Proceedings of the Fourteenth International AAAI Conference on Web and Social Media (ICWSM 2020)

MimicProp: Learning to Incorporate Lexicon Knowledge into Distributed Word Representation for Social Media Analysis

Muheng Yan,[1] Yu-Ru Lin,[1] Rebecca Hwa,[1] Ali Mert Ertugrul,[1] Meiqi Guo,[1] Wen-Ting Chung[2]
[1]School of Computing and Information, University of Pittsburgh
[2]Department of Psychology in Education, University of Pittsburgh
Pittsburgh, Pennsylvania, 15260
{muheng.yan, yurulin}@pitt.edu, hwa@cs.pitt.edu, {ertugrul, meg168, wtchung}@pitt.edu

Abstract

Lexicon-based methods and word embeddings are the two widely used approaches for analyzing texts in social media. The choice of an approach can have a significant impact on the reliability of the text analysis. For example, lexicons provide manually curated, domain-specific attributes about a limited set of words, while word embeddings learn to encode some loose semantic interpretations for a much broader set of words. Text analysis can benefit from a representation that offers both the broad coverage of word embeddings and the domain knowledge of lexicons. This paper presents *MimicProp*, a new graph-mode method that learns a lexicon-aligned word embedding. Our approach improves over prior graph-based methods in terms of its *interpretability* (i.e., lexicon attributes can be recovered) and *generalizability* (i.e., new words can be learned to incorporate lexicon knowledge). It also effectively improves the performance of downstream analysis applications, such as text classification.

1 Introduction

Many interesting applications of social media analysis require sophisticated natural language processing (NLP) tools. For example, sentiment analysis is a cornerstone of abusive language detection (Chen and Delany 2019), suicide prevention (Abboute and Poncelet 2014), and deception detection (Addawood and Ferrara 2019). The underpinning of a successful NLP application is its data representation because this choice impacts later decisions about models and algorithms. For example, past sentiment analysis literature shows that researchers have explored different word representations, from simple bag-of-words, to affect attributes from lexicons (Jurek and Bi 2015), to distributed vector representations (Dhaoui and Tan 2017).

A lexicon is a vocabulary list that maps a word to some attributive knowledge, often manually curated for specific domains. For example, the LIWC dictionary associates a word with several affective and grammatical attributes (e.g., "cried" belongs to "sadness", "negative emotion", "overall effect", and "a past tense verb"). Representing words by their lexicon attributes allows systems to relate words in an interpretable attribution space; however, this approach faces two challenges. First, in practice, a lexicon supports only a small number of attributes; thus, the space of representation is small and not very expressive. Second, manually curated lexicons have limited vocabularies, so out-of-lexicon words cannot be represented.

A word embedding is a dense vector representation of a word that is automatically learned by trying to model word co-occurrences from a large corpus (Mikolov and Dean 2013a). Words with roughly similar semantics tend to be closer to each other in vector space. Because the values of the vectors are continuous, the representation space is arbitrarily large (subject to data needed to train the embedding) and every word in the training corpus can be represented. Many recent works find that this type of distributed representation is useful to NLP applications (e.g., (Dhingra and Cohen 2016; Saha and Hasan 2016)). A notable example is the latest contextualized embedding models (e.g., BERT (Devlin and Toutanova 2018)) that learn the contextual relationships in texts. However, such a representation generally lacks interpretability – the values in the encoded vectors cannot be explained with any human-understandable concepts. Moreover, since it only provides a loose semantic relationship; antonyms are often considered "similar" in this representation.

This work aims to develop a word representation that reconciles the sparse, explicitly coded knowledge from lexicons with the automatically learned dense vectors of word semantics. An approach that has been taken by some researchers (Fu and Meng 2018; Tang and Qin 2014; Yang and Sun 2017) is to augment the embedding training such that lexicon attributes are given as another source of supervised information alongside the context words of the training corpus; however, labeling the corpus for these lexicon attributes is not practical. Our proposed model, "**Mimic**king **Prop**agation" (*MimicProp*), builds on an alternative approach, which applies graph-based label propagation to alter a typical word embedding so it also reflects the lexicon attribute information. Previous efforts along this direction (Faruqui and Smith 2015; Yu and Zhang 2017; Saha and Hasan 2016) were limited by three challenges that *MimicProp* overcomes: (1) in the altered vector (embedding)

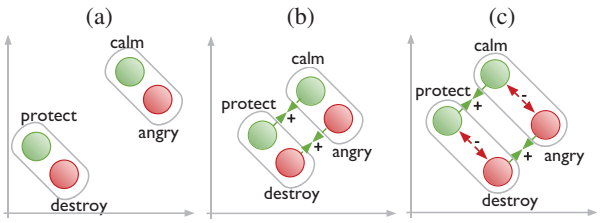

Figure 1: Illustration of different embedding spaces for word representations: (a) The typical semantic embedding overlooks the attribute relationships captured in a human curated lexicon. (b) The altered embedding with only positive constraints tend to bring all words closer in the space. (c) Our proposed method creates an altered embedding that preserve both positive and negative attribute relationships.

space, words are moved closer not only to words having similar attribute values but also to those having opposite attribute values (ref. Fig. 1); (2) most approaches are limited to the existing lexicon vocabulary and cannot be generalized to new (also referred to as *unseen* or *out-of-lexicon*) words; (3) none of them allows the lexicon attributes to be recovered from the word representations and hence the representations lack interpretability.

Our work has three unique contributions: (1) We propose a new method for learning embeddings to incorporate lexicon knowledge. We show that the embedding produced by our method better reflects the lexicon attributes and thus provides *attribute interpretability* of the embedding space for the word representations. (2) We demonstrate that such a new embedding (a lexicon-aligned representation) can be used to effectively reconstruct (for existing lexicon words) and infer (for new words) the lexicon attribute values, which improves the general applicability of many human-annotated lexicons. The technical strength lies in (a) a semi-supervised signed graph propagation algorithm to learn the lexicon aligned word representations that preserve both the semantic and attribute (both positive and negative) relationships among words, and (b) a novel method to construct a propagation graph where a new word mimics the neighborhood of a lexicon word such that both the positive and negative attribute proximities can be preserved in the graph, which allows learning the lexicon attribute for a new word more accurately. (3) We further demonstrate the advantage of our lexicon-aligned embedding through several NLP tasks related to lexicon knowledge.

We evaluate *MimicProp* on three different tasks – attribute reconstruction, attribute inference, and downstream NLP applications – with multiple lexicons including LIWC (Pennebaker and Blackburn 2015), Affect (Warriner and Brysbaert 2013) and Moral Value (Graham and Nosek 2009). The results demonstrate the advantage of *MimicProp* in accurately reconstructing and inferring the lexicon attributes for lexicon and new words, and the superior performance in downstream applications compared to baseline methods.

2 Motivation and Related Work

We hypothesize that text analysis should benefit from a word representation that reconciles the linguistic knowledge from human-curated lexicons with the distributed word representations that are automatically induced from data. Consider the example depicted in Fig. 1 involving four words: *protect, destroy, calm* and *angry*. By their semantic similarity, the vector representations for *protect* and *destroy* should be close to each other in their embedding space; as are *calm* and *angry*. On the other hand, according to an alternative attribute, such as sentiment, *protect* and *calm* share the same polarity, while *destroy* and *angry* belong to the opposite polarity. Under certain situations (e.g., sentiment analysis), capturing the word relationship according to some alternative attribute may be just as important as contextual semantic similarity. Indeed, researchers have compiled extensive lexicons to describe various linguistic attributes, including affects (Bradley and Lang 1999; Warriner and Brysbaert 2013) and moral values (Graham and Nosek 2009).

More formally, a lexicon is a finite word list, where each entry associates a word with an attribute value; the value may be binary (indicating polarity) or continuous (indicating scale). As depicted in Figure 1(c), we seek *a lexicon-aligned word representation* that satisfies: (i) words belonging to the same polarity of a lexicon attribute are closer to each other; (ii) words belonging to different polarities are further from each other; (iii) semantic relationships between words are preserved to the greatest extent possible.

Moreover, we aim to minimize any further human annotation effort. Thus, we do not try to directly train a deep language model (e.g., (Tang and Qin 2014; Yang and Sun 2017; Fu and Meng 2018)), which requires a training corpus to be augmented with lexicon attribute information through expensive human annotation effort. Instead, we seek to generate a new lexicon-aligned word representation by making use of the existing resources (i.e., existing lexicons and pre-trained embeddings). There have been two lines of research closely related to this. The first are works that altered embeddings based on a *graph-based model*. In a graph-based model, a graph was constructed to capture the lexicon information. With the graph, the embeddings of these lexicon words were altered from their pre-trained semantic word representations through a graph-based label propagation method such as Jacobi iteration algorithm (Bengio and Le Roux 2006). For example, Faruqui et al. (2015) proposed a *Retrofit* model to retrospectively alter the representations of the lexicon words (from PPDB and WordNet) to be aligned with their lexical semantics. Yu et al. (2017) proposed a similar model for sentiment embeddings; Saha et.al. (2016) extended the idea with a sentence discourse graph. A common limitation in these works is that the learned representations are restricted to the existing lexicon words (since they exclusively relied on building a graph from the lexicon set); thus, it is yet clear how to transfer the embedding learning to out-of-lexicon words.

In this work, we propose the first graph-based semi-supervised learning framework that overcomes this limitation. Our approach utilizes a novel graph construction

and augmentation method to capture and transfer the lexicon relationships existing in the lexicon words to out-of-lexicon words. We demonstrate that our method not only has superior performance in embedding learning, but also outperforms all the existing methods in downstream NLP tasks. Note that the goal of this work is distinct from those network-embedding approach that primarily focused on learning the embeddings from the given network topology (and some with node attributes, e.g., (Liao and Chua 2018)) without considering the transferring of the attribute knowledge from the unlabeled nodes to labeled nodes.

The second line of works produced lexicon scores on words without altering the word representations. Garten et al. (2018) inferred the lexicon attributes for unseen words directly from a pre-trained embedding space based on the words' cosine similarity distances. Other similarity measures, such as co-occurrence (Velikovich and McDonald 2010) and other distance metrics (Hamilton and Jurafsky 2016) have also been proposed. Hamilton et al. (2016) used a label propagation method to alter the learned scores while the word embedding was used only for calculating the node similarities in a graph and remained unchanged through the learning.

Unlike existing works, we seek to find a lexicon-aligned representation that captures both the semantic and lexicon attribute associations for not only the existing lexicon vocabulary but also out-of-lexicon words, such that words' lexicon relationships can be interpreted through the representation, and new words' lexicon attributes can be inferred based upon the representation. We show the strength of our approach by comparing it with works that are closely related to these lines (Faruqui and Smith 2015; Tang and Qin 2014; Mikolov and Dean 2013a). Moreover, the recently proposed contextualized embedding approach (e.g., BERT (Devlin and Toutanova 2018)), which is purposefully built for learning contextual relationships in text, has led to great progress in many NLP tasks. We show that our method is able to further improve the classification performance over using BERT alone.

3 Method

To formalize our graph model, we introduce the following notations. We have a large set of vocabulary words V with pre-trained semantic representation $\mathbf{X} \in \mathbb{R}^{|V| \times d}$. We also have a human-curated lexicon, which consists of V_{lex} words and their associated attribute values (denoted as s_i for some word $i \in V_{lex}$). Because lexicons are relatively small, we assume V_{lex} to be a proper subset of V. We denote the portion unseen in the lexicon as $V_{un} \subseteq V$, such that $V_{lex} \cup V_{un} = V$.

We then construct the lexicon constraint graph $G = (V, E, \mathbf{W}, \mathbf{X})$, where a node $i \in V$ is a word, and an edge $e_{ij} \in E$ connects nodes i and $j \in V$; it represents the attribute relationship between them. We assign a corresponding edge weight $w_{ij} \in \mathbf{W}$ to express the strength of the relationship (see Sec 3.1). Note that the attribute relationship may be explicit (when $i, j \in V_{lex}$) or inferred (when i or $j \in V_{un}$) (see Sec 3.2).

We also associate a distributed word representation $\mathbf{x}_i \in \mathbb{R}^d, \mathbf{x}_i \in \mathbf{X}$ with each node i. Initially, $\mathbf{x}_i$ is set to be the pre-trained semantic representation; its values will change during the embedding propagation process (see Sec 3.3). Our goal is to learn a lexicon-aligned word representation $\hat{\mathbf{X}}$ that reconcile the semantic information (from the initial semantic embedding) and the lexicon information (captured in the lexicon constraint graph G), based on which the lexicon attributes of words can be reconstructed or inferred (see Sec 3.4).

3.1 Construction of the Lexicon Graph

We construct the graph in two steps: (i) to represent the lexicon information as a graph for lexicon words (in V_{lex}), and (ii) to augment the graph to include unseen words from V_{un}.

The initial graph is constructed from information extracted from the lexicon. We first normalize the lexicon attribute values s to $[-1, 1]$ (for the binary-attribute case, it will be exactly two values, $\{-1, 1\}$). We separate the lexicon words into two groups, V^+ and V^-, based on the sign of their normalized attribute values. We define the edge weights between two nodes i and j as:

$$w_{ij} = \begin{cases} 1 - |s_i - s_j| & s_i s_j > 0 \\ -|s_i - s_j| & s_i s_j < 0 \\ 0 & s_i s_j = 0 \end{cases} \quad (1)$$

Thus, a *positive edge* exists between words of the same attribute polarity, and a *negative edge* between words of different polarities. During embedding propagation, these edges will pull nodes closer or push them apart.

To preserve both the similar and opposing attribute relationships in a graph, we create small and equal numbers of positive and negative edges for each node. Such sparse and balanced edge creation enables efficient learning of node representation, which will be further discussed in the later section (Sec. 3.3). This procedure differs slightly depending on the lexicon attribute type, i.e., a binary or real-valued attribute. In the case of **binary** (or categorical) lexicon (e.g., LIWC lexicon, see Sec. 4) attribute, for each word $i \in V$, we randomly sample m words from V^- and another m words from V^+ to create $2m$ edges for each node. An edge between the same pair of words is created only once. In the case of a **real-valued** lexicon (e.g., Valence lexicon) attribute, the edges between pairs of nodes can be categorized into three groups based on the signs of edge weights (as in Eq. 1): *positive edges*, *negative edges*, and *neutral edges*. For each node, we create m *positive edges* connecting to nodes having the closest attribute values and m *negative edges* connecting to m different nodes randomly sampled from the opposite set. Specifically, the *positive edges* are created by first sorting all words according to their attribute values/scores, and connecting each word with $\frac{m}{2}$ words with the closest smaller lexicon scores and $\frac{m}{2}$ having closest larger scores. The constructed lexicon graph has degree distribution denoted as $\mathcal{D}$.

3.2 Augmenting the Graph

The next step is to add the unseen words as new nodes with edges connecting to the existing nodes in the graph such

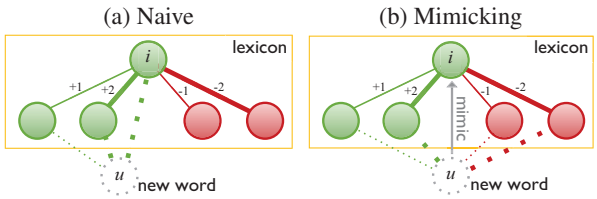

Figure 2: Inferring the representation of new (unseen) words through (a) connecting to similar lexicon words in a naive manner, and (b) mimicking the neighborhood of a similar lexicon word. The naive method tends to reinforce the similar relations but ignore the dissimilar ones in the lexicon. The mimicking method is able to create a balance structure that captures both similar and dissimilar relations.

that the lexicon information can be propagated from lexicon words to unseen words. We consider two ways of augmenting the lexicon graph: a naive approach based on the idea of the k-Nearest Neighbors, and the proposed *node mimicking*.

A Naive Approach A straightforward way to add unseen words into the graph is to create edges connecting the unseen words to related lexicon words where the relatedness may be defined based on certain semantic relationships, as has been proposed in prior works (Hamilton and Jurafsky 2016; San Vicente and Rigau 2014; Tai and Kao 2013). Similarly, we create edges between the unseen and the existing lexicon words based on their semantic closeness as measured in the semantic embedding space. To preserve the degree distribution in the lexicon graph, for each unseen word u, the number of edges (degree), denoted as dgr, is sampled from the degree distribution of the original lexicon graph (i.e., $dgr \sim \mathcal{D}$), and edges are created to connect u to the dgr closest lexicon words based on the distance measured in the semantic embedding space, with an edge weight given by:

$$w_{ui} = \exp(-dist(\mathbf{x}_u, \mathbf{x}_i)) \tag{2}$$

where $u \in V_{un}$ is an unseen word and $i \in V_{lex}$ a lexicon word, and dist$(\cdot)$ is a distance function that captures the distance between the two words represented as vectors $\mathbf{x}_u$ and $\mathbf{x}_i$. It can be chosen from commonly used distance metrics such as cosine or Euclidean distance; we use cosine distance in this work. The edge weight is designed to decrease exponentially with the semantic closeness to give more weights on semantically close pairs.

Note that all edges created following Eq. 2 are *positive edges* (as shown in Fig. 2(a)), and no *negative edges* will be created with this naive method. As a consequence, the positive edges outnumber the negative edges and make it difficult to distinguish words with opposite attribute values from the graph structure.

The Node Mimicking Approach The core idea of our proposed approach is to reproduce both positive and negative lexicon relationships for an unseen word by copying the *neighborhood* of a similar lexicon word, as illustrated in Fig. 2(b). Specifically, for each unseen word $u \in V_{un}$, we identify k closest lexicon words (according to the distance measured in the semantic embedding space) as the set $\mathcal{N}^S(u)$, and for each closest lexicon word $i \in \mathcal{N}^S(u)$, we copy its neighborhood to u by creating an edge between u and a lexicon word z that has been connected to i on the original lexicon graph. The edge weight is given by:

$$w_{uz} = \frac{1}{k} \sum_{i \in \mathcal{N}^S(u), z \in \mathcal{N}^L(i)} w_{ui} \cdot w_{iz}, \tag{3}$$

where $u \in V_{un}$ is an unseen word, $i \in \mathcal{N}^S(u) \in V_{lex}$ is a lexicon word in the set $\mathcal{N}^S(u)$ representing the k lexicon words closest to u. We use $\mathcal{N}^S(\cdot)$ to denote the neighborhood derived from the semantic embedding space, as opposed to $\mathcal{N}^L(\cdot)$ that denotes the neighborhood derived from the lexicon graph. w_{ui} can be defined as in Eq. 2 to capture the semantic closeness between the unseen word u and the lexicon word i, and w_{iz} is given as in Eq. 1 to capture the lexicon relationship between two lexicon words i and z. The edge weight w_{uz} is a function of both the semantic closeness and the lexicon relationship, aggregating through u's neighbors in the semantic space.

Such neighborhood mimicking balances between positive and negative edge creation but leads to considerably larger degrees on unseen words (roughly $k \cdot dgr$) compare to those lexicon words.

As we will discuss in the next section (Sec. 3.3), in the learning process, nodes with a larger number of connections will have a larger influence than those with fewer connections, and the larger degrees on unseen words may overtake the influence of the lexicon words. Thus, we introduce a *graph sparsification* step to make sure that the degrees of non-lexicon words are similar to those of lexicon words. For each non-lexicon word u with connectivity defined by an adjacency vector $\mathbf{w}_u = \left(w_{u1}, \ldots, w_{u|V|}\right)$, we sample a degree $dgr \sim \mathcal{D}$ and preserve the $\frac{1}{2}dgr$ largest (positive) and the $\frac{1}{2}dgr$ smallest (negative) values in $\mathbf{w}_u$ with all other entries set to zero. As a result, the degrees of the non-lexicon words will have a similar distribution with $\mathcal{D}$, with a small but neglectable increase in the degrees of the lexicon words due to newly added nodes. The balance of positive and negative edges connecting to each node can be achieved with the procedure detailed in Appendix A.

3.3 Embedding Propagation on A Signed Network

We describe our method for altering the pre-trained semantic embedding to create a word representation that aligns with the lexicon attribute information captured by the lexicon graph. We expect the word embedding vectors to be similar to/different from those with similar/different lexicon attributes while also preserve the semantic information given by the pre-trained embedding.

The label propagation method based on the Jacobi iteration, which was used in prior works to deal with unsigned graphs (e.g., (Faruqui and Smith 2015)), is not applicable in our case due to the convergence problem in the Jacobi iteration on signed graphs (Bengio and Le Roux 2006;

Saad 2003). Our approach builds on the label spreading algorithm (Zhou and Schölkopf 2004) and the label spreading in signed-network (Zhang and Kuang 2012), which was based on the normalized Graph Laplacian (see below). Moreover, instead of spreading node *labels*, we consider the spreading of multidimensional *embedding vectors* through positive and negative edges. Thus, our *embedding propagation* is to minimize the cost:

$$C(\hat{\mathbf{X}}) = \frac{1}{2}\left(\sum_{i,j}^{n} |w_{ij}| \left\| \frac{\hat{\mathbf{x}}_i}{d_{ii}^{1/2}} - sgn(w_{ij}) \frac{\hat{\mathbf{x}}_j}{d_{jj}^{1/2}} \right\|^2 + \mu \sum_{i}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2 \right) \quad (4)$$

where d_{ii} is from the degree matrix of the graph $\mathbf{D}$, and is defined as $d_{ii} = \sum_j |w_{ij}|$. The first term in the objective function seeks to make the position (given by the embedding vector $\hat{\mathbf{x}}_i$) of a word i in the graph to be closer to those with positive weight connections, and further to those with negative weight connections, while the second term constraints the word's position (the learned representation $\hat{\mathbf{x}}_i$) to be similar to the initial one ($\mathbf{x}_i$). The influence of the two objectives reflected in the two terms is controlled by the hyper-parameter μ. The normalization factor $d_{ii}^{1/2}$ follows the property of normalized graph Laplacians[1]. The parameter μ is used to balance the contribution from the semantic relationships (second term) and the lexicon attribute relationships (first term).

Algorithm 1 provides the iterative updating algorithm that solves the optimization problem, which will converge to the solution $\hat{\mathbf{X}} = (\mathbf{I} - \alpha\mathbf{S})^{-1}(1-\alpha)\mathbf{X}$, where $\mathbf{S}$ is the normalized Graph Laplacian defined as: $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, and $\alpha = \frac{1}{1+\mu}$ is a transformed version of μ. The mathematical derivation of solving the problem is provided in the Appendix A.

Algorithm 1: Embedding Propagation

Input: a graph $G = (V, E, \mathbf{W}, \mathbf{X})$, hyper-parameter α
Output: lexicon aligned embedding $\hat{\mathbf{X}}$
construct the degree matrix $\mathbf{D}$, with diagonal entries $d_{ii} \leftarrow \sum_j |w_{ij}|$;
calculate graph Laplacian: $\mathbf{S} \leftarrow \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$;
initialize $\hat{\mathbf{X}}^{(0)} \leftarrow \mathbf{X}$;
while *not converged* **do**
  $\hat{\mathbf{X}}^{(t+1)} \leftarrow \alpha\mathbf{S}\hat{\mathbf{X}}^{(t)} + (1-\alpha)\mathbf{X}$;
end

[1] In the regularization framework proposed by (Zhou and Schölkopf 2004), the normalization $d_{ii}^{1/2}$ is used such that the property of normalized graph Laplacians, $\mathbf{f}'\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j} w_{ij}\left(\frac{\mathbf{f}_i}{d_{ii}^{1/2}} - \frac{\mathbf{f}_j}{d_{jj}^{1/2}}\right)^2$, where $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, is satisfied. Following this property, the problem can be cast and solved in the form of the standard Rayleigh-Ritz theorem. See, e.g., (Shi and Malik 2000; Von Luxburg 2007), for more details.

Here, we leveraged the label spreading approach (specifically for signed network) to reconcile both the semantic relationships and lexical knowledge. On one hand, the semantic relationships are preserved as much as possible as defined by the second term of the objective function in Eq. 4, i.e., close words in the initial embedding space should have similar representations. On the other, words connecting with positive (negative) edges on the constructed lexicon graph should have similar (dissimilar) representations, as defined by the first term of the objective function in Eq. 4. Our embedding propagation is not simply based on the word distance in the embedding space but the constructed graph that was designed to reflect the positive and negative relationships captured in a given lexicon.

We further discuss some considerations in our graph construction and augmentation: the sparse and balanced edge creation, as well as the preservation of node degree distribution. (1) The sparse graph ensures learning efficiency. The complexity of the graph embedding propagation algorithm depends on the number of edges in the given graph. With a dense graph, e.g., $|E| \approx N^2$, where N is the number of nodes, the complexity can be as high as $O(N^2)$. The introduction of a kNN-like graph construction allows the algorithm to scale linearly with $O(N)$, regardless of the density of the original lexicon relationships. (2) The balance between positive and negative edges per node ensures the distribution of the learned attribute classes to be consistent with the original class distribution in the lexicon nodes. Note that the purpose of the graph construction is to enable the lexicon (attribute) knowledge to be transferred from labeled nodes to unlabeled nodes. The label propagation is influenced not only by the edge weights (connection strength) but also by the edge signs (the presence of positive or negative edges). Positive edges result in "pull" between nodes while the negative edges result in "push" between nodes in the embedding space. Therefore, uneven distribution of positive and negative edges will lead to undesirable results for the unlabeled nodes (i.e., incorrect inference of its attribute polarity) even if the edge weights were perfectly assigned. Moreover, if the attribute classes in the initially labeled set of nodes are already imbalanced, without balancing edges from each class, the nodes with the dominant set of class labels would collectively have more edges and dominate the attribute value propagation to the unlabeled nodes. Similarly, a node with more connections will have a higher influence than a node with fewer connections in the embedding propagation. Thus, the edge balance and the preservation of degree distribution are used to ensure the class influence as well as the influence of individual nodes to be consistent with that in the lexicon data.

3.4 Recovery of the Lexicon Attributes

To recover the lexicon attributes based on the learned representation, we create a frame of reference – two pseudo poles $\mathbf{p}^+$ and $\mathbf{p}^-$ – in the corresponding embedding space. The positive pole is defined as the mean vector of all words with the positive attribute values: $\mathbf{p}^+ = \frac{1}{|V^+|}\sum_{i \in V^+} \mathbf{x}_i$, and the negative pole is: $\mathbf{p}^- = \frac{1}{|V^-|}\sum_{i \in V^-} \mathbf{x}_i$.

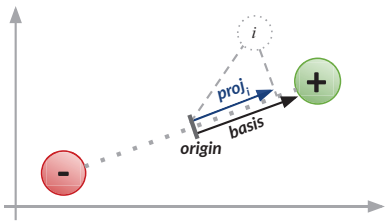

Figure 3: The lexicon attribute of any word can be recovered through the projection on the basis vector in the new embedding space.

A *basis* vector can be defined based on the two pseudo poles as $\mathbf{b}_s = \frac{1}{2}(\mathbf{p}^+ - \mathbf{p}^-)$, with an *origin* given by $\mathbf{o}_s = \frac{1}{2}(\mathbf{p}^+ + \mathbf{p}^-)$. The basis vector represents the axis of the attribute and allows a word's attribute value to be recovered by projecting the word on the basis. As shown in Fig. 3, the recovery of a word's attribute value is given by the inner product between $(\mathbf{x}_i - \mathbf{o}_s)$ and $\frac{\mathbf{b}_s}{||\mathbf{b}_s||^2}$, as $\frac{(\mathbf{x}_i - \mathbf{o}_s)\cdot\mathbf{b}_s}{||\mathbf{b}_s||^2}$. This can be applied to both lexicon and non-lexicon words, for lexicon attribute reconstruction and inference respectively.

4 Experiments

This work seeks to tackle the challenges of "interpretability" and "generalizability" in the learning of lexicon knowledge. Our experiment is designed to answer the following questions:

1) [Interpretability] Can the representation of words from the learned embedding be better associated with the lexicon attributes that would be given by humans? In other words, can we "interpret" the lexicon relationship for a set of words – e.g., whether they have similar sentiments, or similar moral sense – from the learned embedding?
2) [Generalizability] Can the representation of words from the learned embedding be used to transfer the human-annotated lexicon knowledge to unseen words? In other words, given a previously un-annotated word, can we use the learned embedding to infer the word attribute?

To answer the first question, we examine whether the lexicon attributes (lexicon scores) can be reconstructed based on the embedding produced by our method. To answer the second question, we examine whether the attribute of unseen words can be effectively inferred by our method using a hold-out experiment. Then, to further examine the broader applicability of such learned embedding produced by our method, we test whether the embedding can be used in downstream NLP tasks such as sentence-level sentiment classification, as well as sentiment and moral classifications at the document (tweet) level.

The experiments include three main tasks: (1) word-level lexicon reconstruction, (2) word-level lexicon inference (for unseen words) , and (3) text classification (downstream NLP tasks) using lexicon aligned embeddings. For text classification, we conduct three sets of experiments: (a) **sentiment** classification, (b) **valence** regression, and (c) **moral value** classification. The first two tasks are presented in Sec. 4.2, and the NLP tasks are detailed in Sec. 4.3.

4.1 Lexicons and datasets used in the experiments

(a) Sentiment classification with the **LIWC** Lexicon and **Stanford Sentiment Treebank (SST)** Dataset: **LIWC** is a lexicon that includes the attributes of words in many psychological and linguistic categories (Pennebaker and Blackburn 2015). We include the categories of *positive emotion* and *negative emotion* in the experiment, and obtain 4,117 positive and 5,031 negative emotion words (whose embedding vectors can be found in the pre-trained embedding described in Sec. 4.2). The **Stanford Sentiment Treebank (SST)** Dataset provides annotation for each sentence with a sentiment score in five levels from 0 to 4, where the scores represent *very negative, negative, neutral, positive, and very positive* respectively (Socher and Potts 2013). We include two versions of this dataset: (i) *SST-Fine* has the five-level sentiment labels for sentences. There are 8,544 sentences in the training set, 1,101 in the development set, and 2,210 in the testing set. (ii) *SST-Binary* has only 0 and 1 in the labels representing sentences with a negative or positive sentiment, and the neutral sentiment sentences are removed. There are 6,920 sentences for training, 872 for development, and 1,821 for testing in this version.

(b) Valence regression with the **Valence Lexicon** and **Twitter Affect (Tweet-Valence)** Dataset: **Valence Lexicon** from the *valence-arousal-dominance* (VAD) lexicon developed by Warriner et. al (2013), provides the annotated affect of words. In this lexicon, "valence" is defined as the pleasantness expressed by a word, given as a real number from 1 (most negative) to 9 (most positive), with 5 the neutral valence. There are in total 13,915 English words in the lexicon. The **Twitter Affect (Tweet-Valence)** Dataset is obtained from the *SemEval 2018 Task 1*, developed for a valence classification (Mohammad and Kiritchenko 2018). The dataset provides tweets with human-annotated valence scores at the tweet level. The scores are normalized between 0 and 1, where 0 is the most negative and 1 is the most positive. There are 1,181 tweets in the training set, 449 in the development set, and 937 in the testing set.

(c) Moral value classification with the **Moral Foundation Lexicon** and **Moral Tweet** Dataset: **Moral Foundation Lexicon** is a categorical lexicon developed by Graham et al. (2009). Words in this lexicon are assigned with *moral foundation* polarities in five moral dimensions. The words in the lexicon include real English words and word stems. We pre-process the stems into real words that can be found in the pre-trained embedding, which results in 420, 158, 251, 369, and 393 words in the categories of Care/Harm, Fairness/Cheating, Loyalty/Betrayal, Authority/Subversion, and Purity/Degradation, respectively. **Moral Tweet Dataset**[2] developed by Johnson and Goldwasser (2018) is used in the moral value classification task. The dataset provides human-annotated tweets with a moral value conveying the tweet author's perspective. We eliminated tweets that are no longer accessible. Due to the insufficient sample size, only two of the dimensions, *Harm/Care* and *Authority/Subversion*, are used for the polarity classification task. In total, there are 313 and 499 tweets for *Harm* and *Care*, and 187 and 376

[2]https://github.com/kmjohnson/twitter-morals

tweets for *Authority* and *Subversion*, respectively. [3]

4.2 Reconstruction and Inference

We evaluate the effectiveness of our approach in terms of how well the produced word representation enables the *reconstruction* (for lexicon words) and *inference* (for unseen words) of the lexicon attribute values. Our experiment includes different types of lexicon attributes: the **sentiment** lexicon encodes binary attribute and the **valence** lexicon has real-valued scores. We include the following widely adopted or state-of-the-art approaches into four baseline methods.

(i) *Google News Word2Vec* embedding (Mikolov and Dean 2013b) is trained with the skip-gram architecture on the Google news dataset with around 100 billion words and covers a vocabulary of around 3 million tokens. Each token, either a word or a phrase, is represented as a vector $\mathbf{x}_i \in \mathbb{R}^d$ where $d = 300$, and the phrases are captured with a data-driven method described in the work of Mikolov et.al (2013a). In this experiment, the Google Word2Vec is used to create an initial word representation (the pre-trained word embedding) for methods including *MimicProp* and *Retrofit*. We also use it as a baseline method where the attribute values are reconstructed/inferred using the vector projection (ref. Sec. 3.4) in the pre-trained word embedding space.

(ii) *Retrofit* (Faruqui and Smith 2015) is an alternative graph model based on Jacobi Iteration. As described in Sec. 3.3, due to the convergence issue in the Jacobi iteration, to make a meaningful comparison, we create an unsigned version graph that follows our graph construction (Sec. 3.1 and 3.2) except that all lexicon words are connected to $2m$ similarity-based neighbors with positive weights. We then apply *Retrofit* on this unsigned version.

(iii) *DDR* (Garten and Dehghani 2018) which stands for *Distributed Dictionary Representation* is a model that infers the lexicon attribute from distributed word representations. The model represents the vector of a lexicon attribute as the mean vector of all words with that attribute as: $\mathbf{x}_a = E[\mathbf{x}_i], i \in V_a$, and infer the score of a word u as $\hat{s}_u = \frac{\mathbf{x}_u \cdot \mathbf{x}_a}{\|\mathbf{x}_u\| \cdot \|\mathbf{x}_a\|}$ by calculating their cosine similarity with the attribute vector. For a binary attribute (two mean vectors), the score is computed as: $\hat{s}_u = \frac{\hat{s}_u^+}{\hat{s}_u^+ + \hat{s}_u^-}$

(iv) *SentProp* (Hamilton and Jurafsky 2016) is a method that learned lexicon attributes of words from a small set of seed words through label propagation. As described in Sec. 2, this method does not produce new word representations. We implement the method following the details reported by Hamilton et al.

We also conduct an *ablation study* that compares the proposed approach with alternative design choices, including changes in (a) methods for constructing propagation graphs (naive vs. neighborhood mimicking), (b) embedding propagation algorithms (unsigned Retrofit vs. propagation on a signed network), and (c) methods of calculating the attribute scores (cosine method as in the *DDR* work vs. projection on the basis as described in Sec. 3.4). These variant settings cover all alternative design choices in addition to "*MimicProp*" (mimic, signed, proj) and the baseline "*Retrofit*" (naive, unsigned, proj).

Experimental Setup and Metrics. We compare our model with other baseline models on a 10-fold cross-validation experiment. In each fold, 90% of the lexicon words are used as training samples and 10% of the words as unseen words. We construct the graph based on the steps in Sec. 3.1, and set $m = 20, \alpha = 0.85$ (chosen based on the hold-out experiment). The propagation graph is constructed based on the steps in Sec. 3.1 and 3.2. Two different types of graphs are created (the naive propagation graph and the *MimicProp* graph), and are used with both our model and the *Retrofit* baseline. For the *MimicProp* graph, the k is set to 5. After the optimization, the lexicon scores will be reconstructed/inferred for lexicon/unseen words as in Sec. 3.4.

We compare our model with the baselines on the metrics of *Pearson Correlation Coefficient* r and *Macro F1 Score* between the inferred score $\hat{s}$ and the binary ground truth lexicon score s. The correlation coefficient reflects how numerically the inferred scores are accurate, and the Macro F1 score represents the separation of the inferred scores on the lexicon attribute polarities.

Evaluation Results. As shown in Table 1, both *MimicProp* and *Retrofit* have achieved a nearly perfect performance on the LIWC lexicon reconstruction task (*SentProp* is not applicable to the reconstruction task), showing that both methods can create word representations that align well with the lexicon relationships. However, the good performance in recovering attribute values from the known lexicon words does not mean that the method is generalizable to unseen words. Hence examining different approaches' inference capability is crucial. As shown in the inference task, overall *MimicProp* has achieved the best results under all metrics, showing its advantage in accurately generalizing the lexicon information on unseen words and in providing reasonable separation (in terms of the F1 score) of the inferred scores on the attribute polarities.

The second part of Table 1 (MimicProp Variants) reports the results of our ablation study. Note that the results of "*MimicProp*" (mimic, signed, proj) and the baseline "*Retrofit*" (naive, unsigned, proj) have been reported in the first part of the table. In particular, with the unsigned algorithm, the "mimic" variant brings about 3% performance gain over the "naive" graph construction – see differences between *Retrofit* (naive, unsigned, proj) and MP (mimic, unsigned, proj). With the signed algorithm, the "mimic" variant brings about 6% performance gain over the "naive" version – see differences between *MimicProp* (mimic, signed, proj) and MP (naive, signed, proj). However, using signed network propagation alone has no positive effect as there are no signed edge weights to support the learning in the signed network. Our node mimicking approach is designed to support the signed relational learning. The results show that all three components (mimic, signed, proj) in *MimicProp* together help achieve the best performance.

[3] In Johnson and Goldwasser's work (2018), the classification task is to determine whether a tweet is relevant to a moral dimension without distinguishing the polarity, while our experiment is designed to distinguish the vice/virtue of a moral concept; hence we do not compare their method and results in our experiment.

Table 1: Results of Lexicon Reconstruction and Inference.

	LIWC				Valence			
	Reconstruction		Inference		Reconstruction		Inference	
Method	r	F1	r	F1	r	F1	r	F1
MimicProp	1.000	1.000	**0.730**	**0.851**	0.909	1.000	**0.810**	**0.901**
Retrofit	0.994	1.000	0.698	0.825	0.993	0.992	0.764	0.873
SentProp	N/A	N/A	0.448	0.752	N/A	N/A	0.579	0.772
DDR	0.693	0.844	0.686	0.835	0.734	0.862	0.738	0.864
Google W2V	0.684	0.824	0.683	0.821	0.742	0.883	0.741	0.881
MimicProp Variants	r	F1	r	F1	r	F1	r	F1
MP (naive, signed, proj)	0.994	1.000	0.688 (-0.042)	0.807 (-0.044)	0.911	1.000	0.790 (-0.020)	0.882 (-0.019)
MP (mimic, unsigned, proj)	1.000	1.000	0.721 (-0.009)	0.827 (-0.024)	0.981	0.992	0.755 (-0.055)	0.891 (-0.010)
MP (mimic, signed, cosine)	0.991	1.000	0.723 (-0.007)	0.849 (-0.002)	0.909	1.000	0.774 (-0.036)	0.846 (-0.055)
MP (naive, signed, cosine)	0.990	1.000	0.690 (-0.040)	0.834 (-0.017)	0.864	1.000	0.771 (-0.039)	0.841 (-0.060)
MP (mimic, unsigned, cosine)	0.991	1.000	0.662 (-0.068)	0.832 (-0.019)	0.991	0.993	0.727 (-0.083)	0.817 (-0.084)
MP (naive, unsigned, cosine)	0.994	1.000	0.693 (-0.037)	0.841 (-0.010)	0.990	0.993	0.775 (-0.035)	0.838 (-0.063)

*Values in the parenthesis indicate the performance differences from the result of *MimicProp*. The ablation study covers all alternative design choices in addition to "*MimicProp*" (mimic, signed, proj) and the baseline "*Retrofit*" (naive, unsigned, proj).

4.3 Lexicon Embedding in NLP Tasks

We further test the utility of these lexicon-embedded word representations in three downstream tasks: the sentiment classification of sentences (*SST-Fine and SST-Binary*), the valence regression of Tweets (*Tweet-Valence*), and the moral classification of Tweets (*Tweet-Moral*).

We compare the word representations trained by *MimicProp* with those by the aforementioned baselines except for *DDR* and *SentProp* because the two methods only produce attribute scores without any modification of the pre-trained embedding. We include three embeddings from the state-of-the-art methods:

Twitter Embedding (Baziotis and Potamianos 2018): The Word2Vec word representations pre-trained on Twitter datasets are used in our Twitter-related tasks (Baziotis and Potamianos 2018). The 300-dimensional embedding is trained on the unlabeled tweets, with the standard skip-gram model. The Twitter word embedding is used to initialize the models and included as a baseline in the *Tweet-Valence* task, as it performs better than the Google News Word2Vec.

Sentiment Specific Word Embedding (SSWE) (Tang and Qin 2014): A sentiment embedding that is jointly trained on labeled datasets. The model is based on the skip-gram word2vec model with an additional sentiment classification layer, in order to learn the sentiment information simultaneously when training the word representations. We use the embedding[4] as a baseline in the sentiment classification and valence regression tasks.

WordPiece Embedding (Wu and others 2016): A new embedding that is adopted by the state-of-the-art NLP model architecture BERT (Devlin and Toutanova 2018). WordPiece Embedding leverages a special tokenization procedure that aims to balance between the vocabulary size and the number of out-of-vocabulary words, by breaking complex words into pieces (e.g., "kindness" was tokenized as "kind" and "-ness"). WordPiece Embedding has enabled BERT to use only around 30,000 vocabulary words while the out-of-vocabulary words are rarely encountered. We use WordPiece embedding with the BERT architecture as a baseline (details of architecture provided in the next subsection), and also train our model based on WordPiece Embedding to evaluate *MimicProp* with BERT.

Experimental Setup and Metrics. For all the downstream tasks, we compare the performances across different embeddings along with two main-stream sequential model architectures in NLP: Bi-LSTM and BERT. See appendix A for the model architecture details. For each task, the settings of these neural network models remain the same in order to compare the performance of different input embeddings.

To facilitate comparison, we adopt the performance metrics that have been commonly reported in the literature. We report "accuracy" for the sentiment classification task, "Pearson correlation coefficient r" for the valence regression task, and the macro and weighted F1 scores over two classes for the moral classification task due to class imbalance. [5]

Evaluation Results. Table 2a and 2b report the performance results from the three sets of text classification tasks with five different datasets. Overall, our proposed approach outperforms all other baselines in all tasks.

In the sentiment classification tasks, SST-Fine and SST-Binary, the lexicon-aligned word embedding produced by *MimicProp* consistently outperforms all other kinds of embedding. Even though the WordPiecd + BERT has superior performance compared to other baselines, our *MimicProp* still brings additional performance gain to this state-of-the-art technique. In the Twitter-Valence task, MimicProp + Bi-LSTM performs the best. On the other hand, the methods with BERT, trained based on the WordPiecd embedding, ap-

[4]https://www.microsoft.com/en-us/research/people/dutang/

[5]For binary lexicons (e.g. LIWC), F1 scores would be sufficient; yet Pearson r provides additional information about the loss when recovering the lexicon attributes. On the other hand, real-valued lexicons (e.g. Valence) sometimes are used in a binary classification task. Thus, both metrics are reported in the results.

Table 2: Results of Downstream NLP Tasks.

(a) Sentiment Classification and Valence Regression

Method	**SST-5** Acc.	**SST-2** Acc.	**Valence** r
MimicProp + BERT	**0.515±0.003**	**0.942±0.003**	0.827±0.003
Retrofit + BERT	0.486±0.005	0.856±0.012	0.813±0.006
WordPiecd + BERT	0.508±0.009	0.935±0.001	0.814±0.006
MimicProp + Bi-LSTM	0.465±0.010	0.851±0.008	**0.852±0.003**
Retrofit + Bi-LSTM	0.441±0.016	0.826±0.013	0.798±0.003
SSWE + Bi-LSTM	0.411±0.005	0.790±0.017	0.783±0.002
Google W2V + Bi-LSTM	0.458±0.013	0.833±0.015	0.771±0.003
Twitter W2V + Bi-LSTM	N/A	N/A	0.845±0.004
MP(mimic, unsigned) + BERT	0.488±0.004	0.835±0.009	0.813±0.004
MP(naive, signed) + BERT	0.497±0.002	0.936±0.001	0.820±0.002
MP(mimic, unsigned) + Bi-LSTM	0.451±0.005	0.832±0.005	0.802±0.003
MP(naive, signed) + Bi-LSTM	0.464±0.010	0.847±0.006	0.850±0.002

(b) Moral Classification

Method	**Harm/Care** Macro F1	**Harm/Care** Weighted F1	**Authority/Subversion** Macro F1	**Authority/Subversion** Weighted F1
MimicProp + BERT	**0.766±0.003**	**0.855±0.003**	**0.732±0.005**	**0.853±0.004**
Retrofit + BERT	0.732±0.003	0.823±0.001	0.653±0.003	0.793±0.005
WordPiece + BERT	0.734±0.002	0.816±0.003	0.701±0.004	0.824±0.006
MimicProp + Bi-LSTM	0.718±0.004	0.807±0.005	0.655±0.014	0.811±0.014
Retrofit + Bi-LSTM	0.699±0.007	0.797±0.005	0.616±0.015	0.782±0.015
Google W2V + Bi-LSTM	0.702±0.020	0.786±0.027	0.645±0.010	0.806±0.013
MP(mimic, unsigned) + BERT	0.720±0.005	0.817±0.006	0.577±0.005	0.755±0.002
MP(naive, signed) + BERT	0.788±0.002	0.857±0.001	0.687±0.003	0.823±0.002
MP(mimic, unsigned) + Bi-LSTM	0.658±0.013	0.785±0.011	0.626±0.020	0.775±0.023
MP(naive, signed) + Bi-LSTM	0.671±0.009	0.752±0.011	0.637±0.005	0.793±0.012

pear to be less effective than those trained based on the Twitter embedding, as the Twitter embedding better covers tokens (i.e. hashtags, mentions, URLs, etc.) in Twitter texts.

In the moral classification tasks, MimicProp + BERT consistently outperforms the best in both Harm/Care and Authority/Subversion classifications. It outperforms all baselines by 3–19%. Interestingly, the *Retrofit* model (Faruqui and Smith 2015), a label propagation that is closely related to *MimicProp*, often performs the worst among all methods. The results suggest that the label propagation method alone is not sufficient – by introducing the signed edge weights, our proposed approach has significant improvement over the existing label propagation method, and together with the signed network propagation algorithm, it enables learning polarized lexicon attributes through signed edges for unseen words.

The ablation study further shows the importance of the signed network propagation. Between the two design choices (mimic vs. naive, signed vs. unsigned), the "signed" option has a more significant impact in the performance gain in most of the cases – see, e.g., *Retrofit* (naive, unsigned) vs. MP (naive, unsigned) with the same architecture. Though, the "mimic" graph alone (without signed network propagation) is not sufficient. The ablation study clearly demonstrates that combining both (mimic, singed) as in *MimicProp* results in the best performance in these downstream tasks.

5 Qualitative Results

We present qualitative results and case studies to demonstrate the strength of our proposed method.

Reconciliation of semantic information with lexical knowledge. We illustrate how *MimicProp* reconciles the semantic information (from the initial semantic embedding) with the human-curated lexicon knowledge further through network visualization and analysis. Fig. 4 show the effect of applying *MimicProp* on the four sets of lexicon words (Valence, LIWC, Harm/Care, and Authority/Subversion). Using the tSNE visualization algorithm (Maaten and Hinton 2008), we show the networks of the lexicon words *before* and *after* applying *MimicProp* on the left and right panels, respectively. In each network, nodes are lexicon words with green and red colors indicating the polarities (green: positive or virtue; red: negative or vice). The plots on the left show lexicon words with positions determined by their initial embedding vectors, and the plots on the right with positions determined by the new embedding vectors learned from *MimicProp*. The positioning of the words is determined

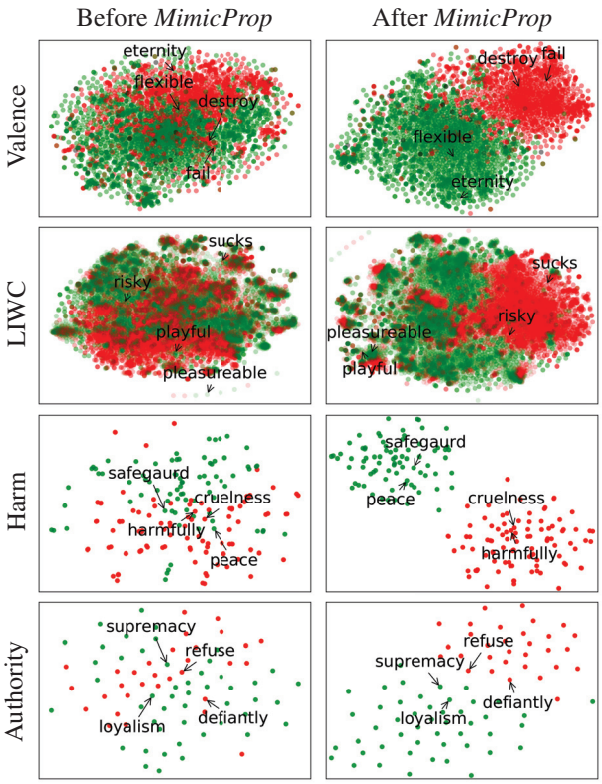

Figure 4: Visualizing the effect of *MimicProp* on the embeddings of lexicon words.
Four sets of lexicon words (Valence, LIWC, Harm/Care, and Authority/Subversion) are visualized using the tSNE algorithm with colors indicating the lexicon categories (green: positive or virtue; red: negative or vice). **Left:** words with initial embedding vectors. **Right:** words with new embedding vectors learned from *MimicProp*.

Table 3: Examples of words and their attributes. For each lexicon, we select both lexicon and out-of-lexicon words (with a learned score "NA") from both polarities. The scores are normalized to [-1, 1] for the real-valued attribute (Valence) and {-1,1} for the binary-valued attribute (other three lexicons).

(a) Valence

Word	Lexicon	Learned
starvation	-0.88	-0.96
egregious	NA	-0.68
favorite	0.68	0.68
☺	NA	0.48

(b) LIWC

Word	Lexicon	Learned
remorse	-1.00	-0.82
idiocy	NA	-0.69
joybird	1.00	0.64
gift	NA	0.23

(c) Harm/Care

Word	Lexicon	Learned
suffered	-1.00	-0.81
murdered	NA	-0.56
safeguarding	1.00	0.84
innovation	NA	0.47

(d) Authority/Subversion

Word	Lexicon	Learned
protest	-1.00	-0.92
distracting	NA	-0.44
leadership	1.00	0.62
sponsor	NA	0.52

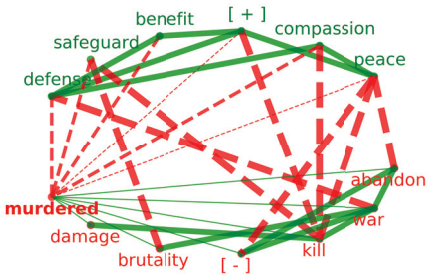

Figure 5: Visualizing the lexicon augmented graph. This graph contains ten words from the Harm/Care lexicon, a new word "murdered," and the two pseudo poles. Words are colored in green/red to indicate Care/Harm and are connected by edges with thickness reflecting the edge weights. The edges of the new word "murdered" are created by mimicking its closest lexicon neighbors, based on which its attribute value (leaning to Harm) is inferred.

by the tSNE algorithm based on the pairwise distances between the corresponding word embedding vectors. It can be seen from the plots that, after applying *MimicProp* (on the right), words are easily separable by their lexicon categories, compared with those with initial embedding (on the left). We label several example words on the plots. For example, the word "fail" (low valence; colored in red) in the Valence lexicon is originally close to other high-valence words (in green), and is moved to the red cluster after its embedding altered by *MimicProp*. The visualization clearly shows that the new embedding produced by *MimicProp* well captures the lexicon information as expected.

We use the network assortativity (correlation) measure (Newman 2003) to quantify the association of word attributes and the distance among them in the embedding space. We find a significant increase in the degree of assortativity after applying *MimicProp*. The network assortativity increases by 83% for Valence network (before: $r = 0.379, p < 10^{-6}$; after: $r = 0.695, p < 10^{-6}$; using bootstrap p-value calculation), 28% for LIWC (before: $r = 0.590, p < 10^{-6}$; after: $r = 0.755, p < 10^{-6}$), 164% for Harm/Care (before: $r = 0.327, p < 10^{-6}$; after: $r = 0.863, p < 10^{-6}$), and 202% for Authority/Subversion (before: $r = 0.267, p < 10^{-6}$; after: $r = 0.807, p < 10^{-6}$). This provides evidence that, even *MimicProp*'s goal is not merely to optimize the network distance according to the node attributes, the adjusted embeddings are more interpretable as they are more aligned with the given node attributes (words with similar attributes become closer).

Case study of words and their attributes. We show example words and their inferred lexicon scores to illustrate the quality of the learned attribute values. Table 3 lists four sets of words for the four lexicons. Each set includes both lexicon and out-of-lexicon words from two polarities. The out-of-lexicon words have no lexicon scores (indicated as "NA"). All scores are normalized to have the range of [-1,1] to facilitate comparison. For the binary-attribute value including LIWC, Harm/Care, and Authority/Subversion, the lexicon scores are exactly one of {-1,1} and the learned scores are within [-1,1]. The words are chosen from those among the highest (lowest) inferred scores. For the lexicon words, by comparing the original lexicon scores with the learned scores, we can see the closeness and correct sign in the learned scores. For the out-of-lexicon words, the learned scores are found reasonable. For example, the word "egregious" (meaning outstandingly bad) is given a low inferred valence score by *MimicProp*, and the emoticon of a smiling face is given a high inferred valence score as anticipated.

Visualizing the Augmented Graph. We further visualize an augmented graph containing a small set of words to illustrate how lexicon information is cast from lexicon words to non-lexicon words. Fig. 5 shows ten words from the Moral lexicon of Harm/Care and a new (out-of-lexicon) word "murdered." Words are colored in green or red to indicate the polarities (green: Care; red: Harm). We also show the two pseudo poles. Words are connected by edges with the edge thickness reflecting the edge weights. Specifically, the weights of edges between lexicon words are given by Eq. 1 and the weights of edges connected to the new word, i.e., the augmented edges, are given by Eq. 3. Given the positive and negative edge weights, the new word "murdered" is inferred to be in the Harm category after applying *MimicProp*, which is a desirable result.

6 Conclusion

This paper presents a novel approach, *MimicProp*, to create word representations that address the limitations of existing lexicon-based methods and word embeddings. *MimicProp* allows human knowledge encoded in lexicons to be fused into any pre-trained embedding that captures the semantic information. By introducing signed edges and a mimicking propagation process, *MimicProp* is able to infer the lexicon attribute for any given words – whether in the lexicon or

not – with an altered word embedding that incorporates the lexicon knowledge. The altered embedding can then be used to improve text analysis tasks.

There are some limitations to our current work. First, *MimicProp* was designed to incorporate the lexicon information from a single lexicon set. While it is easy to concatenate multiple embeddings learned separately from multiple lexicons, future work should consider ways to fuse multiple lexicon sources as well as the possibly different contexts surrounding a word. Second, currently the trade-off between the semantic information and the lexical attributed knowledge is empirically determined and has a global impact on all words. Future work may consider a parameter that can be learned from the data.

Despite these limitations, we have shown through extensive experiments that our approach outperforms the state-of-the-art methods in both lexicon attribute reconstruction and inference. Moreover, we demonstrated in several text analysis tasks (such as the sentiment classification in social media posts) that our lexicon-aligned word embedding can be effectively used to improve the task performance [6].

7 Acknowledgments

The authors would like to acknowledge the support from the DARPA UGB and AFOSR awards. Any opinions, findings, and conclusions or recommendations expressed in this material do not necessarily reflect the views of the funding sources.

A Appendix

Embedding Propagation Algorithm We detail the derivation of our solution in Section 3.3. In the Eq. 4, the first term can be simplified as:

$$\begin{aligned}&\tfrac{1}{2}\textstyle\sum_i\sum_j|w_{ij}|(\hat{\mathbf{x}}_i^2 d_{ii}^{-1}+\hat{\mathbf{x}}_j^2 d_{jj}^{-1})\\&-\textstyle\sum_i\sum_j|w_{ij}|\cdot(sgn(w_{ij})\hat{\mathbf{x}}_i\hat{\mathbf{x}}_j d_{ii}^{-1/2}d_{jj}^{-1/2})\end{aligned}\tag{5}$$

The two terms of $\hat{\mathbf{x}}_i^2 d_{ii}^{-1}$ and $\hat{\mathbf{x}}_j^2 d_{jj}^{-1}$ in Equation 5 are equivalent after summation, thus the Equation 5 is consequently equivalent to:

$$\begin{aligned}&\textstyle\sum_i\sum_j|w_{ij}|(\hat{\mathbf{x}}_i^2 d_{ii}^{-1})\\&-\textstyle\sum_i\sum_j w_{ij}\hat{\mathbf{x}}_i\hat{\mathbf{x}}_j d_{ii}^{-1/2}d_{jj}^{-1/2}\end{aligned}\tag{6}$$

Since $d_{ii}=\sum_j|w_{ij}|$, the first term in Equation 6 can be further simplified as $\sum_i\hat{\mathbf{x}}_i^2$. Now we have the cost function simplified as:

$$\begin{aligned}&C(\hat{\mathbf{X}})=\\&\textstyle\sum_i\hat{\mathbf{x}}_i^2-\sum_i\sum_j w_{ij}\hat{\mathbf{x}}_i\hat{\mathbf{x}}_j d_{ii}^{-1/2}d_{jj}^{-1/2}\\&+\mu\textstyle\sum_i^n\|\hat{\mathbf{x}}_i-\mathbf{x}_i\|^2\end{aligned}\tag{7}$$

The above equation can be re-written as:

$$\begin{aligned}&C(\hat{\mathbf{X}})=\\&\hat{\mathbf{X}}^T\mathbf{I}\hat{\mathbf{X}}-\hat{\mathbf{X}}^T\mathbf{D}^{-1/2}\mathbf{W}\mathbf{d}^{-1/2}\hat{\mathbf{X}}+\mu\left\|\hat{\mathbf{X}}-\mathbf{X}\right\|^2\\&=\textstyle\sum^d(\hat{\mathbf{X}}^T\mathbf{I}\hat{\mathbf{X}}-\hat{\mathbf{X}}^T\mathbf{S}\hat{\mathbf{X}}+\mu\left\|\hat{\mathbf{X}}-\mathbf{X}\right\|^2)\\&=\textstyle\sum^d(\hat{\mathbf{X}}^T(\mathbf{I}-\mathbf{S})\hat{\mathbf{X}}+\mu\left\|\hat{\mathbf{X}}-\mathbf{X}\right\|^2)\end{aligned}\tag{8}$$

[6] https://github.com/picsolab/MimicProp

Differentiate the cost fuction with repect to $\hat{\mathbf{X}}$, we have: $\frac{\partial C}{\partial\hat{\mathbf{X}}}=2(\mathbf{I}-\mathbf{S})\hat{\mathbf{X}}+2\mu(\hat{\mathbf{X}}-\mathbf{X})$. Let $\alpha=\frac{1}{1+\mu}$, then the derivative above can be re-written for each dimension in the word representation as: $\hat{\mathbf{X}}-\alpha\mathbf{S}\hat{\mathbf{X}}-(1-\alpha)\mathbf{X}$,

where α controls the strength of the graph constraint in the training iterations. Let $\frac{\partial C}{\partial\hat{\mathbf{X}}}=0$, we can solve $\hat{X}$ by:

$$\hat{\mathbf{X}}=(\mathbf{I}-\alpha\mathbf{S})^{-1}(1-\alpha)\mathbf{X}\tag{9}$$

As shown in (Zhou and Schölkopf 2004), the iterative solution can be given by:

$$\hat{\mathbf{X}}^{(t+1)}=\hat{\mathbf{X}}^{(t)}-\frac{\partial C}{\partial\hat{\mathbf{X}}^{(t)}}=\alpha\mathbf{S}\hat{\mathbf{X}}^{(t)}+(1-\alpha)\mathbf{X}\tag{10}$$

Balancing the positive and negative edges in the augmented propagation graph The balance of positive and negative edges connecting to each node can be achieved as follows. (i) Based on the lexicon graph construction procedure described in Sec. 3.1, each lexicon word has an equal number of positive and negative edges. (ii) In the node mimicking procedure, each unseen word "mimics" its closest lexicon words by copying their equal numbers of positive and negative edges and connecting to their neighbors. (iii) There might be some overlap among the neighborhoods of the lexicon words being mimicked. We assume the overlapping ratios for the positive and the negative edges are roughly similar. Therefore, each non-lexicon word u will have a roughly equal number of nonzero positive entries and negative entries in the adjacency vector $\mathbf{w}_u$, and will result in a balanced number of positive and negative entries after sparsification. This assumption works well empirically. We did not encounter an imbalanced distribution of positive and negative edges in all the datasets in our experiment. To further deal with the situation where the above assumption doesn't hold, additional step can be added to guarantee the balanced distribution: in the sparsification, m' edges are sampled where $m'=min(\frac{1}{2}dgr, m^+, m^-)$, with m^+ and m^- denoting the numbers of positive/negative entries in $\mathbf{w}_u$.

Details of neural network architectures in the NLP tasks We adopt the neural networks that have been reported in the literature to have the best performance in related tasks.

Specifically, for the Bi-LSTM model in SST and Moral classification, we train a model that consists of an embedding layer with $size=300$ (to work with Word2Vec embeddings and SSWE embedding), a Bi-LSTM layer followed by a drop-out layer, and a dense layer as output. For the Bi-LSTM in Valence regression, we adopt the *NTUA-SLP* model developed by Baziotis et.al (2018). This model achieves the best single-model performance in the SemEval 2018 valence regression task. It contains an embedding layer (of 300 dimensions, to work with Word2Vec and SSWE embedding), a Bi-LSTM layer, two attention layers, and a dense layer for the regression.

We use the pre-trained BERT encoder (Devlin and Toutanova 2018) that by default consists of one WordPiece embedding layer of 768 dimensions, and 6 layes of Transformers (Vaswani and Polosukhin 2017). We append a dense

layer to the encoder for the classification/regression tasks. The encoder is initialized with pre-trained weights released by Google [7], and later fine tuned independently in specific tasks.

References

Abboute, Amayas; Boudjeriou, Y. E. G. A. J. . B. S., and Poncelet, P. 2014. Mining twitter for suicide prevention. In *In NLDB*, 250–253. Springer.

Addawood, Aseel; Badawy, A. L. K., and Ferrara, E. 2019. Linguistic cues to deception: Identifying political trolls on social media. In *In ICWSM*, volume 13, 15–25.

Baziotis, Christos; Nikolaos, A. C. A. K. A. P. G. E. N. N. S., and Potamianos, A. 2018. Ntua-slp at semeval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning. In *In SemEval*, 245–255.

Bengio, Yoshua; Delalleau, O., and Le Roux, N. 2006. 11 label propagation and quadratic criterion.

Bradley, M. M., and Lang, P. J. 1999. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Citeseer.

Chen, Hao; McKeever, S., and Delany, S. J. 2019. The use of deep learning distributed representations in the identification of abusive text. In *In ICWSM*, volume 13, 125–133.

Devlin, Jacob; Chang, M.-W. L. K., and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Dhaoui, Chedia; Webster, C. M., and Tan, L. P. 2017. Social media sentiment analysis: Lexicon versus machine learning. *J Consum. Market.* 34(6):480–488.

Dhingra, Bhuwan; Zhou, Z. F. D. M. M., and Cohen, W. 2016. Tweet2vec: Character-based distributed representations for social media. In *In ACL*, 269–274.

Faruqui, Manaal; Dodge, J. J. S. K. D. C. H. E., and Smith, N. A. 2015. Retrofitting word vectors to semantic lexicons. In *In NAACL*, 1606–1615.

Fu, Peng; Lin, Z. Y. F. W. W., and Meng, D. 2018. Learning sentiment-specific word embedding via global sentiment representation. In *In AAAI*.

Garten, Justin; Hoover, J. J. K. M. B. R. I. C., and Dehghani, M. 2018. Dictionaries and distributions: Combining expert knowledge and large scale textual data content analysis. *Behav. Res. Methods* 50(1):344–361.

Graham, Jesse; Haidt, J., and Nosek, B. A. 2009. Liberals and conservatives eely on sifferent sets of moral foundations. *J Pers Soc Psychol* 96(5):1029.

Hamilton, William L; Clark, K. L. J., and Jurafsky, D. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *In EMNLP*, volume 2016, 595. NIH Public Access.

Johnson, K., and Goldwasser, D. 2018. Classification of moral foundations in microblog political discourse. In *In ACL*, 720–730.

Jurek, Anna; Mulvenna, M. D., and Bi, Y. 2015. Improved lexicon-based sentiment analysis for social media analytics. *Secur. Inform.* 4(1):9.

Liao, Lizi; He, X. Z. H., and Chua, T.-S. 2018. Attributed social network embedding. *In TKDE* 30(12):2257–2270.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *In JMLR* 9(Nov):2579–2605.

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

Mikolov, Tomas; Chen, K. C. G., and Dean, J. 2013b. Efficient estimation of word representations in vector space. *In Workshop at ICLR* 2013.

Mohammad, Saif; Bravo-Marquez, F. S. M., and Kiritchenko, S. 2018. Semeval-2018 task 1: Affect in tweets. In *In SemEval*, 1–17.

Newman, M. E. 2003. Mixing patterns in networks. *Physical Review E* 67(2):026126.

Pennebaker, James W; Boyd, R. L. J. K., and Blackburn, K. 2015. The development and psychometric properties of liwc 2015. Technical report.

Saad, Y. 2003. *Iterative Methods for Sparse Linear systems*, volume 82. siam.

Saha, Tanay Kumar; Joty, S. H. N., and Hasan, M. A. 2016. Dis-s2v: Discourse informed sen2vec. *arXiv:1610.08078*.

San Vicente, Inaki; Agerri, R., and Rigau, G. 2014. Simple, robust and (almost) unsupervised generation of polarity lexicons for multiple languages. In *In EACL*, 88–97.

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *Departmental Papers (CIS)* 107.

Socher, Richard; Perelygin, A. W. J. C. J. M. C. D. N. A., and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *In EMNLP*, 1631–1642.

Tai, Y.-J., and Kao, H.-Y. 2013. Automatic domain-specific sentiment lexicon generation with label propagation. In *In iiWAS*, 53. ACM.

Tang, Duyu; Wei, F. Y. N. Z. M. L. T., and Qin, B. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *In ACL*, volume 1, 1555–1565.

Vaswani, Ashish; Shazeer, N. P. N. U. J. J. L. G. A. N. K. Ł., and Polosukhin, I. 2017. Attention is all you need. In *In NIPS*, 5998–6008.

Velikovich, Leonid; Blair-Goldensohn, S. H. K., and McDonald, R. 2010. The viability of web-derived polarity lexicons. In *In NAACL*, 777–785.

Von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and computing* 17(4):395–416.

Warriner, Amy Beth; Kuperman, V., and Brysbaert, M. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behav. Res. Methods* 45(4):1191–1207.

Wu, Yonghui; Schuster, M. C. Z. L. Q. V. N. M. M. W. K. M. C. Y. G. Q. M. K., et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*.

Yang, Liner; Chen, X. L. Z., and Sun, M. 2017. Improving word representations with document labels. *IEEE Audio, Speech, Language Process.* 25(4):863–870.

Yu, Liang-Chih; Wang, J. L. K. R., and Zhang, X. 2017. Refining word embeddings for sentiment analysis. In *In EMNLP*, 534–539.

Zhang, Wei; Johnson, N. W. B., and Kuang, R. 2012. Signed network propagation for detecting differential gene expressions and dna copy number variations. In *In ACM-BCB*, 337–344. ACM.

Zhou, Dengyong; Bousquet, O. L. T. N. . W. J., and Schölkopf, B. 2004. Learning with local and global consistency. In *In NIPS*, 321–328.

[7] https://github.com/google-research/bert