Distinguish Polarity in Bag-of-Words Visualization

Yusheng Xie,[1,2] **Zhengzhang Chen,**[2] **Ankit Agrawal,**[2] **Alok Choudhary**[2]
[1]Baidu Research, Sunnyvale, CA USA
[2]Northwestern University, Evanston, IL USA
[1]xieyusheng@baidu.com
[2]{yxi389,zzc472,ankitag,choudhar}@eecs.northwestern.edu

Abstract

Neural network-based BOW models reveal that word-embedding vectors encode strong semantic regularities. However, such models are insensitive to word polarity. We show that, coupled with simple information such as word spellings, word-embedding vectors can preserve both semantic regularity and conceptual polarity without supervision. We then describe a nontrivial modification to the t-distributed stochastic neighbor embedding (t-SNE) algorithm that visualizes these semantic- and polarity-preserving vectors in reduced dimensions. On a real Facebook corpus, our experiments show significant improvement in t-SNE visualization as a result of the proposed modification.

Introduction

The word "word" refers to the smallest linguistic unit that conveys nontrivial semantic meaning. In English, *word* is the bridge from 26 simple letters to everything readable: literature, news, tweets, etc. Learning word meaning is fundamental to understanding language for both human and computer. Bag-of-words (BOW) models become widely popular thanks to their simple nature and good performance. For example, latent Dirichlet allocation (Blei, Ng, and Jordan 2003) and tf-idf (Manning and Schütze 1999) rely on the BOW representation.

In spite of its practical success, BOW has received two major criticisms. First, simple BOW models ignore word semantics. Words like *hotel*, *hotels*, and *motel* are just different words with no connections. Second, vocabulary size dictates the complexity of a BOW model. Unfortunately, vocabulary size usually grows really fast with respect to text corpus size. For example, the Brown corpus (Francis and Kucera 1979) contains 1 million English words from a 50,000 vocabulary. But the 1-trillion-word Google Web n-gram English corpus has a vocabulary greater than 13 million (Lin et al. 2012). In addition to data sparsity, it is impractical for most machine learning algorithms to handle 13 million feature dimensions.

BOW model has received many improvements addressing the two issues mentioned above. Distributed vector representation for words (Bengio et al. 2000; Morin and Bengio 2005; Mikolov et al. 2013; Le and Mikolov 2014) and semantic hashing (Salakhutdinov and Hinton 2009) effectively address model's semantic ignorance as well as increasing dimensionality. In distributed vector representation (i.e., word-embedding vectors), each word in the vocabulary W is transformed into a real vector $w \in \mathcal{R}^d$. the dimension of this representation, d, grows only logarithmically with respect to the size of vocabulary $|W|$. Even for very large web-scale text corpus, $d = 200$ is typical and effective (Mikolov et al. 2013). As a result, distributed representation can encode word semantics as (cosine) similarity between the vectors and does not suffer from exceedingly high dimensionality.

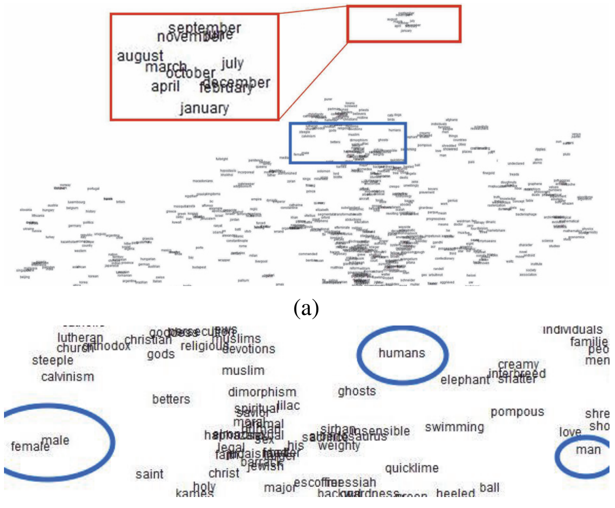


Figure 1: *(a)* 2d t-SNE plots of word vectors. Clustered words for *months* are shown in zoomed-in details within red rectangles. *(b)* Detail of the blue rectangular area in (a): words in blue circles are related to *people*.

However, word polarity information is elusive in word-embedding representations. More specifically, The authors in (Nguyen, im Walde, and Vu 2016) recently reveal that unsupervised embedding vectors cannot represent global or local *polarity* via point-wise similarity (e.g., cosine similarity between word vectors w_a and w_b). We train a set of word vectors on standard Wikipedia corpus using the popular `word2vec` software[1]. It turns out that most opposite words have the highest cosine similarity (similarity value we show in parenthesis):

warm is most similar to *cold* (0.499);
healthy $\sim$ *unhealthy* (0.501);
male $\sim$ *female* (0.687);
summer $\sim$ *winter* (0.750);
thick $\sim$ *thin* (0.694);
obese $\sim$ *underweight* (0.590), etc.

In the bigger picture, Figure 1(a) visualizes word positions as how semantically similar the word is to other words around it. The map is generated using word vectors by t-SNE algorithm. It does a reasonably good job by clustering the words for months (*janurary*, *february*, etc.) into a tight neighborhood (shown in red) on that large canvas. But the visualization is less illustrative when we look at words with polarity. The circles in Figure 1(b) are some words loosely related to *people*. In the global vocabulary, *female* and *male* are close in meaning; but when we think about only the local vocabulary related to *people*, *female* and *male* have opposite meanings under this narrow vocabulary. *Female* and *male* are placed very closely because the two have very similar word vectors. Without polarity being represented in the vectors, algorithms like t-SNE cannot visualize any polarity in Figure 1(b).

Visualizing word polarity first depends on representing it. The possibility of representing word polarity in vector similarity is as good as BOW models' ability to capture polarity from context. BOW models are about context. Their inefficacy in learning word polarity reveals something about how humans learn languages. *Learning another language is not only learning different words for the same things, but learning another way to think about things.*[2] Certain words are better understood from, say, everyday experience than from verbal context. It is hard to explain *warm* without contrasting it with *cold* in real life events such as warm bath and cold ice cream. Though the pair of words have opposite meanings, we use them to describe the same group of nouns: weather, food, etc. They are even paired in figurative senses (e.g., a warm welcome and a cold greeting). Words with opposite polarity occur within very similar context, which misleads BOW to conclude that they are semantically "similar" words.

Our Contribution

We propose a polarity-integration fix to the t-SNE algorithm, a popular tool for visualizing neural network-trained BOW models. Based on simple word literal distances, our integration actually violates t-SNE's input requirement. Finally, we propose a concentration-regularized autoencoder to transform the word literal distances into t-SNE-compliant features.

Related Works

Nonlinear Dimension Reduction

Nonlinear dimension reduction algorithms, many of which are summarized in this survey book (Lee and Verleysen 2007), aim to preserve local data structure. Most earlier algorithms are good at visualizing complex structures in low dimensions (e.g., swiss roll structure in 2d or 3d) but less so at any structure in higher dimensions. It is a common challenge to scale such algorithms to practical, high-dimensional data such as images (e.g., MNIST[3] digits or human faces) while still preserving both local and global structures from the original data. In particular, t-distributed Stochastic Neighbor Embedding (t-SNE) has become popular due to its solutions to the aforementioned challenge.

t-SNE visualizes high-dimensional data points in low-dimensional (often 2d) maps (van der Maaten and Hinton 2008). As a general algorithm, different studies such as (Botha and Blunsom 2014) have used it to reduce feature vectors derived using different algorithms (convolutional neural networks, recurrent neural networks, etc.) from different types of data (text, images, etc.). The basic t-SNE algorithm also spurs various performance-enhanced versions in (van der Maaten 2014) and (van der Maaten 2013).

t-SNE first computes a pairwise distance matrix P from original data points. Then, the algorithm estimates a pairwise distance matrix Q in the reduced dimension space such that the distance between P and Q is minimized. A core technique in t-SNE is to use Student t distribution rather than Gaussian to compute the distance in Q. Having a longer tail than Gaussian, Student t distribution does not separate dissimilar data points overly distant from one another while keeps similar datapoints close enough.

A caveat in practicing t-SNE requires that matrix P fits an assumed distribution. It is a vulnerable requirement to guard if input data points have heterogenous dimensions. Later in this paper, we describe our solution for embedding/visualizing polarity in word vectors and why it would create heterogenous dimensions for t-SNE and how we solve that violation of t-SNE's requirement.

Word Embedding

Mentioned in introduction, distributed vector representation (i.e., word-embedding vectors) is a valuable addition to BOW models. The main difficulty in implementing distributed vector representation for words is learning the vectors, which depends on selecting a good objective function and a fast training algorithm. When the modern idea of training word vectors with neural networks first came up in (Bengio et al. 2000), the training is painfully slow and has improved over the decade. Recent works like (Goldberg and Levy 2014) and (Levy and Goldberg 2014) further point out that a (empirically) good objective is to maximize the probability of context words over the entire text corpus. Formally, the learning goal is:

$$\arg\max_{\theta} \prod_{w \in \text{corpus}} \prod_{u \in U(w)} p(u|w;\theta), \tag{1}$$

where $U(w)$ is the set of context words around w. Variations and regularized versions of Equation 1 exist. For example, some consider the context $U(w)$ as ordered n-grams instead of orderless bag-of-words (bag-of-words context is much

[1]detailed in our experiments.

[2]by Flora Lewis, American journalist

[3]http://yann.lecun.com/exdb/mnist/

faster to train). Word frequency is often regularized in Equation 1: words that appear too frequently in the text corpus are down sampled (Goldberg and Levy 2014) and rare words are pruned (Xie et al. 2015).

Word Polarity Learning

Polarity is hard to capture with BOW. It is partly the reason why *unsupervised* sentiment analysis (i.e., determining polarity) is underdeveloped and often avoided (Paltoglou and Thelwall 2012).

The sentiment analysis literature provides many supervised solutions for decoupling polarity from semantic similarity in BOW models, some of which are even done in the word-embedding vector framework (Socher et al. 2011). Expectedly, polarity labels on individual words or sentences are both proven to be useful in resolving polarity confusion. In contrast to supervised solutions, unsupervised sentiment learning has so far only seen promises in specific areas where the modeler has superior prior knowledge about the text (e.g., topic-centric discussions on social media) or areas where non-verbal context (e.g., emoticons) provides extra prior knowledge (Hu et al. 2013; Zhang et al. 2014).

Beyond sentiment analysis, some investigate into representing morphemes as distributed vectors and then words as compositions of those morpheme-embedding vectors (Lazaridou et al. 2013). Using sub-word structures has several advantages. First, decomposing certain words into morphemes gives the BOW model predictive power on some out-of-vocabulary words because new combinations of existing morphemes may make up a new word. Second, compositionality reduces training complexity because there are less number of unique morphemes than unique words.

However, these promising aspects about compositionality of embedding vectors all depend on how accurate the compositions are. If the composition of morphemes doesn't recover word meanings accurately enough, it cannot be a mainstay in BOW models. It seems that compositional vector models are most effective with supervised models (e.g., query click-through optimization in (Huang et al. 2013)). Compositionality also makes strong assumption about the language formality while misspells and alternative spellings in web text may reduce its power.

Methodology

Propagating Polarity Through Literal Similarity

Words that spell similarly are often similar in meaning due to shared linguistic root (e.g., *table* vs. *tablet*) or grammatical variations to the stem (*community* vs. *communities*, *small* vs. *smallest*, etc.).

Widely used in empirical record linkage (Cohen, Ravikumar, and Fienberg 2003), Jaro-Winkler (J-W) similarity (Jaro 1989; Winkler 1990) is a literal similarity measure that weighs more towards prefix similarity. In many cases, words similar in prefix are more likely to share meanings than words that share suffix (e.g., *development* vs. *developing*; *development* vs. *government*; *developed* vs. *underdeveloped*).

For vocabulary W, we compute a $|W|$-by-$|W|$ matrix E, where $E(i,j) = E(j,i)$ is the J-W similarity between the ith and jth word in W. In addition to J-W, other more sophisticated measures can be explored to construct E including morpheme-based methods (Lazaridou et al. 2013) and factorized methods (Chang et al. 2014).

hot, hotter, hottest are J-W similar with the same polarity while *hot, warm, cold* are J-W dissimilar with opposite polarities (like most things in natural languages, there are exceptions. For example, *heedful, heedless, unheedful* have high J-W similarity but starkly contrasted polarity). Ideally, a visualizing map should place the five words (*hot, hotter, hottest, warm, cold*) in the same local region with the first group (*hot, hotter, hottest*) being placed even closer to each other. Shown in Table 1, concatenating E and X (original $|W|$-by-d matrix of word-embedding vectors) can help formulate such polarity-distinguishing t-SNE objective: preserve existing semantic similarity (X) and asymmetrically incorporate literal similarity (E). To see why, let t-SNE compute a pairwise distance matrix P_X from X. Then, it estimates a pairwise distance matrix Q_Y from Y in the reduced dimension space (corresponding to X) such that the difference between P_X and Q_Y is minimized over Y. The asymmetry arises from the Kullback-Leibler divergence, which is t-SNE's measure of difference:

$$\mathrm{KL}(P_X|Q_Y) = \sum_{i \neq j} P_X(i,j) \log \frac{P_X(i,j)}{Q_Y(i,j)}. \tag{2}$$

Intuitively, we think that the words, which are already semantically close, are likely to express the same polarity if they share similar spelling (not vice-versa). The words, which are already semantically close, may spell very differently if they do not share the same polarity (not vice-versa). On the other hand, if the words are semantically distant, neither spelling similarity or dissimilarity is very useful in deciding the words' polarity orientation.

t-SNE's Input Assumption

A successful application of the t-SNE algorithms hinges on the distribution of pairwise distances among the original data points. Suppose original data matrix X is p-by-d (e.g., if we train $d = 200$ vectors for 50,000 words, X would be 50,000-by-200). Now we make a p-by-p pairwise distance matrix P_X:

$$P_X = \sqrt{\|X\|_2^2 + (\|X^\top\|_2^2 - 2XX^\top)^\top}, \tag{3}$$

where $P_X(i,j)$ is $\|x_i - x_j\|_2$, the l_2 distance between the ith and jth data points, with $P(i,i)$ set to 0 for all $i = 1, \ldots, p$. A core assumption of t-SNE (and many similar algorithms) is that the rows of P_X follow Gaussian distributions with a common mean μ and different variances σ_i^2: $P_X(i,j) \sim$ iid $\mathcal{N}(\mu, \sigma_i^2)$ for each j.

The data X has to fit this assumption before applying t-SNE to it, a point often taken for granted by some practitioners. Among the successful applications of t-SNE, MNIST and word-embedding vectors stand out. MNIST is a primary example dataset from the original t-SNE paper(van der Maaten

Table 1: To best encode both semantic similarity and polarity, minimize the asymmetric t-SNE objective on $[XE]$.

Examples	Ideal distance in map (Y)	Semantic distance (X)	Jaro-Winkler distance (E)	Symmetric obj. (X)	t-SNE obj. (X)	Symmetric obj. ($[XE]$)	t-SNE obj. ($[XE]$)
hot, hotter, hottest	Near	Near	Near	Low	Low	Very low	Very low
hot, warm,cold	Near	Near	Far	Low	Low	Middle	Low
chilly, Chile, chili	Far	Far	Near	High	High	Middle	High
apple, Paris, man	Far	Far	Far	High	High	High	High

and Hinton 2008). For word-embedding vectors, t-SNE is known to visually preserve regularities of the high dimensional vectors as we illustrate in Figure 1. Figure 2(a) shows the distribution of 200 randomly sampled rows from the pairwise distance matrix P generated from MNIST data. Each curve in Figure 2(a) corresponds to a row in P, the same as the distribution of the l_2 distances between a data point in X and all other data points. Similarly, Figure 2(b) shows that for word vectors trained on 100MB Wikipedia text; Figure 2(c) for word vectors trained on 260MB Facebook text (details in experiment section); Figure 2(d) for matrix E, word literal similarity vectors measured by Jaro-Winkler similarity for words in the Wikipedia dataset.

Curves in Figure 2(a), (b), and (c) generally fit Gaussian distribution with different means and variances. But with its long tails, Figure 2(d) does not fit. This violation of t-SNE's requirement means we cannot simply append P_E to the original P_X and hope that t-SNE will sort it out.

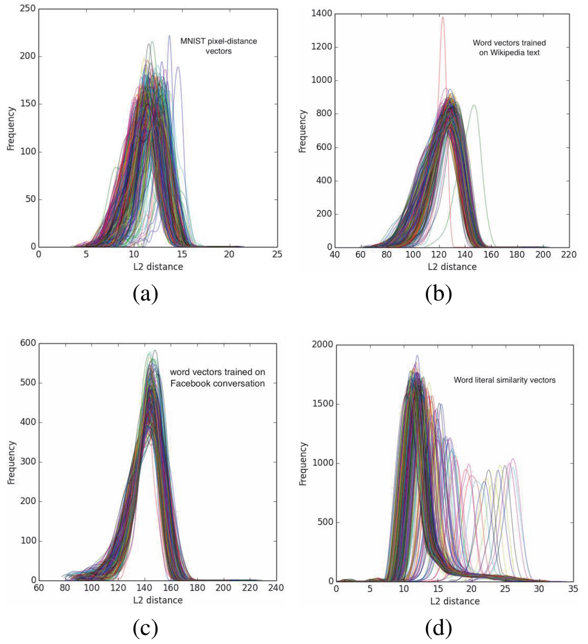


Figure 2: *(a)* Row-wise l_2-distance distributions for MNIST digits; *(b)* $P_X(i,.)$ trained on English Wikipedia corpus; textit(c) $P_X(i,.)$ trained on Facebook bilingual corpus; textit(d) $P_{[XE]}(i,.)$ trained on English Wikipedia corpus.

Autoencoding for t-SNE compliance

Shown in Figure 5 (a), principle component analysis (PCA) is first unsuccessfully tried on the columns of $[XE]$ because PCA does not proactively promote Gaussian-ness in output. A neural autoencoder (AE) is a compelling option because it is theoretically universal (Hornik, Stinchcombe, and White 1989) and regularization can be systematically expressed in its training, which is key to its malleable output distributions.

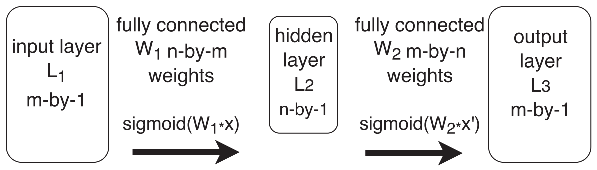


Figure 3: A generic, single-hidden-layer autoencoder.

Neural AE (Hinton and Salakhutdinov 2006) is a popular choice for learning alternative representation from raw data. We construct a single-layer AE in Figure 3: input layer L_1, hidden layer L_2, and output layer L_3. L_1 and L_2 have respective weights (W_1, W_2) and offsets (b_1, b_2 not pictured) to be learned. Its goal is to approximate the identity function over a given input dataset with (any) constraints put on W_1, W_2, b_1, and b_2. For every piece of data $x_i \in \mathbb{R}^m$ from the input dataset $\{x_1, x_2, \ldots, x_p\}$, the AE aims to learn

$$\sigma\left(W_2 \cdot \sigma(W_1 \cdot x_i + b_1) + b_2\right) - x_i = 0, \tag{4}$$

where $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid activation function. To solve Equation 4, let $J(x_i, W_1, W_2, b_1, b_2)$ denote the left-hand side of Equation 4 and find

$$\begin{aligned}\arg\min_{W_1,W_2,b_1,b_2} \sum_{i=1}^{p} \frac{1}{p} J^2(x_i, W_1, W_2, b_1, b_2) + \\ \frac{\lambda}{2}\left(\|W_1\|_2^2 + \|W_2\|_2^2\right) + \beta \sum_{j=1}^{n} \mathrm{KLs}\left(c|\frac{\|\mathbf{c_j}\|_k^k}{p}\right),\end{aligned} \tag{5}$$

where KLs $(p|q)$, a short-hand for applying K-L divergence to scalars $0 < p < 1$ and $0 < q < 1$, is defined as KL $([1-p,p]|[1-q,q])$, the divergence between two 2-bin histograms.

In addition to the λ term (l_2 regularization), Equation 5 regularizes concentration and promotes somoothness in the model's output distributions. For $k = 1$, the β term computes the K-L divergence between a desired sparsity and observed sparsity across the neurons in hidden layer. c is the desired sparsity level (e.g., 0.2) and $\frac{\|\mathbf{c_j}\|_1^1}{p}$ is the average observed neuron activation sparsity at the jth hidden neuron in L_2 over all p data points (each $\mathbf{c_j}$ is a 1-by-p vector). For $k > 1$ (e.g., $k = 2$), KL $\left(c|\frac{\|\mathbf{c_j}\|_k^k}{p}\right)$ penalizes concentrating the already sparse hidden neuron activations and promotes smoothness in the row-wise l_2-distance distributions of the re-encoded

$[XE]$. With concentration penalty, AE is able to reconstruct inputs with a more consistent common mean μ than without it (experimentally shown in Figure 5).

It is possible to extend the proposed AE by adding more layers (Hinton and Salakhutdinov 2006) or by explicitly assuming Gaussian priors throughout each neural layer (Kingma and Welling 2013).

Experiments

Table 2: Statistics on two text corpora.

Data	Size	Vocabulary	Words
`enwik100`	100MB	71,291	16,718,843
`enesfb260`	260MB	96,682	49,275,348

We conduct experiments on two text corpora (MNIST is also used but only to help reader understand the problem in the familiar MNIST setting). The first corpus is the first 100 million bytes of English Wikipedia dump (`enwik100`). The second corpus is 260 million bytes of English/Spanish comments posted publicly on Facebook(`enesfb260`). We include `enesfb260` in our experiments to see how a colloquial text corpus will differ from the classic Wikipedia corpus with misspells and other Internet lingo. The comments are collected from 4 topics:

- *Amazon.com, Amazon Kindle*;
- *Delta airlines, United airlines, Southwest airlines*;
- *Manchester United, Chelsea FC, Liverpool FC*;
- and *NBA, Kobe Bryant, LeBron James, Nike*.

More statistics about `enwik100` and `enesfb260` are summarized in Table 2. We train word vectors on both `enwik100` and `enesfb260` using Mikolov's `word2vec`[4] software with modest settings: 100 dimensional vectors trained with negative sampling (25 negative samples per positive sample).

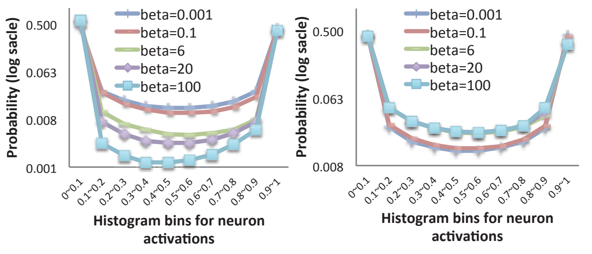


Figure 4: Neuron activation distribution (trained from `enwik100` data) in hidden layer L_2 with different β weights. *(a)* regularization with $k = 0.5$. *(b)* with $k = 2$.

[4]https://code.google.com/p/word2vec/

Evaluating Sensitivity to Parameters β and k

we investigate how effectively β and k regularize on the the objective in Equation 5. Figure 4 shows different sparsity and concentration levels in neuron activation distribution achieved by different k values in the β term introduced in Equation 5, with a fixed sparsity level $c = 0.4$. It is clear that, in Figure 4 (a), the neuron activations are more concentrated in the two extreme bins: $(0, 0.1]$ and $(0.9, 1]$ and more so with larger β weights. On the other hand, in Figure 4(b) as we increase β, neuron activation distribution becomes flatter.

In addition, we show the effects of k on P_E (Equation 3). Figure 5 reflects some insights when we apply different transformations to deskew P_E on `enwik100`. Figure 5 (a) shows linear PCA, whose inefficacy motivates our investigation into concentration-regularized AEs. Figure 5 (b) and (c) shows regularized AE with $k = 2$ and $k = 1$ after just 10 iterations of L-BFGS (Liu and Nocedal 1989), where we notice $k = 2$ converges better than $k = 1$.

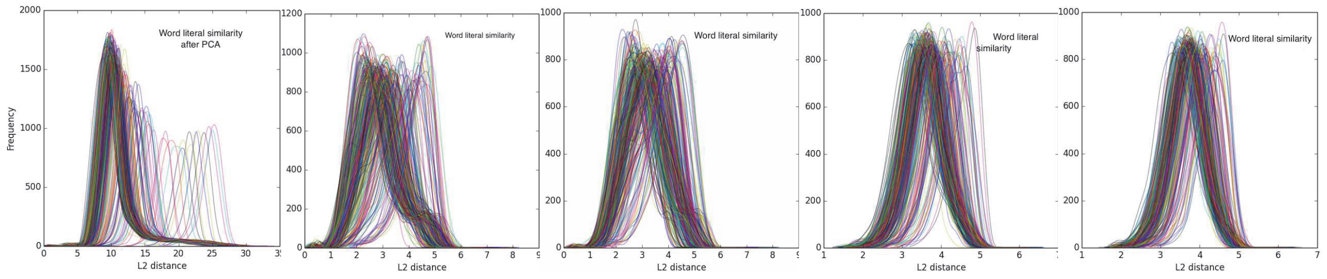
Evaluation on Human-Annotated Vocabulary

We *comparatively* and *quantitatively* assess the word groupings in modified t-SNE plots. To show the model's improvement in understanding polarity, Figure 6 uses t-SNE to plot the unmodified word vectors next to word vectors concatenated with features regularized by smooth ($k = 2$) AE. We comparison with a selected group of polarity words related to *temperature*. In Figure 6 (b), words related to *temperature* fall into three clusters, each of which has clear polarity. The upper-left one is the *warm* cluster and the other two are both *cold* ones (although the ideal visualization should group all *cold* terms into just one cluster). We note that words within each *cold* cluster share similar spellings (i.e., high J-W similarity).

We further select four domains, where polarities occur often. Then in each domain, we select a few seed words (such as *hot*, *cold*, *happy*, *sad*). Then we retrieve dozens of words that are closest to the seeds in terms of `word2vec` cosine similarity. We obtain the polarities of selected words from Thesaurus. This process results in 73 polarizing adjectives from 4 domains:

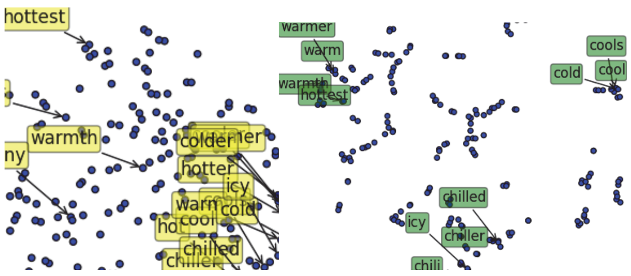
- words in *temperature* domain: warmer, warmest, warm, warmth, hottest, icy, chilled, chiller, chili, cold, colder, coldest, cools, cool, hotter.
- words in *gender* domain: male, female, masculine, masculinity, feminine, man, men, women, woman, boy, boys, girl, girls.
- words in *emotion* domain: happy, happier, happiest, sad, saddest, hate, hates, laugh, laughs, loving, love, loves, smile, likes, liked, happily, fond.
- words in *size* domain: thick, thin, thicker, thickest, thickness, thinning, thinner, thinnest, tall, taller, tallest, short, shorter, shortest, large, small, larger, largest, smaller, smallest, huge, vast, enormous, massive, sizable, immense, tiny, dwindling.

The intention is to measure how the t-SNE distances between polarizing words have reduced after the words are processed using smooth ($k = 2$) AE. To do so, we cluster



(a) (b) (c) (d) (e)

Figure 5: Transformed row-wise l_2-distance distributions for $P_{[XE]}(i,.)$ from Figure 2(c) using: *(a)* Linear PCA; *(b)* AE, $k = 1$, 10 L-BFGS iterations (Liu and Nocedal 1989); *(c)* AE, $k = 2$, 10 iterations; *(d)* AE, $k = 1$, 200 iterations; *(e)* AE, $k = 2$, 200 iterations. The number of principle components in PCA is the same as the number of hidden neurons used in AE.



(a) (b)

Figure 6: t-SNE visualization of *(a)* word-embedding vectors, and *(b)* word-embedding vectors concatenated with literal similarity columns (re-encoded with $k = 2$).

these words into M clusters via nearest neighbor (van der Maaten 2014) and the disagreement on polarity in the m-th cluster is measured using binary cross-entropy:

$$Err_m = \sum_{x}^{\text{all words}} \left[\log \sum_{i=1}^{M} \exp\left(\mathbb{1}_{x=\|i\|}\right) - \mathbb{1}_{x=\|m\|} \right], \quad (6)$$

where $\|m\| \in \{0, 1\}$ denotes the majority polarity of the words in m-th cluster and $x \in \{0, 1\}$ is an iterator over the polarity of all 73 words we consider. Table 3 summarizes the quantitative assessments. As we increase the number M, more local clusters will form. Ideally, a good visual placement of the word positions (from t-SNE) should achieve low entropy across different M values, meaning that the placement is polarity-aware at both granular and macro scales. In certain scenarios, polarity information is already encoded in word vectors. In those case, Jaro-Winkler is not very helpful in reducing the already low cross-entropy (e.g., when cross-entropy is < 0.03), but the proposed smooth ($k = 2$) AE transformation still exhibits robustness over PCA and simple sparse ($k = 1$) AE in all of our test cases.

Table 3: Measuring word grouping quality for different M and domain vocabulary. Each measure is calculated using Equation 6 and averaged by M. Standard deviations in parenthesis.

M	**Domain**	Input to t-SNE			
		X only	$[XE]$ concatenation		
			linear PCA	AE, $k = 1$	AE, $k = 2$
2	*temp.*	.078 (.001)	**.027** (.001)	**.027** (.023)	**.027** (.010)
2	*gender*	.069 (.018)	.072 (.004)	.071 (.015)	**.060** (.020)
2	*emotion*	.042 (.022)	.048 (.015)	.042 (.022)	**.021** (.014)
2	*size*	.139 (.070)	.138 (.075)	.107 (.116)	**.093** (.094)
3	*temp.*	**.025** (.002)	.081 (.039)	.026 (.007)	**.025** (.002)
3	*gender*	.119 (.044)	.128 (.024)	.078 (.027)	**.062** (.011)
3	*emotion*	**.028** (.011)	.113 (.102)	.092 (.061)	.082 (.053)
3	*size*	.216 (.165)	.393 (.276)	.178 (.164)	**.177** (.075)
4	*temp.*	.027 (.001)	.101 (.057)	.027 (.010)	**.024** (.002)
4	*gender*	.116 (.048)	.141 (.077)	.077 (.016)	**.072** (.020)
4	*emotion*	.082 (.073)	.081 (.072)	.108 (.065)	**.035** (.024)
4	*size*	**.175** (.191)	.741 (.726)	.280 (.297)	.259 (.203)
5	*temp.*	**.025** (.002)	.153 (.128)	.060 (.027)	.029 (.012)
5	*gender*	.119 (.086)	.092 (.042)	.113 (.057)	**.077** (.023)
5	*emotion*	**.046** (.042)	.083 (.087)	.186 (.147)	.077 (.029)
5	*size*	.309 (.292)	.845 (.430)	.251 (.156)	**.220** (.160)

Acknowledgments

We thank the three anonymous reviewers for their constructive comments. This work is supported in part by the following grants: NSF awards IIS-1343639, CCF-1409601, CCF-1029166; DOE awards DE-SC0007456, DE-SC0014330; AFOSR award FA9550-12-1-0458; NIST award 70NANB14H012.

Conclusions and Future Work

In this paper, we reveal unsupervised bag-of-words models' overlooking to word level polarity. We propose to investigate the integration of word polarity and word-embedding vectors using the t-SNE algorithm. We show that t-SNE's Gaussianity requirement is nontrivial to meet in this data environment and propose a solution to prepare the data for t-SNE using a regularized neural autoencoder. In the future, we would like to expand our English-based approach to be language independent.

References

Bengio, Y.; Ducharme, R.; Vincent, P.; Operationnelle, D. D. E. R.; and Recherche, C. D. 2000. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.* 3:993–1022.

Botha, J. A., and Blunsom, P. 2014. Compositional morphology for word representations and language modelling. In *ICML*.

Chang, S.; Qi, G.-J.; Aggarwal, C. C.; Zhou, J.; Wang, M.; and Huang, T. S. 2014. Factorized similarity learning in networks. In *ICDM*, 60–69.

Cohen, W.; Ravikumar, P.; and Fienberg, S. 2003. A comparison of string metrics for matching names and records. *KDD Workshop on Data Cleaning and Object Consolidation* 3:73–78.

Francis, N., and Kucera, H. 1979. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US.

Goldberg, Y., and Levy, O. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR* abs/1402.3722.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2(5):359 – 366.

Hu, X.; Tang, J.; Gao, H.; and Liu, H. 2013. Unsupervised sentimentanalysis with emotional signals. In *WWW*, 607–618.

Huang, P.; He, X.; Gao, J.; Deng, L.; Acero, A.; and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, 2333–2338.

Jaro, M. A. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *JASA* 84(406):414–420.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *CoRR* abs/1312.6114.

Lazaridou, A.; Marelli, M.; Zamparelli, R.; and Baroni, M. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *ACL*, 1517–1526.

Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*, 1188–1196.

Lee, J. A., and Verleysen, M. 2007. *Nonlinear dimensionality reduction*. Springer.

Levy, O., and Goldberg, Y. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*. 2177–2185.

Lin, Y.; Michel, J.-B.; Aiden, E. L.; Orwant, J.; Brockman, W.; and Petrov, S. 2012. Syntactic annotations for the google books ngram corpus. In *ACL*, 169–174. Association for Computational Linguistics.

Liu, D. C., and Nocedal, J. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming* 45(1-3):503–528.

Manning, C. D., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Morin, F., and Bengio, Y. 2005. Hierarchical probabilistic neural network language model. In *AISTATS'05*, 246–252.

Nguyen, K. A.; im Walde, S. S.; and Vu, N. T. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *ACL 2016*.

Paltoglou, G., and Thelwall, M. 2012. Twitter, myspace, digg: Unsupervised sentiment analysis in social media. *ACM Trans. Intell. Syst. Technol.* 3(4):66:1–66:19.

Salakhutdinov, R., and Hinton, G. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50(7):969 – 978.

Socher, R.; Pennington, J.; Huang, E. H.; Ng, A. Y.; and Manning, C. D. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, 151–161.

van der Maaten, L., and Hinton, G. 2008. Visualizing data using t-SNE. *The Journal of Machine Learning Research* 9(2579-2605):85.

van der Maaten, L. 2013. Barnes-Hut-SNE. *CoRR* abs/1301.3342.

van der Maaten, L. 2014. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research* 15(1):3221–3245.

Winkler, W. E. 1990. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research*, 354–359.

Xie, Y.; Daga, P.; Cheng, Y.; Zhang, K.; Agrawal, A.; and Choudhary, A. N. 2015. Reducing infrequent-token perplexity via variational corpora. In *ACL 2015*, 609–615.

Zhang, K.; Xie, Y.; Yang, Y.; Sun, A.; Liu, H.; and Choudhary, A. 2014. Incorporating conditional random fields and active learning to improve sentiment identification. *Neural Networks* 58:60–67.