf{v}_{w_1} \cdots \mathbf{v}_{w_p}]$, for a p-word vocabulary, is reported as the language representation – these vectors are used as input for NLP tasks. However, the full representation $\mathcal{V}$ includes mapping from each word to both $\mathbf{V}$ and $\mathbf{U}$.

We apply the `gensim` python implementation of Word2Vec, which fits the model via stochastic gradient descent (SGD), under default specification. This includes a vector space of dimension $K = 100$ and a skip-gram window of size $b = 5$.

2.1 Word2Vec Inversion

Given Word2Vec trained on each of C class-specific corpora $D_1 \ldots D_C$, leading to C distinct language representations $\mathcal{V}_1 \ldots \mathcal{V}_C$, classification for new documents is straightforward. Consider the S-sentence document d: each sentence $\mathbf{w}_s$ is given a probability under each representation $\mathcal{V}_c$ by applying the calculations in (1) and (4). This leads to the $S \times C$ matrix of sentence probabilites, $\mathrm{p}_{\mathcal{V}_c}(\mathbf{w}_s)$, and document probabilities are obtained

$$\mathrm{p}_{\mathcal{V}_c}(d) = \frac{1}{S}\sum_s \mathrm{p}_{\mathcal{V}_c}(\mathbf{w}_s). \tag{5}$$

Finally, class probabilities are calculated via Bayes rule as in (3). We use priors $\pi_c = 1/C$, so that classification proceeds by assigning the class

$$\hat{y} = \mathrm{argmax}_c \ \mathrm{p}_{\mathcal{V}_c}(d). \tag{6}$$

3 Illustration

We consider a corpus of reviews provided by Yelp for a contest on `kaggle.com`. The text is tokenized simply by converting to lowercase before splitting on punctuation and white-space. The training data are 230,000 reviews containing more than 2 million sentences. Each review is marked by a number of *stars*, from 1 to 5, and we fit separate Word2Vec representations $\mathcal{V}_1 \ldots \mathcal{V}_5$ for the documents at each star rating. The validation data consist of 23,000 reviews, and we apply the inversion technique of Section 2 to score each validation document d with class probabilities $\mathbf{q} = [q_1 \cdots q_5]$, where $q_c = \mathrm{p}(c|d)$.

The probabilities will be used in three different classification tasks; for reviews as

a. negative at 1-2 stars, or positive at 3-5 stars;

b. negative 1-2, neutral 3, or positive 4-5 stars;

c. corresponding to each of 1 to 5 stars.

In each case, classification proceeds but summing across the relevant sub-class probabilities. For example, in task a, $\mathrm{p}(\texttt{positive}) = q_3 + q_4 + q_5$. Note that the same five fitted Word2Vec representations are used for each task.

We consider three comparator techniques.

Doc2Vec is also fit via `gensim`, using the same latent space specification as for Word2Vec: $K = 100$ and $b = 5$. Le and Mikolov provide two alternative Doc2Vec specifications: distributed memory (DM) and distributed bag-of-words (DBOW). We fit both. Vector representations for validation documents are trained without updating the word-vector elements, leading to 100 dimensional vectors for each document for each of DM and DCBOW. We input each, as well as the combined 200 dimensional DM+DBOW representation, to separate maximum likelihood logistic regressions.

MNIR, the multinomial inverse regression of Taddy (2013a; 2014a), is applied as implemented in the `textir` package for R. MNIR maps from text to the class-space of interest through a multinomial logistic regression of word (or phrase) counts onto variables relevant to the class-space. Here, we regress word counts onto stars expressed numerically and as a 5-dimensional indicator vector, leading to a 6-feature multinomial logistic regression. The MNIR procedure then uses the $6 \times p$ matrix of feature-word regression coefficients to map from word-count to feature space, resulting in 6 dimensional 'sufficient reduction' statistics for each document. These statistics are then input to maximum likelihood logistic regressions.

Phrase regression applies logistic regression of the response classes directly onto word or phrase counts. The phrases are obtained using `gensim`'s phrase builder, which simply combines highly probable pairings; for example, it yields `first_date` and `chicken_wing` in this corpus (these phrases are also used as the bag-of-

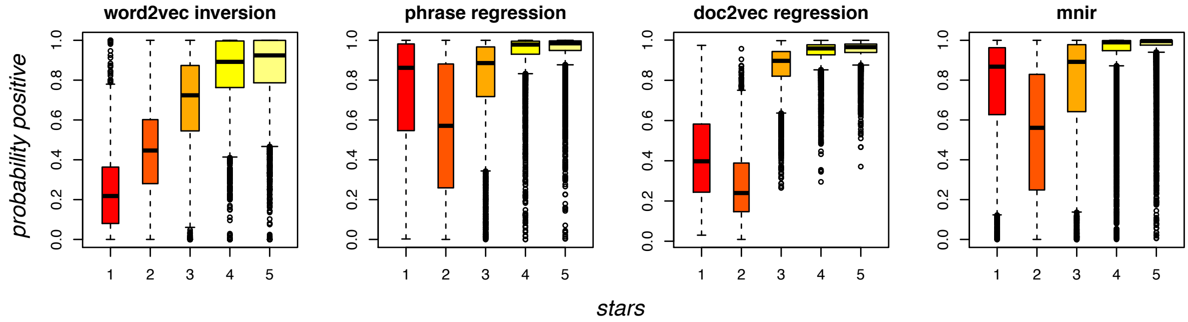

Figure 2: Out-of-Sample fitted probabilities of a review having greater than 2 stars.

	a (NP)	b (NNP)	c (1-5)
W2V inversion	.099	**.189**	.439
Phrase regression	**.084**	.200	**.410**
D2V DBOW	.144	.282	.496
D2V DM	.179	.306	.549
D2V combined	.148	. 284	.5
MNIR	.095	.254	.508

Table 1: Out-of-sample misclassification rates.

words input to MNIR). The logistic regressions are then fit under L_1 regularization with the penalties weighted by phrase-count standard deviation (feature scaling) and selected according to the corrected AICc criteria (Flynn et al., 2013), using the `gamlr` R package of Taddy (2014b). For multi-class tasks $b-c$, we use `gamlr` in distribution via the `distrom` R package of Taddy (2014a).

3.1 Results

Misclassification rates for each task on the validation set are reported in Table 1. Simple phrase-count regression is consistently the strongest performer, bested only by Word2Vec inversion on task b. This is partially due to the relative strengths of discriminative (e.g., logistic regression) vs generative (e.g., all others here) classifiers: given a large amount of training text, asymptotic efficiency of logistic regression will start to work in its favor over the finite sample advantages of generative discrimination (Ng and Jordan, 2002; Taddy, 2013b). However, the comparison is also unfair to Word2Vec and Doc2Vec: both phrase regression and MNIR are optimized exactly under AICc selected penalty, while Word and Doc 2Vec have only been approximately optimized under a single specification. The distributed representations should improve with some careful engineering.

Word2Vec inversion outperforms the other generative classifier alternatives (except, by a narrow margin, MNIR in task a). Doc2Vec under DBOW specification and MNIR both do worse, but not by a large margin. Their similar performance might be expected: both have a version of logistic (or softmax) regression for word-counts as their first step. MNIR regresses the words onto observed class information, while Doc2Vec DBOW regresses onto a latent vector space. In contrast to Le and Mikolov, we find here that the Doc2Vec DM model does much worse than DBOW.

Looking at the fitted probabilities in detail we see that Word2Vec provides a more useful document *ranking* than any comparator (including phrase regression). For example, Figure 2 shows the probabilities of a review being 'positive' in task a as a function of the true star rating for each validation review. Although phrase regression does slightly better in terms of misclassification rate, it does so at the cost of classifying many terrible (1 star) reviews as positive. Word2Vec inversion is the *only* method that yields positive-document probabilities that are clearly increasing in distribution with the true star rating. It is not difficult to envision a misclassification cost structure that favors such nicely ordered probabilities.

4 Discussion

The goal of this note is to point out inversion as an option for turning distributed language representations into classification rules. We are not arguing for the supremacy of Word2Vec inversion in particular, and the approach will work with alternative representations. Moreover, we are not even arguing that it will always outperform purpose-built classification tools. However, it is a simple, scalable, interpretable, and effective option for classification whenever you are working with such distributed representations.

References

Cheryl Flynn, Clifford Hurvich, and Jefferey Simonoff. 2013. Efficiency for Regularization Parameter Selection in Penalized Likelihood Estimation of Misspecified Models. *Journal of the American Statistical Association*, 108:1031–1043.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31 st International Conference on Machine Learning*.

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.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252.

Andrew Y. Ng and Michael I. Jordan. 2002. On Discriminative vs Generative Classifiers: A Comparison of Logistic Regression and naive Bayes. In *Advances in Neural Information Processing Systems (NIPS)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

David Rumelhart, Geoffrey Hinton, and Ronald Williams. 1986. Learning representations by backpropagating errors. *Nature*, 323:533–536.

Richard Socher, Cliff C. Lin, Chris Manning, and Andrew Y. Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642.

Matt Taddy. 2013a. Multinomial Inverse Regression for Text Analysis. *Journal of the American Statistical Association*, 108:755–770.

Matt Taddy. 2013b. Rejoinder: Efficiency and structure in MNIR. *Journal of the American Statistical Association*, 108:772–774.

Matt Taddy. 2014a. Distributed Multinomial Regression. arXiv:1311.6139.

Matt Taddy. 2014b. One-step estimator paths for concave regularization. arXiv:1308.5623.