# Affect Lexicon Induction For the Github Subculture Using Distributed Word Representations

by

Yuwei Jiao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2018

## Abstract

Sentiments and emotions play essential roles in small group interactions, especially in self-organized collaborative groups. Many people view sentiments as universal constructs; however, cultural differences exist in some aspects of sentiments. Understanding the features of sentiment space in small group cultures provides essential insights into the dynamics of self-organized collaborations. However, due to the limit of carefully human annotated data, it is hard to describe sentimental divergences across cultures.

In this thesis, we present a new approach to inspect cultural differences on the level of sentiments and compare subculture with the general social environment. We use Github, a collaborative software development network, as an example of self-organized subculture. First, we train word embeddings on large corpora and do embedding alignment using linear transformation method. Then we model finer-grained human sentiment in the Evaluation-Potency-Activity (EPA) space and extend subculture EPA lexicon with two-dense-layered neural networks. Finally, we apply Long Short-Term Memory (LSTM) network to analyze the identities' sentiments triggered by event-based sentences. We evaluate the predicted EPA lexicon for Github community using a recently collected dataset, and the result proves our approach could capture subtle changes in affective dimensions. Moreover, our induced sentiment lexicon shows individuals from two environments have different understandings to sentiment-related words and phrases but agree on nouns and adjectives. The sentiment features of "Github culture" could explain that people in self-organized groups tend to reduce personal sentiment to improve group collaboration.

## Acknowledgements

This thesis cannot be completed without the help from many people. I would like to take this opportunity to thank them all.

First and foremost, I would like to thank my supervisor, Professor Jesse Hoey. It was a great honour to work with him throughout my studies. Thanks to Professor Mei Nagappan and Professor Yaoliang Yu for being on my committee. Thanks to all my friends and labmates for their support. Last but not least, I would also thank my boyfriend Jun Zhao and my parents for supporting all the time.

## Dedication

This is dedicated to the one I love.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Sentiment is closely associated with human daily life and activities. It could be the consequence of single or multiple events, and also continuously affects human cognition and behaviors. It is commonly agreed that sentiments are complex states of feelings involving several components, including social environments, subject experiences and psychological changes. Understanding the underlying sentimental states of individuals or groups would be beneficial to many research areas as well as to industrial products. For example, intelligent home assistants could gather the user's sentiment and provide more customized service; leaders and managers in companies or organizations could do group collaboration evaluation with the help of individuals' sentiment analysis; sociologists and politicians could measure public reactions to newly published regulations.

In the THEMIS.COG project, we aim at modeling collaboration dynamics in distributed communities and providing insights into the social and psychological mechanisms of self-organized collaborative groups like Github. There are pieces of evidence that sentiments play an essential role in group collaboration [22, 4], as positive sentiments encourage people to contribute and innovate, while negative sentiments reduce the community's creativity and vitality. Previous work has shown great success in modeling and analyzing sentiments and interactions on Github [75, 76]; however, it ignores the sentimental divergence across communities and cultures. Sentiment is generally recognized as a universal construct and scientists believe it as genetically determined so that basic facial expressions and spoken pitch patterns are interpreted and perceived in the same way across regions and cultures. For example, smiling face represents friendliness while yelling at people shows anger. However, there exist subtle cultural differences in sentiment because the environment and social experiences also shape the way people feel and express [59].

So what is culture? Culture is made up of the shared patterns of behaviors and interactions, cognitive constructs and understandings that are learned through socialization. A subculture is a group of people within a larger culture who develop norms and maintain values that are different from the general culture. Github as a self-organized collaborative software development network could be recognized as a subculture for several reasons. First, people in this community are mainly programmers and know about coding. Second, they share the same goals of software and tool iteration and development. Third, there are specific rules to communicate using pull requests comments and commit messages. Usually, it is difficult to describe the sentimental features of a given subculture like Github because it requires quantities of human annotated data on images, texts or audio. However, such high-quality data would be expensive as well as hard to collect, since people are influenced by more than one subculture, and it's almost impossible to ensure the annotation mainly represents the subculture we want. Moreover, it is complicated to describe the sentimental states of individuals in a subculture, so it would be even more challenging to extract the general features of a group of people and compare horizontally across subcultures.

Word embedding [62, 60, 52] is a common used method in most recent natural language processing projects. Much work has been done in the field of semantic shifts and syntax changes across time and cultures. Austin Kozlowski et al. [49] and Nikhil Garg et al. [24] explore the linear relationship of word embeddings and demonstrate applications to analyze gender and class association in history. William Hamilton et al. [31, 32] propose a method for quantifying semantic change by evaluating word embeddings and prove the rate of semantic change is proportional to word frequency and polysemy. Aparna Garimella et al. [25] identify a set of words with significant usage difference between American and Australian personal writings and investigate language attributes and underlying psychology mechanisms across cultures. To best describe and compare group sentiments, we use the continuous three-dimensional Evaluation-Potency-Activity (EPA) space. Introduced by Osgood [69, 70], it is a vector space that describes the direction and magnitude of an affective meaning in three dimensions: Evaluation (pleasant/unpleasant), Potency (powerful/powerless) and Activity (active/inactive). There are several cross-culture empirical studies of EPA ratings. Each affect meaning is measured on a scale from -4.3 (infinitely negative) to +4.3 (infinitely positive). Studies have shown that most of the affective meanings are shared across culture and would not change significantly over time. However, specific social events could make a difference to a set of related affective meanings [40].

This thesis is mainly based on two hypotheses: first, that the cultural differences in language usage could be reflected in distributed word representations; second, that the affective meanings of sentiments vary from one subculture to another. In this thesis, we demonstrate a new method to induct and expand sentiment lexicons across subcultures. Moreover, we propose a framework to describe the sentimental features of subcultures and explore sentimental divergence across cultures.

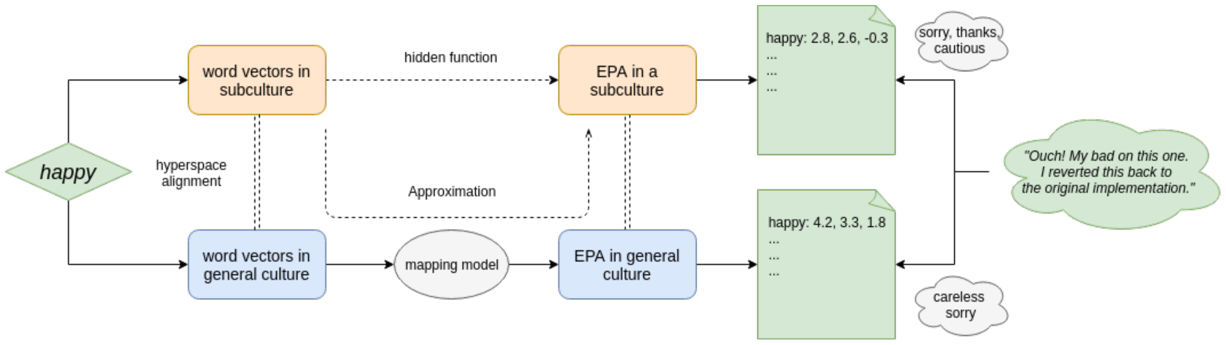


Figure 1.1: The architecture of our proposed approach. There are three components of the architecture. First is word vector hyperspace alignment (Chapter 3). Next is word vector to affective dimensions mapping models (Chapter 4.1). The final component is the applications our predictions and models (Chapter 5), where the green parts show how to predict words' affective meanings in subcultures and apply them to classify sentiments triggered by sentences.

## 1.2 Contributions and Outline

Figure 1.1 shows the architecture of our proposed approach in this thesis. The main contributions of this thesis are:

- Align two distributed word vector spaces using a statistical inference (singular-value decomposition) method and linear transformation with stochastic gradient descent modeling. The average cosine distance reduces from 1.0 to 0.6 by using the proposed alignment approach.

- Use distributed word representations to predict a sentiment lexicon using graph-based semi-supervised learning, support vector regression and artificial neural networks. The mean absolute errors of 13,790 concepts are 0.63, 0.70, 0.75 on three affective dimensions respectively, scaling from -4.3 to 4.3.

- Generate the sentiment dictionary of 39,366 concepts from general culture to a given subculture (Github community culture in this thesis). Evaluating on 371 concepts, the mean absolute errors are 0.57, 0.66, 0.58 on three affective dimensions in the same scales as above.

- Integrate the augmented sentiment lexicon with a sentence sentiment recognizer using Long Short-Term Memory (LSTM) model. The model achieves a 95% accuracy of predicting sentiments triggered by news headlines, but only 48% when performing on Github comments data.

- Demonstrate the sentiment divergence between the universal general culture and a self-organized collaboration community (Github). There are significant affective shifts of words related to sentiments and emotions.

The structure of this thesis is as follows. Chapter 2 provides background information and related work in recent years regarding natural language processing, sentiment analysis as well as culture difference in sentiment. Chapter 3 introduces a set of commonly used methods to align distributed high-dimensional word embeddings, including singular value decomposition (SVD) and linear transformation modeling with stochastic gradient descent. We evaluate the performance of these methods on several separately trained word vector models under different anchor words selection strategies. Within the aligned word vector space, we propose doing sentiment lexicon expansion for a subculture and discuss the sentimental meaning divergence on the level of words across subcultures in Chapter 4. More details about datasets, sampling methods, model selection, and evaluation metrics are provided in this chapter too. In Chapter 5, we apply the aligned distributed word representations and augmented sentiment lexicon to predict sentiments evoked from sentences using a LSTM network. Sentiment analysis results support the hypothesis that Github is a collectivist culture instead of individualist. Conclusion and future work are in Chapter 6.

# Chapter 2

# Related Work

## 2.1 Sentiment Analysis

### 2.1.1 Evaluation-Potency-Activity Model

There have been several theories trying to describe sentiments. A sociology theory called Affect Control Theory (ACT) uses a three-dimensional vector to describe sentiments. The basis vectors of the sentimental space in this theory are called Evaluation (pleasant/unpleasant), Potency (powerful/powerless) and Activity (exciting/calm). Affect Control theory proposes that individuals process words and events as symbols or concepts which are shared among groups of people. ACT represents social behaviors with an Actor-Behavior-Object model. It uses a three-dimensional EPA profile as the fundamental sentiments and proposes that it is important for individuals to maintain the transient impressions. Fundamental sentiments are representations of social objects, such as interactants identities, behaviors and environmental settings in EPA space. Transient impressions are the result of social events corresponding to fundamental sentiments. Sentiments arise because of the differences between fundamental sentiments and transient impressions. The equations have been obtained through empirical studies.

Usually, EPA profiles are measured through surveys where annotators rate identities and behaviors on a numerical scale from -4.3 (infinitely bad, powerful or inactive) to 4.3 (infinitely good, powerful or lively). People from a similar cultural background would reach an agreement about the EPA values, and the affect ratings are relatively stable over time. For example, a baby is seen as (1.63, -1.64, 0.3) which is described as good, quite weak and not very active. While the EPA value of a mother is (2.48, 1.96, 1.15) which means kind/good, much more powerful and active compared to a baby [39].

### 2.1.2 Hyperspace Alignmemt

High-dimensional distributed representations of words and phrases enable people to reason about new facts from texts, conversations, and knowledge graphs. The similarity of two entities could be defined by their cosine distances in this space. Usually, word vectors are trained online, where a text corpus is fed into the training network all at once. Online training ensures vectors across language, regions, time series and knowledge graphs are within the shared space. Mikolov et al. [61] show that word vectors can also be obtained offline. It enhances the feasibility and scalability to obtain sets of word vectors independently. Tomas Mikolov et al. first proposed the approach of using similarity of distributed word representations [61] as complementary to the existing statistical machine translation systems [47, 46, 30], which shows the geometric arrangement similarity of similar words (numbers and animals) in high dimensional vector space between English and Spanish. Taking the advantage of a linear transformation (rotation and scaling), they can infer missing dictionary entries of words and phrases through projections even between languages that are substantially different, for example, English and Chinese. Moreover, this method could also give a similarity score for each word pair. The linear projection method archives great success despite the simplicity. Following Mikolov's work [61], Samuel Smith et al. [82] prove further that the optimized linear transformation between two vector spaces is orthogonal. Therefore the alignment could be achieved using singular value decomposition (SVD) instead of using SGD. The SVD method is more robust to noise compared to SGD, and they improve the precision from 34% in previous work to 43% by introducing inverted softmax to identify translation pairs.

### 2.1.3 Sentiment Lexicon Induction

One of the most popular methods of sentiment analysis tasks is to use sentiment lexicons that associate words with their polarity, either positive or negative. Such lexicons are composed of a relatively small set of words with semantic orientation (valence), or in multi-dimensional space of evaluation (valence), potency (dominance), and activity(arousal). Osgood first carried out a large set of cross-cultural studies and introduced the first affective meaning dictionary [68, 70, 69]. Based on this work, Heise further measured the affective ratings in several countries [37, 38, 39].

There are two primary methods used for polarity acquistion: corpus-based methods and thesaurus-based method. The corpus-based methods utilize a small set of labeled words and induce the sentiment lexicon based on co-occurrence statistics from the corpus. Some general co-occurence features include term frequency-inverse document frequency (TF-IDF), SVD [88], syntactic information [34], etc. People also use statistics gathered online to create lexicon for specific domain. Astudillo et al. [5] proposed a regression model to create a Twitter sentiment lexicon using skip-gram word embeddings. Rothe et al. [77] transformed word embeddings to lower dimensional representations by training a gradient descent algorithm with two objective functions. Fast et al. [20] combined skip-gram word embeddings with crowd-sourcing annotations to map words into 200 predefined categories. The thesaurus-based methods rely on lexical relationships in WordNet and other human-annotated resources, for example, WordNet [63], General Inquirer [84], etc. Kamps et al. [45] built a lexical network of snonyms and antonyms and computed the sentiment orientation by measuring the relative distances between words and seeds (e.g., bad and good). Rao et al. [74] presented extensive evaluations of label propagation methods using dictionay, copurs, and graph to induce words polarity. Joseph et al. [43] futher compared the relationships of senmantic similarity and the identities of annotators (e.g., student, mother). A number of studies use graph-based learning methods to induce sentiment polarity. The general idea is to use a few already labeled data to label a set of unlabeled data within a graph, which encodes the relationships between words. Velikovich et al. proposed to build co-occurance frequency graph and induce polarity lexicons [89]. A few researchers explore to expand multi-dimensional sentiment lexicons. Kamps et al. [45] extended a three-dimensional sentiment lexicon based upon WordNet similarities, focused mainly on adjectives.

### 2.1.4 Affective State Prediction

Sentiments have been studied extensively in many fields like computer science, psychology, and sociology in recent years. Although there is psychological evidence showing sentiments are perceived in more than one dimension, most proposed theories adopt discrete appraisal theory by representing sentiments with discrete labels, such as positive, negative, neutral, or happy, sad. Some commonly used features in sentiment analysis include part-of-speech tagging, n-grams, term weighting, sentiment lexicons, and syntactic dependencies [71]. Unsupervised learning approaches usually calculate the difference between the point-wise mutual information (PMI) and determine the semantic orientation of sentence or document. Advanced approaches build more structured models by taking semantic taxonomy, the interaction of words or bag-of-words techniques into consideration. A framework is proposed by Coecke et al. which constructs a sentence vector as a function of its word vectors [14]. There are several recent studies tackled to predict readers' sentiments instead of from the writers' perspective.

Alm et al. [3] annotated and analyzed sentiments in a corpus of children's stories. They utilized Ekman's basic sentiments [18] and concluded that the sentiments in fairy tales often start with neutral sentiments and end with happy one. Lin et al. [56] proposed their method of using bigrams features, affix similarities, and text metadata to classify Chinews news articles. Bhowmick et al. [8] presented a multi-label classification method to classify news articles into multiple sentiment classes (e.g., disgust, happiness, and sadness) with an accuracy of 77%. Kozareva et al. [48] presented an unsupervised learning method that computes PMI scores by querying search engines. Ahothali and Hoey explore predicting sentiments for each news headline using an augmented EPA lexicon and ACT equations [1]. They decompose sentence into subject-verb-object and associate subject with the actor, verb with the behavior and object with the object in ACT. MingLei Li et al. further propose to use LSTM learning method using word embeddings to predict the affective states of a described event [54]. They claim the method outperform the linear model in the ACT and most importantly, there is no need to construct affect lexicons manually.

### 2.1.5 Sentiment Analysis on Github Data

Github [1] is a collaborative code hosting site built on top of the git version control system. It integrates social features which allow developers to communicate and collaborate. The main feature of Github is the fork and pull model, which enables users to create a copy of the repository and submit the pull requests to the project master branch. Human activities evoke sentiments and have positive or negative effects on group collaborations. It is a challenging task to determine sentiments in open source projects.

There has been some work done in this direction. Guillory et al. show the possibility of sentiment analysis on mailing lists and discussion boards [28]. Murgia et al. also perform a feasibility study of sentiment mining using issue reports of Apache software foundation [66]. Their work is based on Parrott's sentiment framework [72] that classifies six basic sentiments into love, sadness, anger, joy, surprise, and fear. Guzman et al. analyze 60,425 Github commit comments [29] with the help of a lexical sentiment extraction tool called *SentiStrength*, which can rate comments within the range of [-5, 5]. However, the tool only analyzes the polarity (positive, negative or neutral) without a finer granularity of sentiments. Pletea et al. explore sentiment analysis on software security discussions

[1]https://github.com/

on Github [73]. They mainly use the NLTK library to predict the probability of being positive, neutral or negative.

However, Lin et al. exploit commonly used tools to identify the sentiment of software engineering related texts in their studies [55], and their results show none of the sentiment analysis tools are ready for real usage in software engineering related discussions yet. In their studies, they focus on three software engineering datasets, categorize sentiments as positive, neutral and negative, and experiment with six state-of-art sentiment analysis tools. The three software engineering datasets are from Stack Overflow, issue tracker comments, and mobile application reviews. The analysis of results reveals that all the experimented tools fail to discriminate between positive/negative sentences with neutral ones. Therefore, the tools are not reliable at all. As a conclusion, they suggest we should be more careful when doing sentiment analysis and opinion mining in software engineering in practice.

## 2.2 Natural Language Processing

### 2.2.1 Bag of Words

The bag-of-words model is a simplifying term used in natural language processing area [90]. In this model, a sentence or a document is represented as a multiset of its words. A common problem with text is that the text is messy, and machine learning algorithms prefer well-defined input and output format. A bag-of-words model extracts features from text for use, and discards any information about the order and structure of the document. The basic intuition is that, documents are similar if they share same content. Despite the simlicity, the bag-of-words model suffers from some shortcomings. For example, it requires careful-designed vocabulary and word representations are usually sparse vectors in large dimensional space.

### 2.2.2 Distributed Word Representation

Distributed word representations represent words in dense, continuous, high-dimensional space. These vector space models, know as word embeddings, have embedded much information of interest and attracted much attention among computer scientists and computational linguists in recent years. One of the major features of word embeddings is the ability to capture complex semantic relations between words. In word embedding models, words sharing similar contexts would be positioned nearby in the vector space.

The *word2vec* is the most widely used word embedding algorithm nowadays, which uses a shallow, two-layered neural network architecture to optimize the prediction of context words. There are two distinct network architectures under word2vec: continuous bag-of-words (CBOW) and skip-gram (SG). Under the skip-gram architecture, the model predicts context words given a center word in a sliding window of k words. While the CBOW architecture works similarly, except that predicts the center word given k context words. Several other approaches take the statistical analysis into account to generate a word vector position, for example, n-grams. The word2vec model ignores all words with total frequency lower than *min_count*, so the sliding window is the k context words among the surviving words.

Past work with word embeddings also show semantic relations between word pairs. For example, the analogy "man is to woman as king is to $\underline{\text{queen}}$" can be solved by computations on word embeddings trained with sufficient text. Computer scientists believe that word embeddings are promising tools for cultural analysis [32, 31]. It reveals the fundamentally relational nature of their meanings. We also believe that word embeddings could shed light upon sentiment analysis and explore this direction in this thesis.

### 2.2.3 Word Similarities

In general, there are two types of word similarities used in natural language processing tasks, which are attributional and relational similarity. The attibutional similarity measures the degree of correspondence between attributes of words, while the relational similarity measures the correspondence between internal relationships. The term "semantic similarity" used in this thesis mainly refers to the attributional similarity by calculating the Eucliden or cosine distances between word embedding vector pairs.

# Chapter 3

# Alignment Word Space

## 3.1 Introduction

Distributed representations of words and phrases in high dimensional space enable people to reason about new facts from texts, conversations, and knowledge graphs, where the similarity of two entities could be defined by their cosine distances in this space. For example, bilingual machine translators utilize word embeddings to describe semantic associations between word pairs across language. Targeting knowledge graph completion usually designs a set of vector operations for reasoning and assertion. Linguistic change detectors can construct time series of distributional vectors for each word and track its linguistic displacement over time.

Usually, word vectors are trained online, where a text corpus is fed into the training network all at once. Online training ensures vectors across language, regions, time series and knowledge graphs are within the shared space. Mikolov et al. [61] show that word vectors can also be obtained offline. It enhances the feasibility and scalability to obtain sets of word vectors independently. However, because independently obtained word embeddings vary in spatial orientation in this case, an additional step of vector space alignment is needed to ensure vectors across language, time serious, knowledge graphs, or region are within in the shared space.

Usually, two simplifying assumptions are made to aid the alignment process: first, the spaces are assumed to be equivalent under linear transformation [51]; second, the meaning of most words is assumed to be stable across different contexts. Vivek Kulkarni et al. [51] align the embeddings in one unified coordinate system by learning a linear transformation matrix $\boldsymbol{W_{t'\mapsto t}(w)}$that maps a word for each time snapshot. The matrix minimizes average distance between each anchor word and $k$ nearest words in the embedding space. However, the objective function relies heavily on synonyms, and the researchers fail to provide more information on the transformation evaluation. The method is not feasible for alignment tasks in semantic analysis. Tomas Mikolov et al. [61] share the same idea, but instead of using the $k$ nearest words in space, they utilize the publicly available *WMT11* datasets and claim to achieve over 90% precision in an English-Spanish translation task. Samuel Smith et al. [82] further prove the linear transformation between two spaces is orthogonal and could be obtained using the singular value decomposition (SVD). They extend the method to bilingual translation from English sentences to Italian with a precision of 68%. Zhen Wang et al. and Huaping Zhong et al. [92, 95] propose a new alignment model based on text descriptions of entities. The model consists three components: a knowledge model embedding entities and relations describing the plausibility of a triplet; a text model embedding the statistical attributes of pairs of words, for example, the probability of co-occurrence; and an alignment model. The aligned word vector model achieves more than 80% hit rates on semantic and syntactic reasoning tasks.

In this thesis, we explore using the linear transformation method and singular value decomposition (SVD) method to do the alignment of word vectors obtained from Github datasets to the pre-trained Google News word vectors. We denote the learned alignment model as alignment-model as a reference. We use two methods to construct the anchor word sets. One is to pick a set of words at random from the intersected word lists, the other is to construct anchor word datasets manually from stop words like *what, the, an, some* under the assumption that stopwords shift little across context. We perform extensive experiments on semantic and syntactic reasoning task before and after the alignment. The average of word pairs distance is more than 1.0 before the alignment and reduces below 0.6 afterwards. Furthermore, we demonstrate the semantic shifts across subculture captured by our algorithm.

## 3.2 Background

Tomas Mikolov et al. first proposed the approach of using similarity of distributed word representations [61] as complementary to the existing statistical machine translation systems [47, 46, 30]. This method makes only a few assumptions, so it is feasible to apply to any texts. Figure 3.1 illustrates the mechanism of the method. It shows the geometric arrangement similarity of similar words (numbers and animals) in high dimensional vector space between English and Spanish. Taking the advantage of a linear transformation (ro-

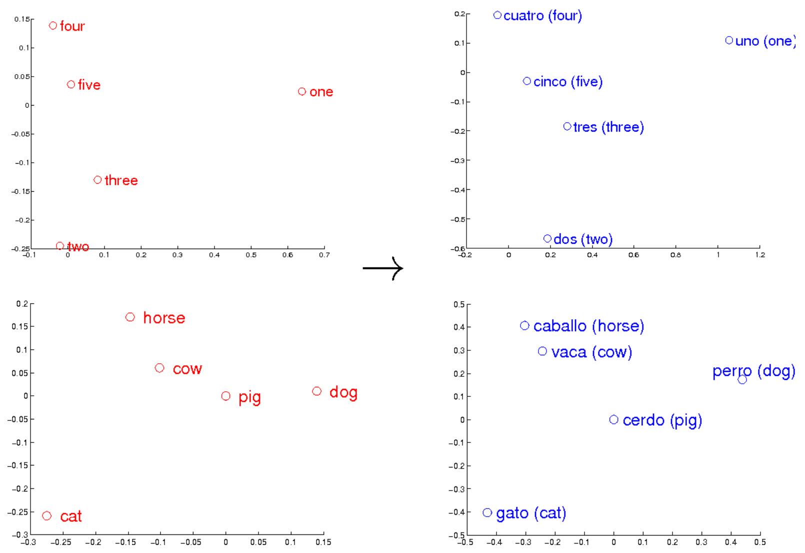


Figure 3.1: Simple 2D visualization of distributed word vector representations of numbers and animals in English (left) and Spanish (right) [61].

tation and scaling), they can infer missing dictionary entries of words and phrases through projections even between languages that are substantially different, for example, English and Chinese. Moreover, this method could also give a similarity score for each word pair. The linear projection method archives great success despite the simplicity.

Suppose there are a set of word pairs and their word vector representations obtained using word2vec models (either skip-gram or continuous bag-of-words model) as $\{x_i, z_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^{d_1}$ and $z_i \in \mathbb{R}^{d_2}$ is the distributed representation of word $i$ in target and source language respectively. The objective of the linear projection is to learn a transformation matrix $W$ such that $Wx_i$ approximates $z_i$ in vector space. They formalize the problem as an optimization task:

$$\min_W \sum_{i=1}^{n} \|Wx_i - z_i\|^2$$

where stochastic gradient descent (SGD) is utilized to solve the equation.

Following Mikolov's work [61], Samuel Smith et al. [82] prove further that the optimized linear transformation between two vector spaces is orthogonal. Therefore the alignment could be achieved using singular value decomposition (SVD) instead of using SGD. The SVD method is more robust to noise compared to SGD, and they improve the precision from 34% in previous work to 43% by introducing inverted softmax to identify translation pairs. The inverted softmax $P_{j\to i}$ is the inverted calculation of finding target words, and it provides a confidence estimation for the space alignment. Alexey Zobnin applies the SVD method to the Russian language and re-learns word embeddings to improve component stability [98].

The process of orthogonal transform action is performed by defining similarity matrix $S = YWX^T$, where $X, Y \in \mathbb{R}^{n\times d}$ are word vector matrices in source and target space respectively. The vocabulary size is denoted as $n$ and the dimension of word vector space is denoted as $d$. Therefore each entry of similarity matrix is:

$$\begin{aligned} S_{ij} &= y_i^T W x_j \\ &= y_i \cdot \ (W x_j) \end{aligned} \tag{3.1}$$

The largest value in a column represent the most similar source word for each target word, while the largest value in a row is the most similar target word for each source word. Then they form a second similarity matrix $S' = XQY^T$ so that the matrix $Q$ maps target word back to the source.

$$\begin{aligned} S'_{ji} &= x_j^T Q y_i \\ &= x_j \cdot \ (Q y_i) \end{aligned} \tag{3.2}$$

To keep self consistency, they require that $S' = S^T$. Since $S^T = XW^TY$, there is $Q = W^T$. It is natural that mapping between source and target word vector space is reversible, and $x \sim W^T y$, $y \sim Wx$. Thus $x \sim W^T W x$ holds for any word vector $x$ and it is concluded the transformation matrix $W$ is an orthogonal matrix $O$ satisfying $O^T O = I$. After normalizing word vectors $x$ and $y$, the objective is to optimize the cosine distance between possible pairs:

$$\max_O \sum_{i=1}^{n} y_i^T O x_i, \text{subject to } O^T O = I, \tag{3.3}$$

which is equivalent to the "orthogonal Procrustes problem" [81]:

$$\min_O \sum_{i=1}^{n} \|y_i - O x_i\|^2, \text{subject to } O^T O = I. \tag{3.4}$$

The proposed solution in their paper [82] is to form two ordered dictionary pair matrices $X_D, Y_D$ (where $X_D, Y_D$ are ordered $X, Y$), by computing the SVD of

$$\begin{aligned} M &= Y_D^T X_D \\ &= U \Sigma V^T, \end{aligned} \tag{3.5}$$

and minimizing with:

$$O = UV^T, \tag{3.6}$$

the optimized similarity matrix becomes as the following:

$$S = YUV^T X^T \tag{3.7}$$

$$\begin{aligned} S_{ij} &= y_i^T UV^T x_j \\ &= (U^T y_i) \cdot (V^T x_j). \end{aligned} \tag{3.8}$$

Therefore they could map from source space to target by applying the transformation matrix $UV^T$ to the source language matrix. This method is highly efficient and is proved to get a translation precision on 200k Italian sentences corpus with a precision of 68% [82].

## 3.3 Method

In this thesis, we mainly explore the methods of using linear transformation with stochastic gradient descent and singular value decomposition to the language matrix. Here we first give a brief introduction of the two algorithms. Also, we describe our strategy of selecting word pairs from two domains.

### 3.3.1 Linear Transformation

In mathematics, linear transformation $T : V \to W$ is an important rule to make geometric changes between two subspaces while preserving the operations of addition and scalar multiplication. Basically, linear transformation satisfies the following two conditions:

- $T(\vec{u} + \vec{v}) = T(\vec{u}) + T(\vec{v})$ for all vectors $\vec{u}$, $\vec{v}$ in space $V_1$,
- $T(c\vec{u}) = cT(\vec{u})$ for all vectors $\vec{u}$ in space $V_1$ and a scalar $c$.

The linear transformation is useful because most geometric operations are linear transformations, including rotations, scaling, and reflections. The artificial neural network (ANN) takes the idea of linear transformation by producing the linear combinations of input signals and weight matrix as output. Moreover, neural network introduces bias inputs, nonlinear functions and so on. Despite its simplicity, a linear transformation still acts as an essential component in the architecture of modern machine learning and deep learning.

### 3.3.2 Stochastic Gradient Descent

Usually, gradient descent (GD) is used to minimize the least squared error by updating coefficients in linear regression and weights in neural networks. Learning rate refers to the size of steps specified by the negative gradient. With a high learning rate, the algorithm can cover more parameter space quickly under the risk of missing the optimal state. And with a low learning rate, the algorithm scans over the space more precisely with more iterations. Therefore, the learning rate can vary as a function of the number of iterations and decay rate, so that the algorithm updates quickly at the beginning and precisely when approaching the optimal state.

The main problem in gradient descent is that calculating the parameters in each iteration would be expensive, especially when the data is large. Therefore, the basic idea of stochastic gradient descent is to optimize by sampling on the training dataset. The result after each iteration is an estimation towards the globally optimal state. With stochastic gradient descent, the algorithm does not compute the exact derivative of loss function. Instead, it is estimating it on a small batch of data. To get a better estimation of the optimal direction, momentum is used as the moving average of gradients. In this thesis, we set the learning rate as 0.1, the decay rate as $1.0 * 10^{-6}$ and momentum as 0.9.

### 3.3.3 Singular Value Decomposition

Singular value decomposition is a factorization of the matrix in linear algebra. It has many useful applications in statistics and signal processing. Here is the statement of singular value decomposition over rectangular matrix $M \in R^{m \times n}$:

$$M = U \Sigma V^*$$

where

- $U$ is a $m \times m$ unitary matrix over field $K$.
- $\Sigma$ is a diagonal $m \times n$ matrix with non-negative values on the diagonal.
- $V$ is a $n \times n$ unitary matrix over $K$ and $V^*$ is the conjugate transpose of $V$.

Therefore, the geometric meaning of the SVD theorem can be concluded as finding the orthonormal basis of two linear spaces. It is widely used in the field of unsupervised machine learning, so we can decompose a matrix into lower rank matrices without losing too much information.

## 3.4 Evaluation

We perform distributed word embeddings alignment on two datasets: the 300-dimension pre-trained Google News corpus word vector model (denoted as Google-wv) with over three billion words and phrases, and 300-dimension word vectors (denoted as Github-wv) trained on a recently collected Github corpus consisting of pull requests, issue comments and commit messages. We further evaluate the trained word vector models using semantic and syntactic tasks. We implement singular value decomposition and linear transformation in stochastic gradient descent approaches. At last, we evaluate the aligned word vector space and conclude some words who have significant semantic shifts in high dimensional space.

### 3.4.1 Github Dataset

The Github dataset used in this thesis is firstly collected by Achyudh Ram, which includes raw data of pull requests, issue comments and commit messages from 4124 repositories and the size is around 9.5G. Repositories are selected on the basis of them being "engineered" projects that are maintained and developed by groups of people in contrast to student projects, forked project, etc [65]. A full list of engineeered projects can be found on the website of RepoReapers [1]. Each record is stored in the format of JSON. Of the whole dataset, there are 53,430 pull requests, 1,495,156 issue comments, and 1,933,243 commit messages. We first extract useful text information from the raw data including the title, description, and following comments on an issue post, pull request and commit record. We believe communications under a single item could give enough background information and context. Then we filter out any URL or non-alphanumerics to get a clean vocabulary. The last step of pre-processing is tokenization and lemmatization on the raw data.

Tokenization and lemmatization are widely used when processing texts in natural language processing. Giving a character sequence or document, tokenization is the task of splitting it into pieces called tokens. Tokens are usually referred to words or phrases with semantic meanings. Lemmatization removes derivation of words and returns the base format in the dictionary. We denote the Github dataset as Github-comments in the remainder of the thesis.

[1]https://reporeapers.github.io/results/1.html

### 3.4.2 Pre-trained Word Embeddings

We perform distributed word representation alignment between two pre-trained word vector spaces. The target hyperspace is the Google-wv trained on Google News corpus. The source hyperspace is word embeddings trained on subculture corpus. As Section 2.2.2 describes the pre-trained approach, we obtain word embeddings using several training methods including word2vec [62] and fastText [44]. Let Github-wv denote word embeddings trained on Github-comments, and Github-wv-aligned denote the results after using the alignment methods introduced in this chapter. Considering people leave their online comments informally, we also utilize another two datasets as comparison models. In addition to the Github corpus, we collect about 44G real-time tweets in English from Twitter within a three-days time window. Moreover, we use WikiText dataset that contains 600 articles from Wikipedia [2] as a comparison. Both Github and Twitter datasets uses mainly day-to-day (informal) language, except there are more terminologies in the Github community. On the contrary, WikiText dataset is well formatted and contains mainly formal language.

To select the best model, we try a series of configuration combinations and perform semantic and syntactic tasks for each model. These configurations include using skip-gram or CBOW training algorithm, the dimensions of feature vectors, the minimum token frequency threshold, and neural network model (word2vec or fastText).

There are several methods to evaluate the word embeddings [42, 6]. Extrinsic evaluation methods are based on the ability of word embeddings to be used as features of machine learning tasks, such as sentiment recognition, Part-of-Speech tagging, and topic modeling. On the other hand, intrinsic evaluation methods are experiments of word embeddings on human annotations of word relations, for example, word similarity, word analogy, and word pair association. In this thesis, we take Wordsim353 [21] and Questions-words [3] [62, 60] as evaluation datasets. The dataset Wordsim353 provides a collection of human-assigned word pair similarity scores on a scale of 0.0 to 10.0 (e.g., train car 6.31), and it has recently emerged as a commonly used dataset for evaluating relatedness measures. There are two sets of English word pairs in Wordsim353. The first set contains 153 word pairs in 13 subjects, while the second set contains 200 word pairs in 16 subjects. The dataset Questions-words implies words relationships. There are five types of semantic and nine types of syntactic questions in the test set. For example, the semantic relationship type between (Athens, Greece) and (Oslo, Norway) is called *common capital city*, and the syntactic relationship type between (possibly, impossibly) and (ethical, unethical) is called

[2] https://en.wikipedia.org/wiki/Main_Page

[3] code.google.com/p/word2vec/source/browse/trunk/questions-words.txt

*opposite*. Overall, there are 8,869 semantic and 10,675 syntactic questions. We use built-in modules in Gensim [4] to evaluate word vectors.

### 3.4.3 Anchor Words Selection Strategy

The selection of anchor words could have a significant influence on the quality of aligned word vector models. If the training word dataset is too small, the alignment model learns little about the entire vocabulary space and performs poorly when testing on larger datasets. However, if the training word dataset is too large and introduces a considerable portion of noisy word pairs, the model would try to map between irrelevant vectors. Thus over-fitting occurs. In general, we want to increase the training data size while keeping the semantic alignment of two domains.

Stopword is a commonly used concept in search engines that a particular set of words, like *the* and *that*, are ignored by the program when searching and indexing. Researchers believe that stop words would not change their semantic meanings significantly across language domain, therefore utilizing them as anchor words for alignment models could probably increase the accuracy [85].

However, there is one problem that the stop word list has limited tokens (around 50 words), but the transformation matrix in the space of 300 dimensions generally requires more than 100,000 parameters. Therefore, we consider enlarging the anchor word dataset by sampling from the Warriner-EPA vocabulary at random. We further explore the relationship between anchor words and mean distance of alignment models.

### 3.4.4 Metrics

To assess the performance of the alignment model from Github subculture to general culture, we measure the average cosine distance between each word in two corpora before and after the alignment. If the average distance decreases significantly after alignment, it demonstrates our alignment model could transform the word vectors to the same high dimensional space effectively. Word vectors in Github-wv-aligned do not necessarily to be exactly the same as they are in Google-wv because they represent two different corpora. However, they should be close in general.

[4]https://radimrehurek.com/gensim/index.html

To illustrate the semantic shift across Github and Google News corpus, we further compare words based on the Part-of-Speech and construct a few word lists of several topics, including sentiments and actions.

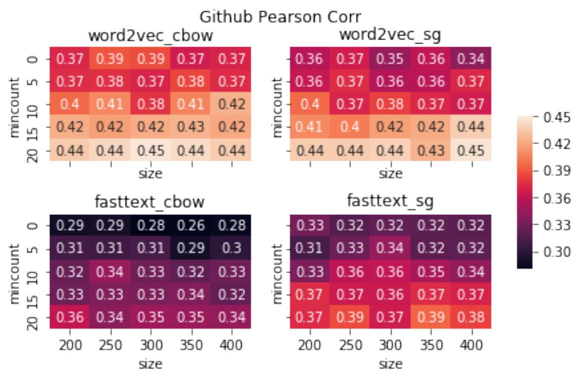
## 3.5 Results and Discussion

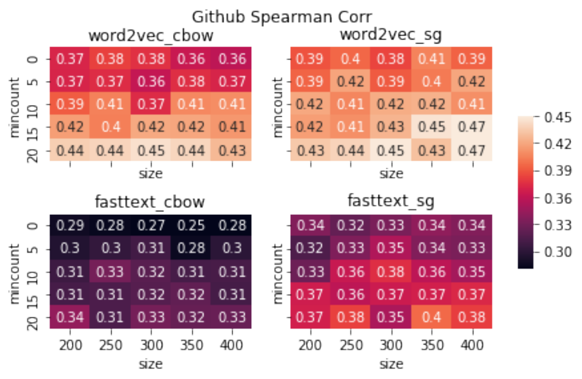
This section presents the results of parameter tuning, evaluation of training word vector models, and aligning high dimensional spaces. It could be divided into three parts: parameter tuning and word embeddings evaluation (*Part One*), comparisons of word vector space alignment strategies (*Part Two*) and findings of cross-culture semantic shifts (*Part Three*).

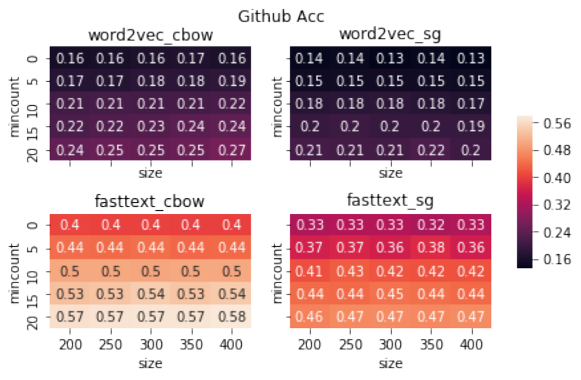
### 3.5.1 Part One

To get the best-trained word embedding model, we try different configuration combinations including the algorithm (word2vec or fasttext), architecture (skip-gram or CBOW), word vector dimension size as well as the minimum token frequency threshold, and do extensive evaluations for each model. Figure 3.2 shows the results of the parameter tuning on the Github corpus where the lighter color indicates the better performance. We mainly focus on two evaluation methods of word vectors: one is the word similarity regression test (Wordsim353), and the other is the word association test (Questions-words). The word similarity regression test is to predict the similarity score (from 0 to 10) of given word pairs, like bread and butter, computer and keyboard. We then calculate the Pearson Correlation and Spearman Correlation comparing predictions and human annotated Wordsim353 scores (see Fig 3.2). The correlation scores range between 0.3 and 0.5 and p-values are around $10^{-12}$. The word association test Questions-words is about the linear relationship of word vectors (*woman* - *man* $\sim$ *king* - *queen*). The test dataset consists of 14 sections such as capital/country, currency, family, past tense of verbs, plurals of nouns. The trained word vector models can answer the questions with an accuracy of 0.58 in the best case, but only 0.13 in the worst case. Note, we ignore those questions that include any word which is not in the vocabulary of word embeddings Github-wv. For example, the word "furnace" in Wordsim353(furnace stove 8.79) does not appear in Github-wv, so we do not count it in our evaluation stage. Same with (Athens Greece Tehran Iran) in Questions-words dataset.

We present the parameter tuning results in two evaluation tasks in Figure 3.2. Figure 3.2a and Figure 3.2b are about correlations of the word similarity regression test.


(a)


(b)


(c)

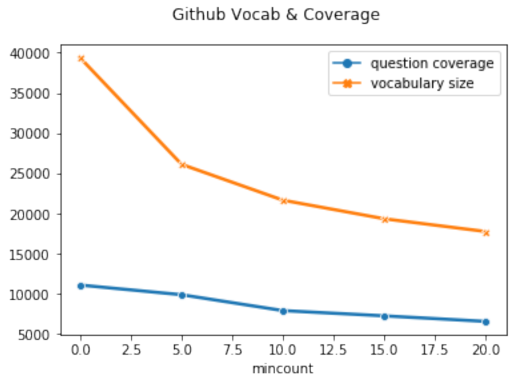
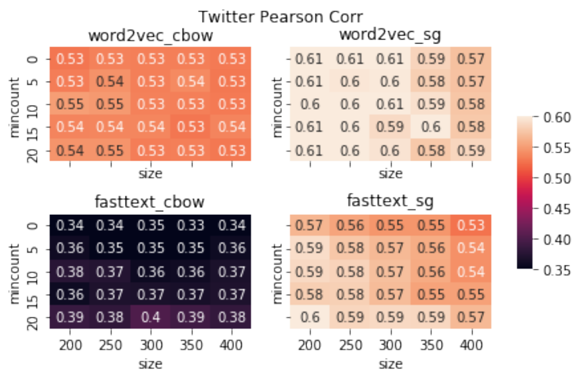
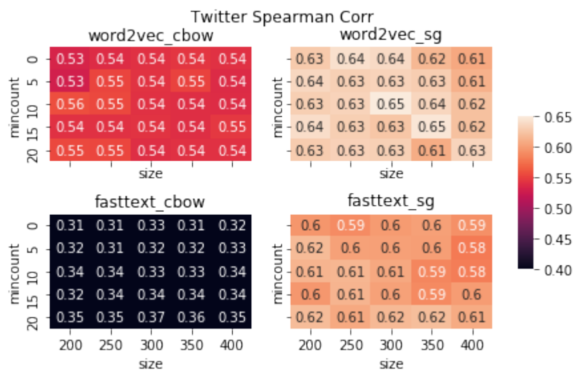

(d)

Figure 3.2: The Github word embeddings evaluation results of word similarity test (Wordsim353) in Fig 3.2a and Fig 3.2b, and word association test (Questions-words) in Fig 3.2c. There are four models: word2vec_cbow, word2vec_sg, fasttext_cbow, and fasttext_sg used in the experiments. Vector space dimensions (size) ranges between 200 and 400, and token frequency threshold (mincount) ranges between 0 and 20. Figure 3.2d demonstrates the trade-offs between vocabulary size and quality.

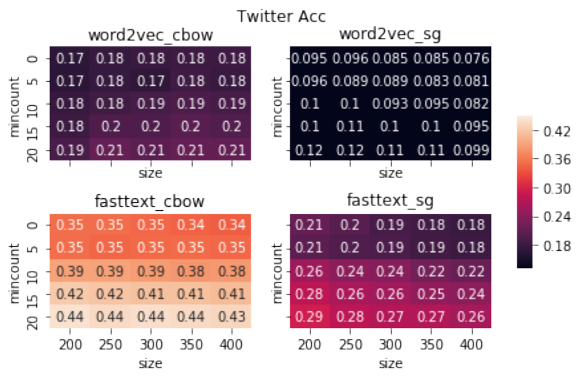
Figure 3.2c and Figure 3.2d are about the word association test. Specifically, Figure 3.2c shows the accuracy of calculating the missing word based on their linear relationship, and Figure 3.2d plots the vocabulary size of Github-wv and intersected vocabulary size with Questions-words task. We compare the similarity regression correlations of using four algorithms, which are word2vec + CBOW (word2vec_cbow), word2vec + skip-gram (word2vec_sg), fasttext + CBOW (fasttext_cbow), and fasttext + skip-gram (fasttext_sg) in each figure. For each model, we study how vector dimensions ("size" as the horizontal axis) and mini-



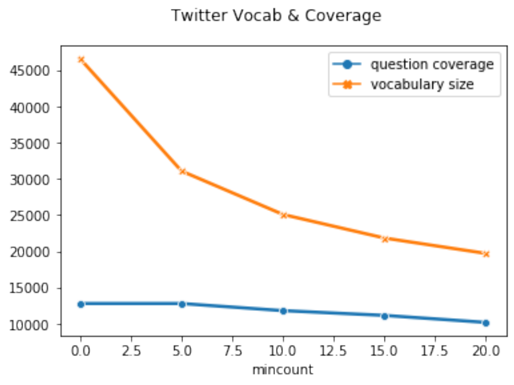
(a)



(b)



(c)

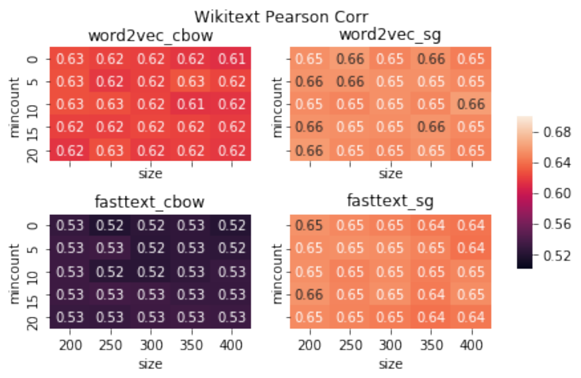


(d)

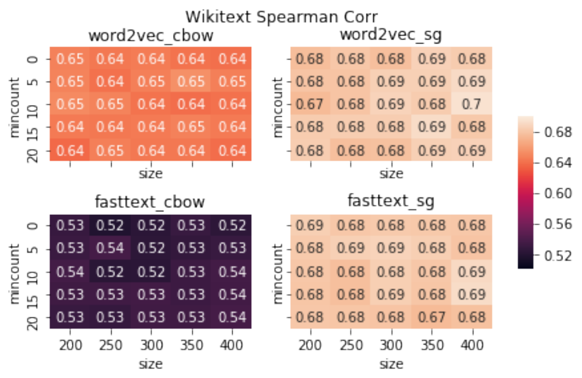
Figure 3.3: The Twitter word embeddings evaluation results. They are similar to the experiments described in Fig 3.2.

mum token frequency threshold ("mincount" as the vertical axis) affect the performance. We use heatmap to show the results, where the a lighter color cell represents a higher correlation or accuracy rate.

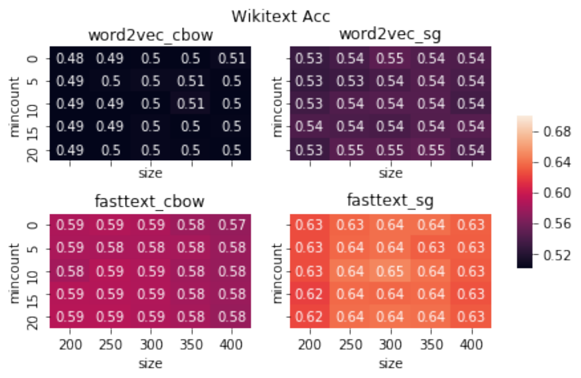
As Figure 3.2a and Figure 3.2b indicate, in general, word2vec models can get higher correlation scores on word similarity test compared to fasttext, and using the skip-gram architecture is slightly better than using CBOW. However, it is not the case in word association test as shown in Figure 3.2c, where fasttext performs significantly better than word2vec, and CBOW further improves the accuracy than skip-gram. All the three heatmaps indicate vector dimensions ("size") ranging from 200 to 400 have little impact on two evaluation tasks. On the contrary, increasing the minimum token frequency threshold ("mincount") can have better results. However, as Figure 3.2d demonstrates, the Github-wv vocabu-



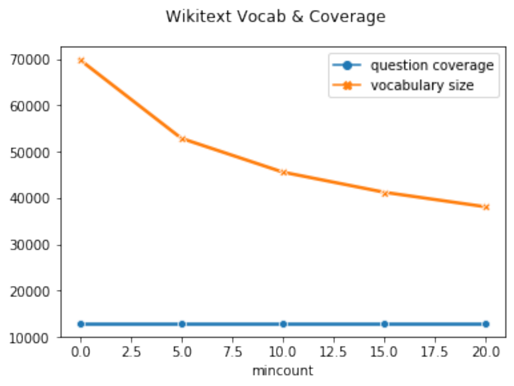
(a)



(b)



(c)



(d)

Figure 3.4: The WikiText word embeddings evaluation results. They are similar to the experiments described in Fig 3.2.

lary shrinks a lot when setting “mincount” greater than 10. It is a trade-off between the vocabulary size and the word embeddings quality.

We further evaluate models on Twitter and Wikitext corpora, and the results agree as above (see Fig 3.3 and Fig 3.4). The results further support our conclusion above, that word2vec + sg models are better at word similarity test, while fasttext + CBOW can achieve higher accuracy on word association test. Vector dimension does not affect the performance a lot, but increasing frequency threshold (“mincount”) up to 20 can significantly improve the performance. In fact, “mincount” filters out noisy examples that occur occasionally in the raw data but have negative impacts on the training process. We also compare the model on different corpora horizontally as shown in Table 3.1. Compared to models on other corpora, the Github word embedding model is relatively small and

| Result | Github | Twitter | Wikitext | Google News |
|---|---|---|---|---|
| **Pearson Corr** | 0.45(5.6E-15) | 0.61(2.5E-32) | 0.65(1.5E-42) | 0.63(1.8E-39) |
| **Spearman Corr** | 0.47(8.4E-15) | 0.63(5.6E-37) | 0.67(1.7E-47) | 0.66(2.5E-45) |
| **Question Coverage** | 6,548 | 10,182 | 12,902 | 13,191 |
| **Question Accuracy** | 0.58 | 0.44 | 0.64 | 0.77 |
| **Vocabulary Size** | 39,366 | 46,586 | 69,828 | 3,000,000 |

Table 3.1: Model evaluation and comparison on five corpora. Pearson and Spearman correlation with p-values are the results in word similarity task. Question coverage and accuracy are the results in word association task. Parameter settings: fasttext_sg model, vector dimension as 300, and mincount as 5.

less accurate, which is partly because people in Github community care less about topics like politics, fashion, and economy than on Twitter or Google News. Taking all factors into consideration, we would use the 300-dimensional skip-gram word2vec model with the frequency threshold set as 20 as Github word vector model in the remainder of the paper.

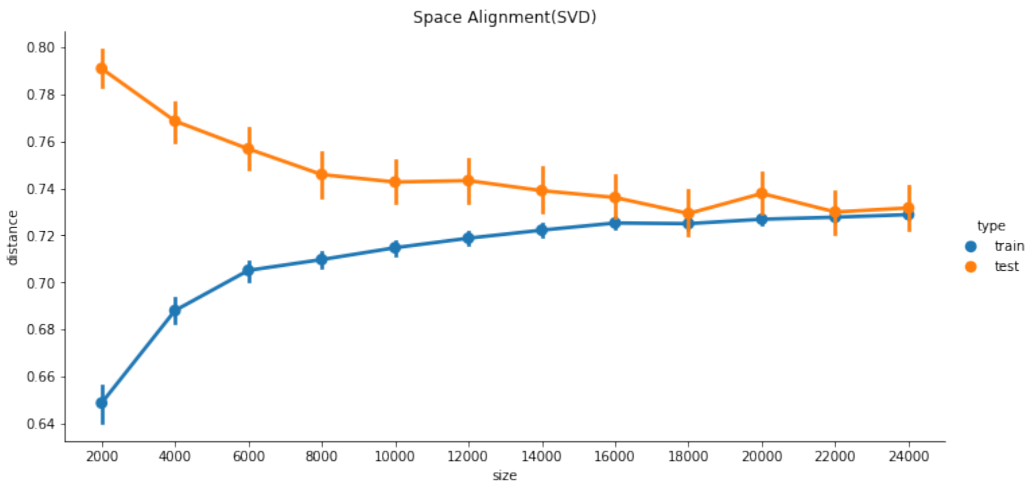
### 3.5.2 Part Two

As described above, we have two methods to align high dimensional space: singular value decomposition (SVD) and linear transformation. In this part, we compare the alignment performance of each method, and further investigate the impact of seed words size.

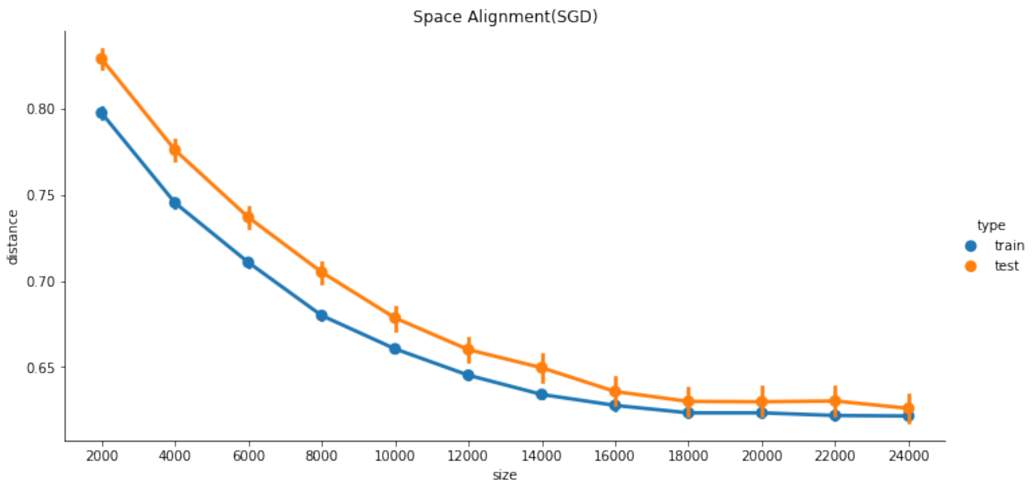
Figure 3.5a plots the results of aligning word vector space using the SVD method under different scales of the training dataset. As it indicates, the average cosine distance of word pairs in two models increases as the training dataset enlarges. However, the standard deviation decreases. The aligned models are tested with 1,000 tokens selected from the vocabulary space at random. As it shows in the figure, the initial average distance is around 0.8 with 2,000 training words and the distance decreases to 0.74 with 24,000 training words.

Figure 3.5b shows the results of using a linear transformation to align word vector space. We use 1,000 randomly selected words for testing as in the SVD method. As it is indicated, the average cosine distance for both training and testing decreases from over 0.80 to under 0.65 as the size of anchor words increases. We notice the performance would become stable with more than 18,000 anchor words.

In general, larger anchor words dataset can improve the alignment performance. However, it has a different impact on the training process. When using a small bunch of data,



(a) Average cosine distance using SVD method.



(b) Average cosine distance using SGD method.

Figure 3.5: The figures show the performance changes under different scales of training dataset using SVD and SGD method. The blue line indicates the average distance between word pairs in training process, while orange line is the average distance on 1,000 testing word pairs.

SVD method could align the training word vector space pretty well. However, the linear transformation does not align word pairs well. As training words increased, the SVD method shows its limitation in learning from big data, but linear transformation improves the results gradually. Overall the linear transformation method beats the SVD method with an average cosine distance below 0.62 when testing. The two figures also indicate training such alignment models requires a large dataset. However, stop words sampling method cannot satisfy the requirement even if it has better quality. We can continue trying this strategy if having a larger stop word vocabulary in the future. We think using 20,000 tokens including stop words as aligning anchor words should be a good strategy. In the following of this thesis, we apply a linear transformation method to align word vector models.

### 3.5.3 Part Three

In this part, we turn our attention to analyze the semantic shifts across the web environment. We apply our space alignment method to general culture (Google-wv) and the web subcultures (Github-wv). Firstly we divide all tokens into three-word sets according to Part-of-Speech (POS) tag: Noun, Verb, and Adj/Adv. For each word set, we calculate the average cosine distance and standard deviation of cross culture word pairs. As Table 3.2 shows, there are 8212 nouns, 3021 verbs, and 3872 adjectives or adverbs in total. In general, nouns have the highest mean distance at 0.643 and the lowest standard deviation as 0.118, which means there are significantly semantic shifts of nouns in space after alignment. However, the standard deviation of verbs is the highest at 0.126, which implies a part of verbs in vocabulary aligns well and has little semantic shift, but the other part shifts a lot. Adjectives and adverbs keep the lowest cosine distances across aligned culture space, so they have little semantic shifts in this case.

| **alignment** | **size** | **average distance** | **distance std** |
|---|---|---|---|
| **noun** | 8212 | 0.643 | 0.118 |
| **verb** | 3021 | 0.621 | 0.126 |
| **adj/adv** | 3872 | 0.588 | 0.125 |

Table 3.2: Using NLTK Part-of-Speech tag, we divided the intersected vocabulary of Google-wv and Github-wv-aligned into three catogories: noun, verb, and adjective/adverb. We calculate the average cosine distance and standard deviation (std) for each word in two spaces.

Table 3.3 shows examples of semantic changes between general culture and Github. Verb *commit* has significantly semantic shift in the Github culture. In general culture, the word is most close to *contribute*, *perpetrate* and *engage* (e.g. he *committed* an uncharacteristic error). However, in the context of Github culture, *commit* is often related to code

| **word** | **general** | **github** | **distance** |
|---|---|---|---|
| commit | contribute, perpetrate, engage | revision, patch, merged, pr | 0.960 |
| thread | purls, silk ribbons | worker, process, daemon | 0.613 |
| bug | worm, virus, insect, flaw | issue, problem, typo, glitch | 0.608 |
| crash | accident, collision, wrech | corruption, error, hang | 0.591 |
| merge | merger, combine, divest | develop, add, resubmit | 0.575 |
| password | passphrase, login, PIN code | credential, authentication, token | 0.358 |
| happy | glad, pleased, overjoyed, thrilled | willing, grateful, inclined, glad | 0.384 |
| sad | heartbreaking, bittersweet, tragic | uncomfortable, nervous, uneasy | 0.398 |
| angry | irate, annoyed, impatient | upset, annoyed, nervous | 0.434 |
| excited | thrilled, pleased, proud | stoked, thrilled, pleased | 0.382 |
| cautious | careful, mindful, hesitant | careful, skeptical, worried | 0.392 |
| difficult | impossible, tricky, tough | challenging, complicated, hard | 0.250 |
| important | vital, crucial, essential | crucial, valuable, useful | 0.278 |
| significant | substantial, markedly | considerably, substantial | 0.368 |
| efficient | economical, efficiency, streamlined | robust, flexible, sophisticated | 0.321 |

Table 3.3: Examples of words and their synonyms in Google-wv and Github-wv-aligned. Action words show significant semantic shift, while sentiment words do not have much shift. But sentiment words show subtle potency differences across the two cultures.

changes and the usage would be like "he *committed* a new revision". Another example is the word *bug*. In general culture, the word *bug* refers to *worm* and *insect*. However, it mainly refers to software issue or problem instead of small insect in zoology.

We further found the words that represent human sentiments are not shifting a lot across culture. Table 3.3 gives a few examples of word synonyms by calculating cosine distances in hyperspace. The words with underline are related to sentiments, while words with dotted line are actions. Word "commit", "crash", and "merge" have large hyperspace distances after alignment, and we can see significant semantic shifts as those words are used in software community. On the other hand, word "happy", "sad", and "angry" do not have much semantic shifts, and the hyperspace distances are much smaller than these action verbs. We further construct an sentiment word list from an online website [5]. The word list consists of 147 words that describe sentiments, for example, alarmed, shamed, trust, etc. The average distance of sentiment words in the word list is around 0.35, while it is about 0.6 for action words in Table 3.2. This is because most sentiments have universally shared meanings, but verbs are usually semantically overloaded. However, we still can find

[5]http://www.psychpage.com/learning/library/assess/feelings.html

subtle differences of sentiment words in two context. The word *happy* is similar to *glad*, *pleased* and *overjoyed*. However, in Github environment, it is close to *willing* and *inclined* which show preference to do something. For example, "I am happy to do this" is probably what people are saying in GitHub, whereas in general it is "I am happy" or "I feel happy today". In next Chapter, we will introduce the Evaluation-Potency-Activity space, which could quantify those subtle changes in affective dimensions.

In a word, we believe verbs usually get apparent semantic shifts across subcultures and have different common usages. Moreover, sentiment words vary on the axis of potency. People tend to be close to neutral in Github culture compared to the general environment. We will prove this hypothesis in Chapter 4.

## 3.6 Conclusion

In this Chapter, we train and select the word embedding model on a Github corpus. We perform extensive parameter tuning and evaluation, and we further compare horizontally on Twitter and Wikitext corpus. The wikitext model gets the highest correlation score and prediction accuracy. We believe it is because of wikitext data is constructed formally and contains more linguistic information compared to Github and Twitter, since most pull requests, commit messages and tweets are short in length with misspelling and abbreviation.

We applied two methods to align word vector space to make a further comparison of Github culture and general culture. We find linear transformation with at least 20,000 anchor words could get the best performance in learning enough space information in common while keeping the differences. Using the aligned Github model and GoogleNews model, we make a coarse-grained word-level comparison. We find verbs and nouns would have more semantic shifts across online web cultures while adjectives and adverbs keep relatively the same. Some words like *commit*, *merge*, *thread* have developed specific usage in subculture and therefore become like terminologies. We also find people show different intensity when using sentiment words. In general, they behave more neutral in the Github environment without showing extreme intensities.

# Chapter 4

# EPA Expansion

## 4.1 Introduction

Sentiment analysis is widely applied to a variety of important areas, such as health informatics [35], recommendation system [19], intelligent assistant [87] and customer surveys [67]. It is a popular area of interest in natural language processing. Most sentiment analysis relies heavily on a human annotated dictionary that maps words to sentiment, while some researchers are working on generating such lexicons automatically from a small bunch of seed words. Expanding sentiment lexicons from text by machine learning models reduces the cost of manual annotation, and it is more feasible to apply to a variety of applications. However, most explorations in sentiment lexicon expansion methods use one-dimensional score, for example, the polarity of attitude as positive, negative or neutral [57, 17]. One dimensional sentiment model could be a good fit in some applications, like customer review analysis. However, it shows limitations when dealing with more complicated human sentiments, like happy, sad, angry and afraid. In this thesis, we propose to expand sentiment lexicons using the Evaluation-Potency-Activity (EPA) model, which maps words or phrases to a three-dimensional space to best describe the complexity of sentiments.

Osgood first carried out a large set of cross-cultural studies and showed that the EPA model could be used to characterize the shared affective meaning successfully [68, 70, 69]. Based on this work, Heise further measured the EPA space in several countries [37, 38, 39]. Besides, participants in his studies were asked to rate identities (e.g., mother), behaviors (e.g., help), adjectives (e.g., kind) and institutions (e.g., school). Sharing similar idea, Warriner et al. construct the most recent manually annotated sentiment lexicon [93].

These culturally dependent sentiment dictionaries are collected by asking random people who belong to this desired country or region to rate each word or phrase in the surveys. However, it becomes challenging if the target culture is an online culture because multiple subcultures usually influence participants and it is hard to ensure the survey data quality. An alternative solution is to take a look at what people are talking about and how they are saying if in the online subculture. In this thesis, we propose to collect sentiment lexicons of online subcultures by analyzing word and sentence usage. More specifically, as it shows in Figure 4.1, we train mapping models $f$ that map general word vector spaces Google-wv to the three-dimensional sentiment space Warriner-EPA with a small dictionary, and then expand the subculture sentiment dictionary Github-EPA by utilizing aligned word vector models Github-wv-aligned introduced in Chapter 3. Evaluated on a affective ratings dataset measured in the Github community (Themis-EPA), we demonstrate that our proposed approach $f(g)$ is a good approximation of the hidden function $f'$. We denote the trained mapping model as mapping-model as a reference.

Alhothali proposed to expand sentiment lexicons using a graph-based semi-supervised method. Her method could achieve high correlation ($\tau = 0.51$) and low error rate (mean absolute error $< 1.1$) [1, 2]. Based on her work, we further explore using artificial neural networks to acquire expanded sentiment lexicons. We compare the performance of using graph based semi-supervised method, support vector regression as well as artificial neural networks. The shortcoming of using a graph-based label propagation method in our case is to induct sentiment lexicons for other domains, such as the Github culture. It is unavoidable to insert the word node into the graph and update the affinity matrix each time predicting the EPA value for a new word vector. This process could be extremely time consuming and require significant memory. On the contrary, supervised learning could make label prediction offline, which is a great convenience. We further reduce the mean absolute error rate from 1.1 to 0.6 with well-trained neural networks. Applying the model to aligned Github word vectors, we finally obtain the sentiment dictionary for Github culture. At last, we carry out word level EPA comparison and analysis for general culture and Github culture.

## 4.2 Background

Alhothali and Hoey first propose an extension to graph-based sentiment lexicon induction methods by incorporating distributed and semantic word representations [2, 1]. Their work shows the ability to generate a significant number of new sentiment assignments with high accuracy. The highest correlation is 0.51, and the lowest error is less than 1.1, which

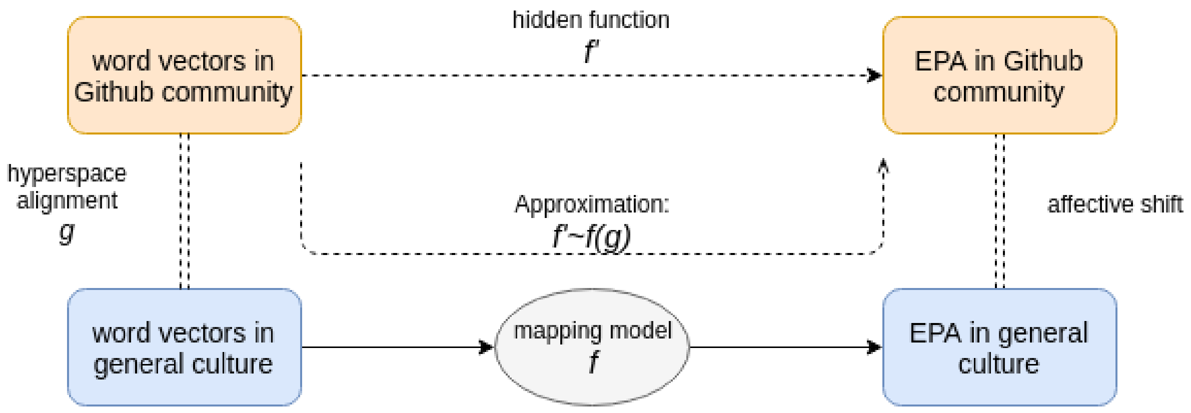


Figure 4.1: The workflow of EPA expansion on Github. In last chapter 3, we demonstrate learning the hyperspace alignment function $g$. In this chapter, we focus on training the mapping model $f$, which maps from word vector space to three-dimensional EPA space.

outperformed most label propagation models and approaches a supervised model (SVR).

In their work, they implement and evaluate label propagation using four different word representations to build the similarity graph. The first is built based on the semantic relationship between words. Two semantic lexicons used are WordNet dictionary (WN) and paraphrase database (PPDB). The semantic-based similarity of any pair of words in the vocabulary is calculated based on synonym relationships. The second method is based on the count-based co-occurrence statistics that is aggregated from different corpora. N-gram features from one million news articles dataset are also used to generate the word pair co-occurrence matrix. Then they construct the statistical word vectors by computing the smoothed positive point-wise mutual information (PPMI) of the co-occurrence matrix and factorizing with truncated Singular Value Decomposition (SVD). The affinity matrix is computed from the cosine distance of statistical word vectors. The third method uses pre-trained word embeddings, and affinity matrix is computed using the cosine similarity between vectors in space. The fourth method combines both semantic and distributional information from the semantic lexicon (or dictionary) and word embeddings. The affinity matrix is the averaged scores in method 1 and 3.

Comparing their induced EPA scores using the label propagation algorithm against experimental values, it shows that the highest error rate (MAE) ranged between 0.8 and 1.3. The result of combining both semantic and neural word embedding is better than the corpus-based or semantic lexicon-based algorithms. Moreover, it is comparable with those generated using a supervised learning algorithm. With the induced lexicon, they further predict reader's reaction towards news articles.

## 4.3 Methods

The field of machine learning is traditionally divided into three sub-fields: supervised learning that observes labeled datasets denoted as (feature, label) pairs and predicts incoming data in the future [11]; unsupervised learning that learns from a set of unlabeled data and organizes the items [33]; reinforcement learning that repeatedly takes actions and gets rewards from environment in order to maximize further benefits [86].

In this chapter, we introduce the models used in this thesis: graph-based semi-supervised learning [12, 96], support vector regression [83, 7] and artificial neural network [80, 64] (Figure 4.2). Previous attempts to expand sentiment lexicons using the graph-based label propagation algorithms achieve greater accuracy than other methods. In this thesis, we further demonstrate that using neural networks could be a better choice.

### 4.3.1 Motivation

Previous studies and research have shown affect divergence does exist across cultures. We believe that comparing with using general lexicon, using an affective lexicon for the specific subculture can be beneficial to sentiment analysis research, and can explain some previous findings. However, recruiting participants to annotate concepts would be difficult, and it is not easy to ensure the collected data quality. This is because people would usually rush through the survey, or could not fully understand the affective ratings questions. In this thesis, we propose to use machine learning and word embeddings and give approximations for subculture affective meanings.

As it shows in Fig. 4.1, we denote the mapping function from word vectors to affective ratings in a general cultural setting as $f$, and the underlying function for a given Github subculture as $f'$. Function $g$ represents the hyperspace alignment model. In this Chapter, we propose that:

$$f'(x) \approx f(g(x)), \tag{4.1}$$

and our method does capture subtle affect changes between the general culture and the Github community subculture.

### 4.3.2 Graph-based Label Propagation

One of the significant problems in machine learning is that most data we have is unlabeled, so we lack enough labeled dataset to train good classifiers. However, people recently

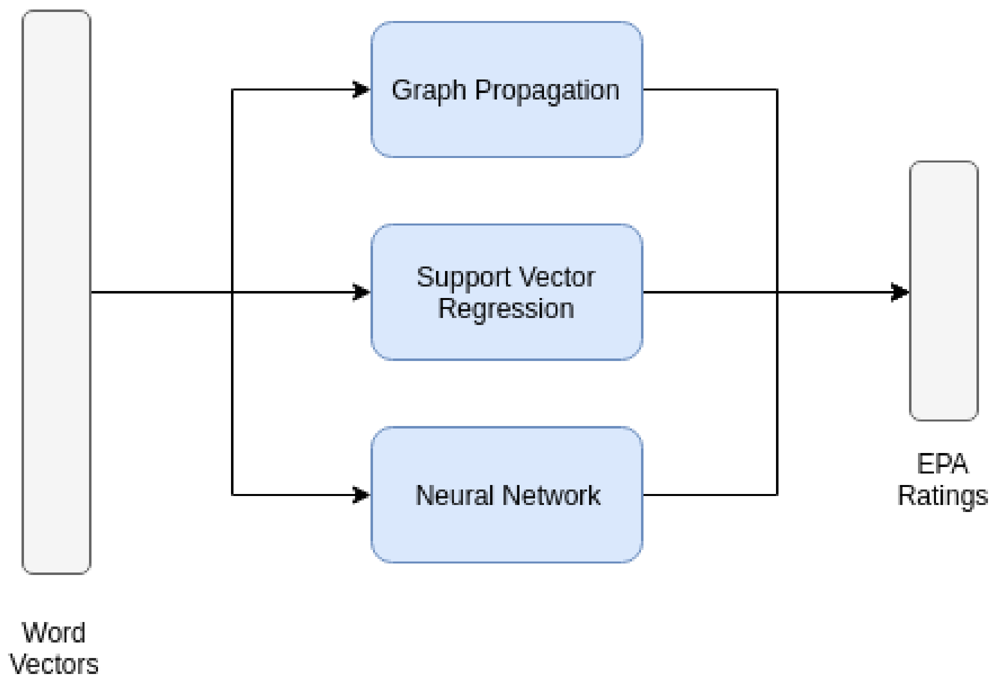


Figure 4.2: In this thesis, we propose using three methods to train the mapping models, which are graph-based semi-supervised learning, support vector regression, and neural network.

discovered that unlabeled data could be helpful as shown in Fig 4.3 [79]. The unlabeled data provide information about data distribution so we can model more accurate decision boundary. The idea of using labeled and unlabeled data to archive better performance with less annotation effort is called semi-supervised learning [12, 96].

Graph-based label propagation is a commonly used semi-supervised learning algorithm, which represents the whole dataset as a graph. It makes much sense because some datasets, like social network, and citations, are in the format of a graph by nature. Nodes in the graph are either labeled or unlabeled data which are connected to other nodes by some defined distance functions. Labels propagate between nodes in each iteration under the smoothness assumption, which proposes two instances should have similar output labels if they are similar according to the graph.

There are two stages in a graph-based label propagation algorithm, which are graph construction and label inference. In general, there are two methods in graph construction [58]. K-nearest neighbor (k-NNG) graph adds edges between an instance and its k-nearest neighbors, while e-neighborhood method adds edges to all instances inside a ball

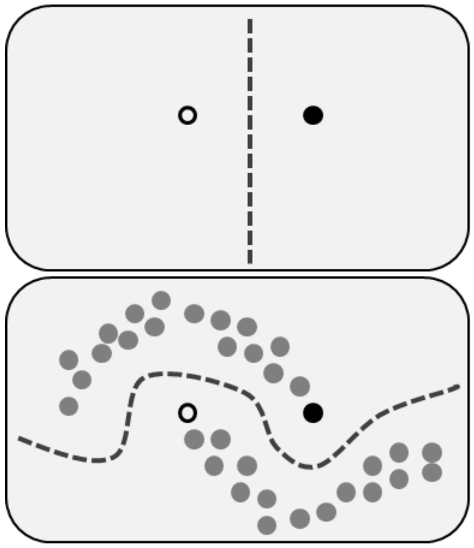


Figure 4.3: A illustration of why unlabeled data could be helpful [15]. The knowledge of having more points in the banana shape allows to decide more accurate boundaries for classification. Otherwise, the decision boundary would be a simple straight line separating the two labeled nodes.

of radius e. Both methods have pros and cons. K-NNG results in an asymmetric and irregular graph where edges and weights are unidirectional. E-neighborhood is sensitive to the value of the radius. For example, unreasonably small radius leads to disconnected components in the graph. Some other graph construction approaches include linear neighborhood [13], b-matching [91] and fitting graph to vector data [16]. In this thesis, we perform exhaustive parameter tuning experiments on the radius range in Section 4.5.1.

We first give some notations before the stage of label inference [97]. The algorithm is based on a given labeled data $L = (X_l, Y_l)$ and unlabeled $U = (X_u, Y_u)$, where $X$ is input features (word embeddings) and $Y$ is output labels (affective ratings). The graph weight matrix is denoted as $W$ by performing transformation kernels on edge distances. In this thesis, we apply Gaussian kernel $w_{ij} = e^{-\|x_i - x_j\|^2/2\sigma^2}$. Degree matrix is denoted as $D$ where $D_{ii} = \sum_j W_{ij}$. We compute the random walk normalized Laplacian matrix $\Delta = D^{-1}W$ so propagation becomes $Y \leftarrow \Delta Y$. In order to power the graph towards convergence, we clamp labeled the data to initial value with factor $\alpha$ after each round as follows:

$$Y(t+1) = \Delta\alpha Y(t) + (1-\alpha)Y^0,$$

where $Y^0$ is the initial labels, and $Y(t)$ is the predicted labels from previous iteration.

Combining the initial values and calculated gradient, the algorithm is making adjustments to predicted labels $Y(t+1)$ at current iteration. The propagation continues til $Y$ converges. Or we can set pre-defined criteria to speed up the algorithm, for example, the algorithm can calculate the average change $Y(t+1) - Y(t)$ and stop if the value becomes below a threshold, or return after propagating for $T$ iterations. Zhu et al. proved the clamping method would finally converge close to a simple solution [97, 96].

Graph propagation has been widely applied to natural language processing areas. Wilson et al. and Blair et al. obtain good results in predicting polarity lexicons (positive or negative) [94, 9, 89]. WordNet is commonly used in these tasks as it defines synonyms, antonyms, etc., which gives much information to graph construction. In this theses, however, we mainly use word2vec models and the geometric features. We talk more about this in the next section.

### 4.3.3 Support Vector Regression

Support vector machine (SVM) is a supervised learning algorithm used for solving classification and regression problems [36]. The main idea of SVM is to construct one or sets of hyperplanes in a high dimensional space. Non-linear kernels are applied in SVM to make data distributed in high space linear separable. A good separation in SVM is a set of hyperplanes that has the most significant distance of the nearest instance of each category, which refers to as "maximum-margin hyperplane". Suppose we are given a training dataset of $n$ entries of the form $(x_0, y_0)$, $(x_1, y_1), ...,$ $(x_i, y_i), ...,$ $(x_n, y_n)$, where $x_i$ is a m-dimensional feature vector, and $y_i$ is the category of item $x_i$. The objective function of SVM for classification is to optimize the normal vectors $w$ to the hyperplane, that is:

$$\text{minimize } \frac{1}{2}\|w\|^2 \tag{4.2}$$

$$\text{subject to } \begin{cases} y_i - wx_i - b \leq \epsilon \\ wx_i + b - y_i \leq \epsilon \end{cases} \tag{4.3}$$

where the $\epsilon$ is a margin of tolerance.

The supported vector regression (SVR) shares the same idea of kernel and margin as SVM in classification problems only with a few minor differences [83, 7]. Since the output is a continuous real number in SVR instead of categorical label, there is a margin of tolerance $\epsilon$ in the objective function as it is shown in Figure 4.4. That is to say, SVR solves this function:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{4.4}$$

$$\text{subject to } \begin{cases} y_i - wx_i - b \leq \epsilon + \xi_i \\ wx_i + b - y_i \leq \epsilon + \xi_i^* \\ \quad \xi_i, \xi_i^* \geq 0 \end{cases} \tag{4.5}$$

Some common kernels in SVM include the polynomial kernel, rbf and sigmoid. In our experiment, we mainly use *rbf* kernels and set the $\epsilon$ as 0.05 and $C = 10$. We utilize the implementation of SVR in sklearn [1] to train models. In this thesis, the input feature vector $x$ is word embeddings trained on corpora, and the output is 3-dimensional affective ratings. For each affective dimension, we obtain a separate SVR regressor.

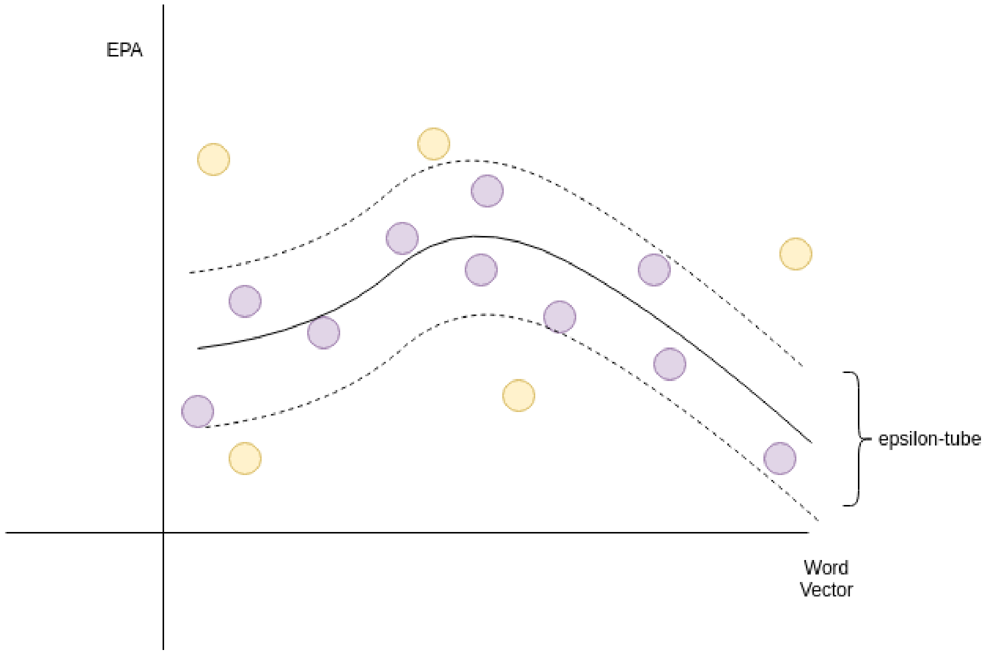


Figure 4.4: The Structure of SVR. $y_i = w \cdot x_i + b$ is the prediction for the sample. The purple points are predictions within the $\epsilon$ error ranges, while yellow points are beyond the margin.

[1]http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

### 4.3.4 Artificial Neural Networks

Artificial neural network (ANN) is inspired by biological neural networks that constitute animal brains. The basic component in artificial neural networks is called the neuron, which receives and processes signals and eventually triggers other neurons connecting to it. Just as biological neurons, the connection between neurons varies in the signal strength, which is defined as edge weights in artificial neural networks. Each neuron could decide how it processes input signals, which is called activation function. Some commonly used activation function include *sigmoid*, *tanh* and rectified linear functions.

There are two stages of training a neural network: forward-propagation and back-propagation. Forward-propagation refers to the process of calculating neuron values from network input to the output layer by layer, while back-propagation is to adjust weight matrices based on error functions.

There are many variants of the architecture of artificial neural networks. Multilayer perceptron (MLP) is a fully connected neural network with at least three hidden layers of nonlinearly-activating nodes [23]. It is a strong classification model dealing with some complex problems, yet, the calculation complexity limits its feasibility. Convolutional neural network (CNN) is a simplified neural network and widely applied in computer vision [50]. Neurons in convolutional neural networks share weight matrices in some degree. Furthermore, there are convolutional layers and pooling layers, which are easy to obtain compared to fully connected layers. Other popular networks include recurrent neural network, generative adversarial network, etc. It is easy to design a new neural network architecture and train models according to needs. The study of artificial neural networks is now referred to as "deep learning", and it is a trendy area both in industry and academia [53, 27].

In this thesis, we use word embeddings as the input vector, and EPA ratings as the output vector (see Figure 4.6). We introduce the architecture of our neural network in Section 4.4.3.

## 4.4 Evaluation

### 4.4.1 Datasets

Distributed label propagations, including graph-based semi-supervised method, support vector regression supervised learning and neural network based, are implemented with pre-trained word embeddings (see Chapter 3). The method described above relies on Warriner et al. EPA lexicon, which contains 13,915 English words [93].

#### 4.4.1.1 Pre-Trained Word Embeddings

In this thesis, we assume word embeddings trained on Google News corpus (Google-wv) could represent the general meanings and understandings of words and phrases. The word embedding model trained on Google News data using *word2vec* contains 3 million 300-dimensional word vectors [2]. The *word2vec* approach learns phrases in a simple data-driven approach, where unigram and bigram count are rating the scores. The obtained subculture word embeddings are trained and aligned to Google News word embeddings using the methods described in Chapter 3. In this thesis, we take Github as an example. The Github word embedding model we use (Github-wv) consists 26,104 words and is represented in a 300-dimensional vector space. It is trained on a Github corpus, which includes the pull requests, commit comments and discussions from 4,124 repositories (see Section 3.4.1).

#### 4.4.1.2 Warriner EPA dataset

The Warriner EPA dataset [93] consists affective norms of valence (evaluation), arousal (activity), and dominance (potency) (VAD) [10] collected from Amazon Mechanical Turk [3] for 13,915 words. The ratings range from 1 to 9. Since EPA ratings are within the range of -4.3 to 4.3, we scale it using min-max scaling method. Basically the min-max normalization is a linear transformation given the formula:

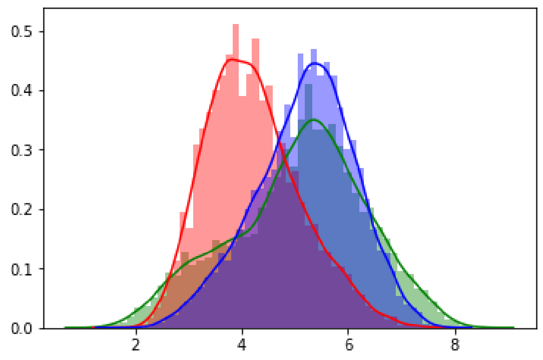
$$y_i = \frac{x_i - min_A}{max_A - min_A} \times \ (max_B - min_B) + min_B,$$

where $x_i$ is the given value from source space $A$, $y_i$ is the mapping value in target space $B$, $min$ and $max$ stand for the minimum and maximum value in each space respectively. Let denote the Warriner EPA dataset as Warriner-EPA as a reference.

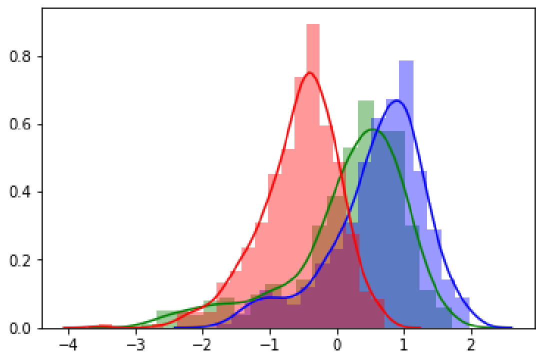
As Figure 4.5a illustrates, the VAD scores do not distribute uniformly. Instead, they look close to normal distributions. Since we try graph-based label propagation method, which is a mostly linear transformation between neighbors, we would like to try to add non-linear relationships to this algorithm and see if there is any improvement. After min-max normalization to EPA space, we further normalize the data to Gaussian(0, 1) by subtracting the average and dividing by the standard deviation (shown as Table 4.1): $y = \frac{x-\mu}{\sigma}$. Then transform using cumulative distribution function: $z = \frac{1}{2}Erfc(-\frac{y}{-\sqrt{2}})$. In such way, the ratings distribute uniformly between 0 and 1.

[2]https://code.google.com/archive/p/word2vec/

[3]https://www.mturk.com/



(a) VAD Distribution of Warriner Dataset. The ratings of affective dimension distribute from 0 to 9 normally.



(b) EPA Distribution of Themis Dataset. The ratings of affective dimension distribute from -4.3 to 4.3 normally.

Figure 4.5: Green: evaluation/valence. Red: activity/arousal. Blue: potency/dominance. The average values for evaluation/valence and potency/dominance are positive, however, it is negative for activity/arousal.

#### 4.4.1.3 Themis EPA dataset

In THEMIS.COG project, they ask 500 Github users to rate 587 concepts regarding their evaluation, potency and activity polarity and strength. Each concept has 50 participants ratings. Of the 50 people, there are 35 white and 15 non-white people. The observations of the survey also include participant's location, gender, nationality and more, but we do not use that information in this thesis. Before using the collected data, we do some processing for quality checks. We filter out those participants' ratings who did not take the survey seriously. The average skip time and average time spent on each concept are

| Ratings | Valence | Dominance | Arousal |
|---|---|---|---|
| Min | 1.26 | 1.68 | 1.6 |
| Max | 8.53 | 7.9 | 7.79 |
| Ratings | Evaluation | Potency | Activity |
| Mean | 0.20 | 0.55 | -0.67 |
| Std | 1.51 | 1.30 | 1.25 |

Table 4.1: The VAD Distribution in Warriner's dataset. valence=evaluation, dominance=potency, arousal=activity.

| Ratings | Evaluation | Potency | Activity |
|---|---|---|---|
| Mean | 0.16 | 0.60 | -0.58 |
| Std | 0.87 | 0.71 | 0.59 |

Table 4.2: The EPA Distribution in Themis EPA dataset. The variance values of three affective dimensions are half of the Warriner's dataset.

used to measure respondents tendency in general. Direction checks (left, neutral, right) are used as an indication of the participants give ratings in the same way, for example, all very positive scores. We filter out participants who fail at least two checks. After processing the raw data, there are more than 30 participants' ratings left for each concept. We calculate the average rating and standard deviation for each concept on three affective dimension as an evaluation dataset (denoted as Themis-EPA).

In Figure 4.5b and Table 4.2, we show the distribution of evaluation-potency-activity affective dimensions in collected Themis-EPA dataset. Compared with Warriner-EPA, the affective ratings have close average values, however, they have small variances.

## 4.4.2 Methods for Sampling Seed Words

The training dataset has a significant impact on the performance of EPA expansion models. In this thesis, we propose three methods of selecting words from lexicons and evaluate the performance in the next section.

The first strategy adopts the Osgood et al. findings [68] (Table 4.3). They select 43 words (E+, E-, P+, P-, A+, A-) from a dictionary that are extremely expressive along the dimension of evaluation, potency, and activity, either positively or negatively. We use these 43 words as seed words along with another randomly picked word set. The second strategy shares the idea of using descriptive words in the training process, however, instead of fixed seed words, we set a threshold for each dimension and select at random from word pools where the absolute EPA value for words is above the threshold. Note, to ensure the trained models could learn equally in each EPA dimension, the word candidate pools are independent of each other, which is to say, we avoid the cases where selected words are all with high scores in a single dimension. For example, if selecting 300 tokens from the lexicon and setting the threshold as 1.0, there are 400, 700, 800 concepts whose absolute evaluation, potency, and activity scores are above the threshold. We select 100 out of 400, 700, and 800 respectively and combine the three subsets together as final selection. Otherwise, a large portion of the selected 300 concepts would be the ones with strong

| Words | Seed Words |
|---|---|
| E+ | good, nice, excellent, positive, warm, correct, superior |
| E- | bad, awful, nasty, negative, cold, wrong, inferior, |
| P+ | powerful, strong, potent, dominant, big, forceful, hard |
| P- | powerless, weak, impotent, small, incapable, hopeless, soft |
| A+ | active, fast, noisy, lively, energetic, dynamic, quick, vital |
| A- | quiet, clam, inactive, slow, stagnant, inoperative, passive |

Table 4.3: Osgood proposed 43 words that are extremely expressive on three affective dimensions [68]. E+/E- represent those words are very good/bad; P+/P- represent those words are very powerful/powerless; A+/A- represent those words are very active/inactive.

activity scores, which leads to bias in the training datasets. The third strategy makes a random selection from all candidate words no matter how the EPA values look.

To explore the impact of seed words size, we carry out extensive experiments for the three strategies with different settings of seed words numbers.

### 4.4.3 Neural Network Structure

In this project we use *keras* [4] to build and compile neural network models. At first, we attempt to implement a convolutional neural network to predict EPA labels. However, the results turn out to be rather disappointing even after careful tuning. MAE is around 1.0 and stays stable after the first epoch, which is far below our expectation because we believe neural networks could get better results than semi-supervised training. So we discard using convolution layers in the network and instead, we apply dense layers. The new network structure gets better performance and beats all other algorithms. The architecture of our neural network models is as shown in Fig. 4.6.

### 4.4.4 Evaluation Metrics

The Warriner's dataset is divided into training and testing subsets at random to evaluate the performance of EPA expansion models. The training dataset (denoted as EPA-training) includes one-third of the whole data , which is 5000 words, while the testing part (denoted as EPA-testing) includes 8000 words. We exclude those words that do not exist in the

[4]https://keras.io/

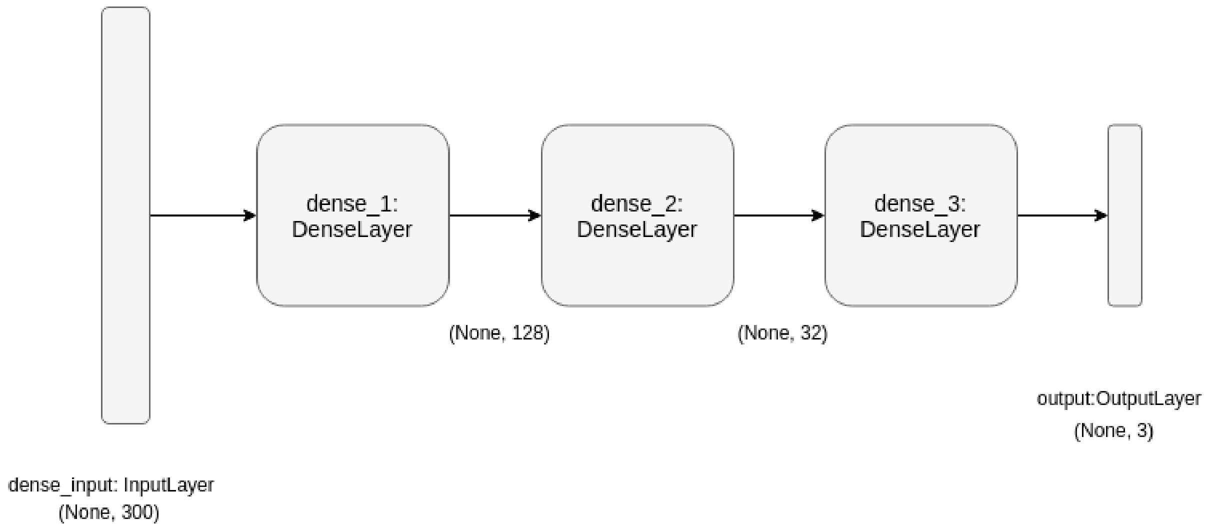


Figure 4.6: The Structure of Neural Network in EPA Expansion. The input vector has 300 dimensions. The first layer has 128 units followed with an "tanh" activation layer. The second layer has same structure with the first layer except there are 32 units. The last layer is a three-unit dense layer without activation function.

pre-trained word embeddings (Google-wv). Let us denote the predicted ratings using our proposed approach as General-EPA. We use the mean absolute error (MAE) together with the root mean square error (RMSE) as evaluation metrics to compare EPA-testing and General-EPA.

### 4.4.5 Methods for Github EPA Expansion

As Figure 4.1 shows, since we have trained a mapping model $f$ predicting the affective meanings from general word embedding space Google-wv to EPA space Warriner-EPA, we would like to apply the mapping model to Github corpus Github-wv-aligned and obtain the affective lexicon for the Github subculture. Let denote the predicted affective meanings for words in Github subcultures as Github-EPA. We evaluate the $f(g)$ prediction performance by comparing Github-EPA with Themis-EPA first. Then we try to describe the features of Github subculture on the level of word usage.

## 4.5 Results and Discussion

In this chapter, we first compare different machine learning strategies to get good models mapping from word vector space to EPA space. We carry out extensive parameter tunings in Section 4.5.1. Then we apply the best model to predict the EPA value of the given subculture words/phrases. Here we only explore on the Github word space. We evaluate the EPA label induction on Github culture by calculating MAE/RMSE on a small human-annotated lexicon. At last, we conclude some features describing the culture of Github.

### 4.5.1 Training Mapping Models to Predict EPA

To select the best model mapping from word vector space to EPA space, we train and compare three machine learning algorithms under different parameter settings. Here we list and explain the parameters used in our project in Table 4.4.

We would first discuss the performance of setting specific parameters for each algorithm and then give horizontal comparisons of three models regarding EPA threshold and seeds number (see Figure 4.10).

#### 4.5.1.1 Graph

Since we build a large network connecting labeled and unlabeled data in graph-based semi-supervised learning, it matters a lot what strategy is used to construct the network. At first, we believe the radius of neighbors and the $\epsilon$ in the Gaussian function are important because the two parameters would shape the network structure and degree matrix. However, as Fig 4.7 indicates, the value of $\epsilon$ used in Gaussian function does not play an important role. Even though smaller $\epsilon$ makes the cosine distance differences more significant in degree matrix, the model improves very little. Neighborhood radius affects model performance as much as we expect, and the best setting would be around 0.6. Smaller radius reduces the number of neighbors for each node. There are two downsides of a sparse network: first, label propagation slows down due to the sparsity; second, rare words/phrases become difficult to obtain a label for, even become disconnected from the network. While higher radius leads to almost a fully connected graph, which increases overfitting but reduces the convergence probability.

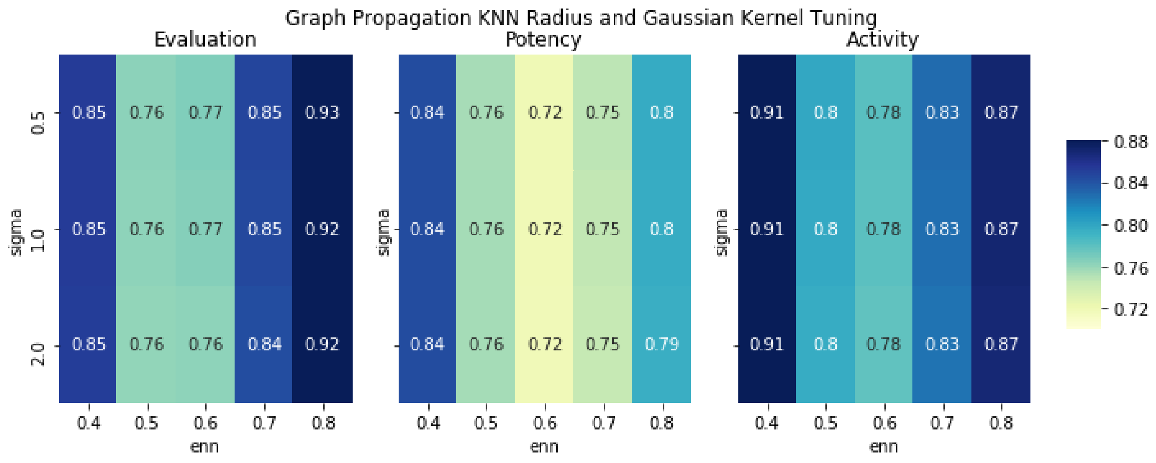
Propagation rounds also play an essential role in the final results. Some nodes that are too far away from labeled nodes can only obtain values after specific iterations. In Fig 4.7 we show the changing curve under various iteration settings. As the figure indicates,

| **Graph-Based Label Propagation** | |
|---|---|
| enn | radius of k-nearest-neighbors. labels will not be propagated to nodes out of this range. |
| sigma | Gaussian kernel params. $w_{ij} = e^{-\|x_i - x_j\|^2/2\sigma^2}$ |
| iteration | label propagation iterations. |
| alpha | clamping factor. labeled nodes will be reset to initial value with clamping factor. |
| **SVR** | |
| epsilon | Epsilon in the epsilon-SVR model. It specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value. |
| **Neural Network** | |
| epoch | neural network model epoch number: one forward pass and one backward pass of all the training examples. |
| batch | neural network batch size: the number of training examples in one forward/backward pass. |
| **Common Parameters** | |
| epa | epa threshold when constructing training dataset. |
| seed | seeds number in training dataset. |

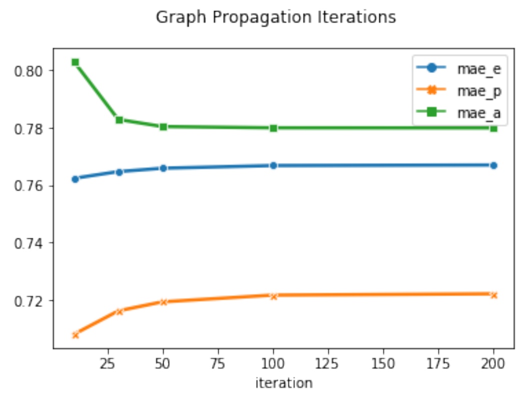
Table 4.4: Parameter reference table for three models used in this chapter, which are graph-based label propagation, support vector regression (SVR), and neural network. In the following sections, we first do experiments on parameter tuning for each model, then study on the common parameters. Finally we give comparisons of the best model using three algorithms.

evaluation, potency, and activity vary from the sensitivity to iterations. As propagation round increases to 50, the model performs significantly worse on potency axis and on the evaluation a little bit, while better on activity axis. Increasing iterations after 50 rounds could not make much difference to all three axes. Therefore we think 30 rounds should be the best iteration setting.

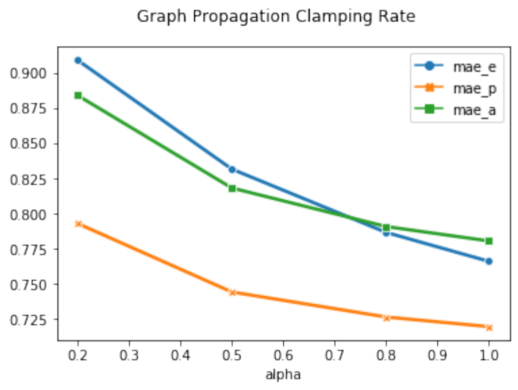
Another parameter in the graph label propagation algorithm is the clamping factor $\alpha$, which is a trade-off between consistency of the graph and the consistency of the original labels. Clamping to initial value enforces the algorithm converging to a global state. It is called a hard clamping if $\alpha = 1$, and relaxed clamping if $\alpha < 1$, for example $\alpha = 0.8$. We train the model on 8,500 words and evaluate the prediction performance on 1,000 unlabeled



(a)



(b)



(c)

Figure 4.7: Graph Propagation Parameter Tuning. Figure 4.7a shows the MAE values on three affective dimensions of the graph propagation models trained with various enn and sigma settings. Figure 4.7b shows how propagation iterations affect the model performance (MAE). Figure 4.7c demonstrates the relation between clamping factor and MAE value.

data selected randomly from the corpus and iterate for 50 rounds, and the result shows that the overall performance improves as $\alpha$ gets close to 1. Relaxed clamping might slow down learning rate. To make sure each node in the graph can converge to the solution, we increase propagation iterations to 50 rounds in this experiment.

#### 4.5.1.2 SVR

The value of $\epsilon$ specifies the margin within which no penalty is added to the loss function. Usually $\epsilon$ is set to 0.1 as default. A wider epsilon-margin relaxes the objective function in Equation 4.4, which might have a significant impact on the trained model.

We train SVR models on 8,500 labeled data and test the performance on 1,000 unlabeled words, just as in previous Section 4.5.1.1. As Fig 4.8 shows, SVR algorithm slightly achieves better results compared to the graph propagation method. Specifically, the MAE of evaluation, potency and activity reduce from 0.77, 0.72, 0.78 to 0.72, 0.75, 0.74 respectively. The value of $\epsilon$ has a different effect on the three dimensions as expected.

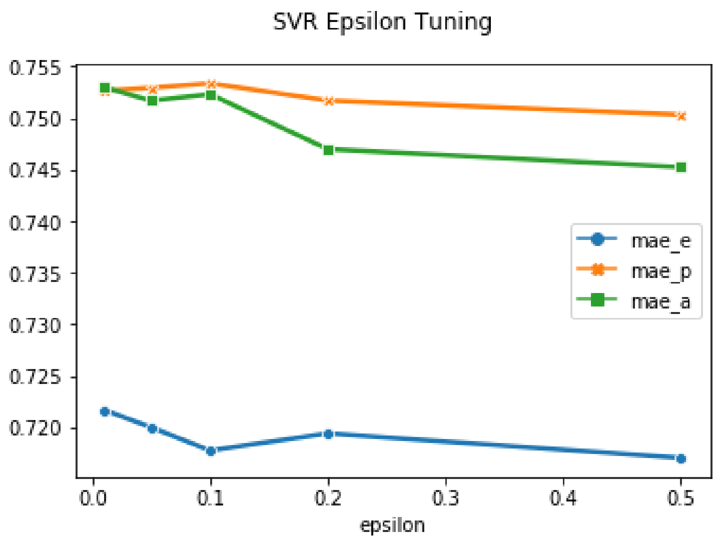


Figure 4.8: SVR Epsilon Tuning. We experiment on training SVR models with the epsilon ranging from 0 to 0.5. The figure shows how epsilon affect the perforamance (MAE) of three affective dimensions.

#### 4.5.1.3 Neural Network

Here we mainly discuss setting epoch and batch size in neural networks. To avoid overfitting and underfitting problems, we try to find the most appropriate training parameters. Just as previous experiments, we select 8,500 labeled data at random for training and evaluate on another 1,000 validation dataset logs. We plot the heatmap under epoch ranging from 5 to 200 and batch size ranging from 5 to 100. We can see from Fig 4.9, increasing training

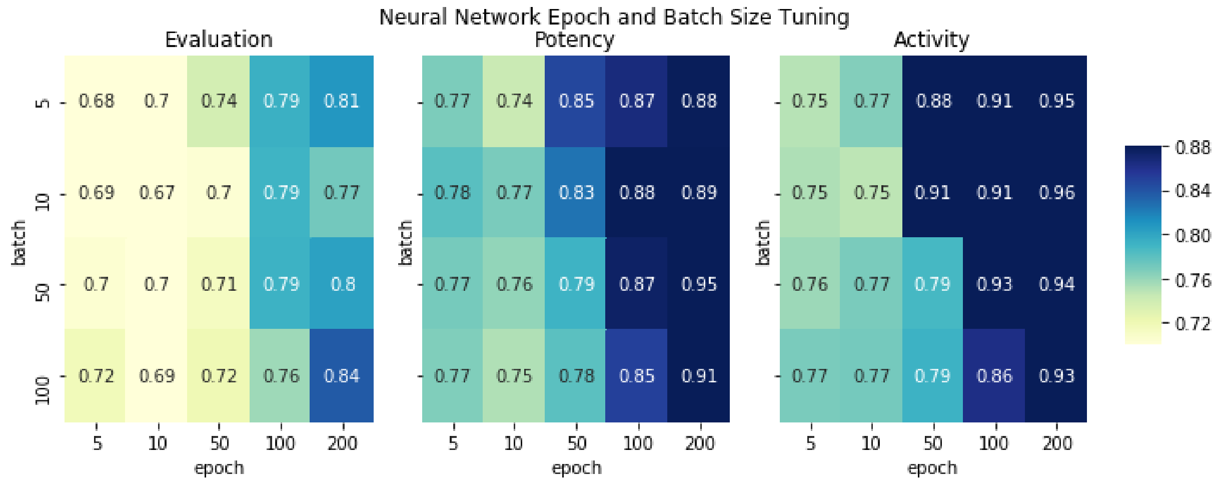


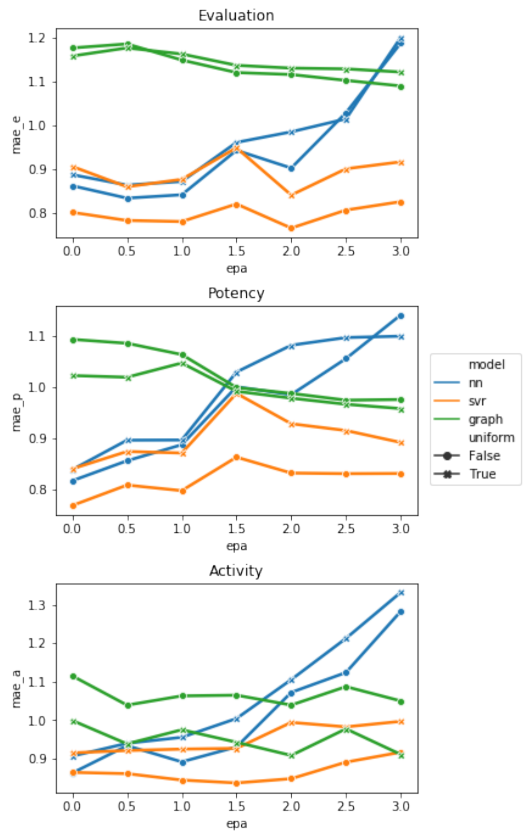
Figure 4.9: Neural Network Epoch and Batch Size. We experiment on training neural network models with different batch size and epoch settings. The figure shows the mean absolute errors (MAE) on evaluation-potency-activity dimensions.

epoch over 100 would hurt the performance significantly due to overfitting. However, batch size does not affect the performance a lot. 10 to 50 entries per batch should be good enough for our neural network model.

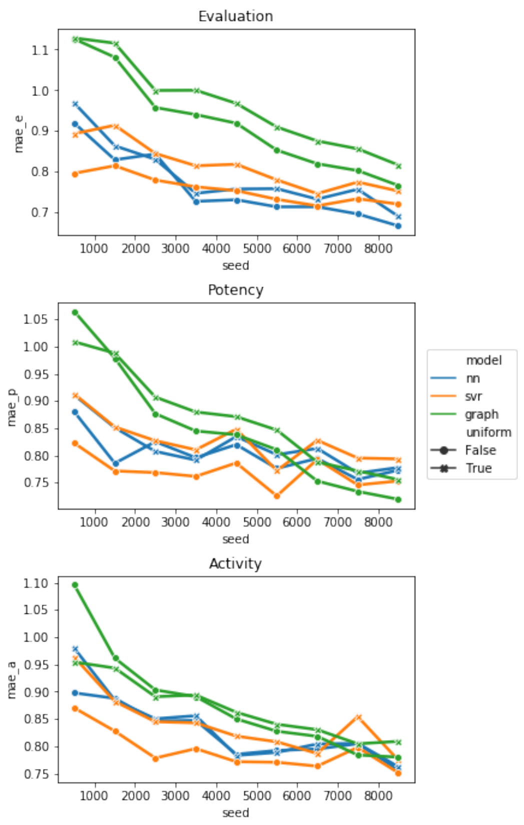
#### 4.5.1.4 Comparing Seed Words Threshold and Size

To evaluate three machine learning models, we compare the MAE results in Fig 4.10. As we mention in Section 4.4.2, we try three strategies to construct training dataset: fixed seed sets, random selection by EPA and random selection. Using fixed seeds did not work as expected, mainly because of the limitation of labeled data. In Fig 4.10, we explore constructing a training dataset of 600 words and testing on 1,000 words. We select lexicon from datasets whose EPA is over the set threshold at least in one dimension. It is a random selection when the EPA threshold is 0. We use the best-performing models from previous parameter tuning here. We use the following parameter settings for each model by default unless further specifying. Graph: $enn(0.6)$, $sigma(1.0)$, $iteration(50)$, $alpha(0.8)$; SVR: $epsilon(0.2)$; NN: $epoch(50)$, $batch(50)$.

Algorithms show variety to the EPA threshold sensitivity. The neural network model is the most sensitive one because the MAE increases significantly as the EPA threshold grows close to 2.0. On the contrary, graph-based semi-supervised model archives slightly better



(a)

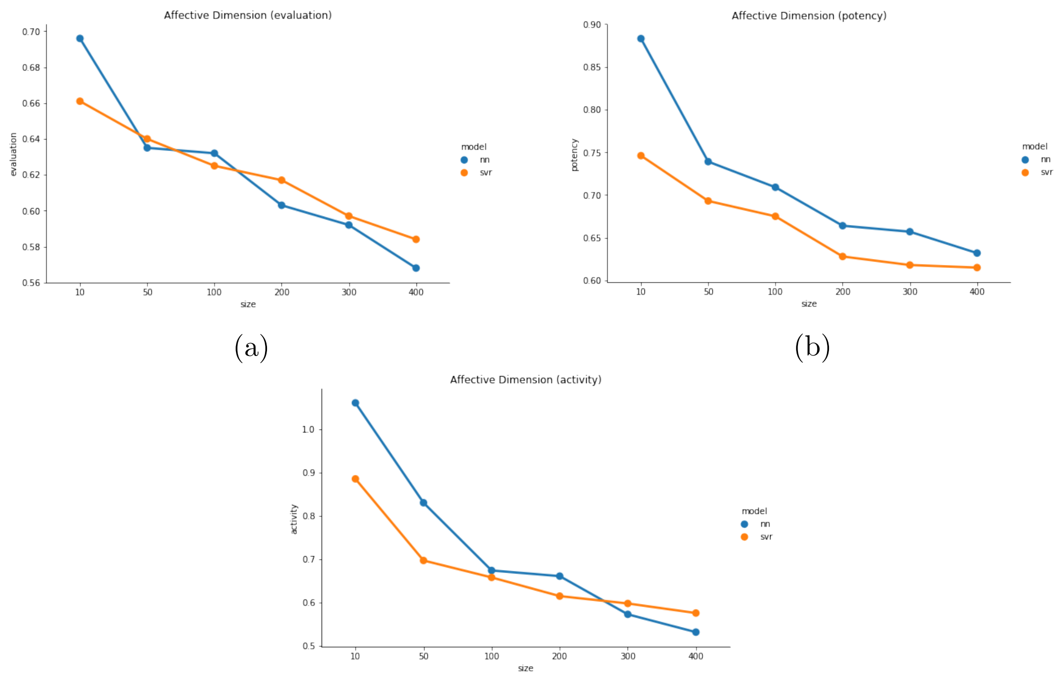


(b)

Figure 4.10: Parameter tunings for training dataset selection (parameter *epa* and *seed*). We experiment on three machine learning models described above: graph propagation (graph), SVR (svr), and neural network (nn). And we further test if applying uniformization algorithm described in Section 4.4.1.2 can improve the performance or not (denoted as *True* in the legend). Figure 4.10a and Figure 4.10b show the mean absolute error (MAE) on three affective dimensions of the three models with and without uniformization.

results under high threshold settings. SVR is more neutral and stable overall, however, there is a spike around 1.5.

We also test on a different scale of the training dataset. As it shows in Fig 4.10, in general, the prediction MAE reduces as training data increases. Graph propagation model is the most sensitive model in all. There is no significant difference across the three models if using more than 4,500 training data; however, the graph propagation method is



(a) (b)

(c)

Figure 4.11: A comparison of the supervised learning models performance (SVR and NN). Because of the limit of labeled data (vocab(Github-wv) $\cap$ vocab(Themis-EPA) $=$ 434 labeled concepts), we apply 5-fold cross validation.

not doing well in predicting the evaluation axis. Fig 4.10 also shows the results of using uniformization strategy introduced in Section 4.4.1.2. However, this improvement strategy does not decrease the error rate as we expect.

#### 4.5.1.5 Training Hidden Models within Github Subculture

Since we have obtained mapping models from Google-wv to Warriner-EPA, our next step is to predict Github-EPA using Github-wv in conjunction with space alighment and mapping models. Before applying $f(g)$ to approximate the hidden function $f'$, we would like to evaluate the performance of training models from Github-wv to Github-EPA directly as a benchmark.

We train models using three machine learning algorithms discussed above, however, input word vectors are from Github-wv and output affective ratings are from Themis-

| MAE | Evaluation | Potency | Activity |
|---|---|---|---|
| **Graph** | 0.69 | 0.71 | 0.83 |
| **SVR** | 0.61 | <u>0.62</u> | 0.60 |
| **NN** | <u>0.59</u> | 0.65 | <u>0.57</u> |

(a)

| MAE | Evaluation | Potency | Activity |
|---|---|---|---|
| **Graph** | 0.77 | <u>0.73</u> | 0.78 |
| **SVR** | 0.72 | 0.75 | <u>0.74</u> |
| **NN** | <u>0.67</u> | 0.74 | 0.75 |

(b)

Table 4.5: Two approaches of training mapping models. Figure 4.5a shows the mean error rate (MAE) of training on Github-wv word embeddings to predict 434 Themis-EPA ratings, and Figure 4.5b shows the mean error rate of our proposed approach, training on Google-wv word embeddings to 1,000 Warriner-EPA ratings.

EPA. Moreover, we experiment on using various scales of labeled concepts as training words. There are 434 labeled concepts available in the intersected vocabulary of Github-wv and Themis-EPA. We use 5-fold cross-validation to train the models. First, we split the 434 words into five pieces, and there are four subsets with 87 words and one subset with 86 words. Then we use each piece as the test dataset once, and the remaining four pieces as training dataset. There are 347 (or 348) labeled concepts in the training dataset after splitting. Of the training dataset, we select $k$ words at random for model training (denoted as *size*). The unselected words would not be used in this round. We experiment on training models on a scale of 50 to 400 labeled words, where words might be selected twice if selecting more than 347 (or 348). The MAE is the average result from each cross-validation fold. We do not apply cross-validation within the training process.

Figure 4.11 shows the results of using SVR and NN mapping models on different scale of training words. As the size of training words increases, the mean absolute errors decrease on three affective dimensions. SVR mapping models can receive slightly better results because we train a SVR model for each affective dimension. However, we use a single neural network model to predict three affective dimensions. The MAE values decrease greatly with more than 50 words and are as low as (0.57, 0.61, 0.55) if training with 400 labeled words. In Table 4.5, we further compare the performance of training mapping models in the Github subculture and general culture. Metrics in Table 4.5a are evaluated on 434 words using the 5-fold cross-validation method discussed above, while metrics in Table 4.5b are from 1,000 randomly selected concepts in general culture. In general, supervised learning methods (SVR and NN) can achieve better results than graph-based semi-supervised learning. Modeling affective ratings in general culture has higher mean error rate as it is shown. We think it is because Themis-EPA ratings have smaller standard deviations compared to Warriner-EPA (see Figure 4.5).

#### 4.5.1.6 Examples of Predicted EPA lexicon

Building on the results and discussions in the last section, we finally decide that the best way to do EPA lexicon induction and expansion with pre-trained word vector representations is to utilize artificial neural network models. To test this, we go beyond the random samples of 1000 as selected in the last section, instead we predict and evaluate on the whole Warriner-EPA dataset. The MAE values in this section are calculated on 13,790 labeled concepts. We select the seed words for model training process from the Warriner's dataset at random. To obtain the models with the best performance, we sample 8,500 concepts from the dataset using the method described in Section 4.4.2 with the dimension threshold set as 1.0. We train the models for ten epochs with ten entries per batch. Word vector representations trained on Google News are the input vectors of our network models. We predict the affect ratings for each word or phrase in Google-News-Word-Vectors and examine the results with ground truth collected by Warriner et al. We list part of the results in Table 4.6. There are 13,790 out of 13,915 concepts in the intersected vocabulary of Warriner-EPA and General-EPA, and the mean absolute errors of 13,790 words on three affective dimensions are 0.63, 0.70, 0.74 respectively.

### 4.5.2 Exploration on Expanded EPA lexicon

Since we have obtained models that could give relatively accurate predictions of word EPA ratings, we apply it to the Github corpus. From the experiments and discussion above, we determine to use artificial neural network models trained on 8,500 labeled words whose EPA are above 1.0 in at least one dimension. The training process takes ten epochs with ten entries per batch. Since the uniformization strategy does not improve prediction MAE at all, we do not use it here. As Chapter 3 shows, the EPA lexicon induction is performed on aligned word vectors of Github subculture. We use skip-gram word2vec model where vectors are 300 dimensional and min count sets as 20.

We further evaluate the induced EPA lexicon of Github corpus with a collected Github EPA lexicon (Themis-EPA). Of the 587 collected items representing people's cognition

| Word | Warriner-EPA | General-EPA | Absolute Error |
|---|---|---|---|
| **Good Predictions** | | | |
| disobey | [-2.017, -0.636, 0.91] | [-2.029, -0.64, 0.866] | [0.01, 0.0, 0.04] |
| heating | [1.13, 1.438, -1.507] | [1.099, 1.439, -1.523] | [0.03, 0.0, 0.02] |
| suffocation | [-2.904, -2.24, 0.424] | [-2.928, -2.267, 0.436] | [0.02, 0.03, 0.01] |
| ram | [-0.621, 0.664, -0.271] | [-0.593, 0.656, -0.316] | [0.03, 0.01, 0.04] |
| bureau | [-0.231, 0.553, -1.327] | [-0.197, 0.53, -1.379] | [0.03, 0.02, 0.05] |
| boarder | [0.302, 0.899, -1.313] | [0.337, 0.924, -1.369] | [0.04, 0.03, 0.06] |
| prejudiced | [-2.597, -1.383, 0.271] | [-2.55, -1.3, 0.292] | [0.05, 0.08, 0.02] |
| intruder | [-2.88, -2.226, 1.341] | [-2.866, -2.168, 1.421] | [0.01, 0.06, 0.08] |
| cranberry | [1.733, 1.314, -1.466] | [1.737, 1.379, -1.376] | [0.0, 0.06, 0.09] |
| esteem | [1.757, 2.295, -1.091] | [1.802, 2.201, -1.1] | [0.05, 0.09, 0.01] |
| pencil | [0.893, 1.714, -2.202] | [0.795, 1.675, -2.215] | [0.1, 0.04, 0.01] |
| seam | [0.42, 0.747, -1.66] | [0.419, 0.671, -1.744] | [0.0, 0.08, 0.08] |
| jelly | [1.189, 1.231, -1.48] | [1.242, 1.17, -1.535] | [0.05, 0.06, 0.05] |
| degradation | [-2.419, -1.936, -0.604] | [-2.409, -1.815, -0.576] | [0.01, 0.12, 0.03] |
| floozy | [-1.118, -0.567, 0.271] | [-1.205, -0.548, 0.221] | [0.09, 0.02, 0.05] |
| negligee | [1.544, 1.175, -0.035] | [1.508, 1.198, -0.142] | [0.04, 0.02, 0.11] |
| cram | [-0.254, 0.539, -0.702] | [-0.319, 0.572, -0.769] | [0.07, 0.03, 0.07] |
| exorcism | [-2.064, -1.493, 1.758] | [-2.025, -1.588, 1.706] | [0.04, 0.09, 0.05] |
| limo | [1.473, 1.507, -0.757] | [1.47, 1.542, -0.614] | [0.0, 0.04, 0.14] |
| herring | [0.006, 0.539, -1.869] | [0.031, 0.47, -1.781] | [0.03, 0.07, 0.09] |
| prone | [-0.231, 0.843, -1.41] | [-0.25, 0.71, -1.357] | [0.02, 0.13, 0.05] |
| nest | [0.893, 0.982, -1.869] | [0.827, 1.072, -1.833] | [0.07, 0.09, 0.04] |
| sprite | [1.13, 1.231, -1.202] | [1.121, 1.413, -1.213] | [0.01, 0.18, 0.01] |
| pyramid | [0.929, 1.272, -1.688] | [0.777, 1.256, -1.652] | [0.15, 0.02, 0.04] |
| **Bad Predictions** | | | |
| download | [1.307, 2.032, -1.521] | [0.789, 1.706, -1.244] | [0.52, 0.33, 0.28] |
| tourist | [0.964, 0.359, -1.563] | [1.18, 0.779, -1.069] | [0.22, 0.42, 0.49] |
| specific | [0.692, 1.521, -2.98] | [0.751, 1.719, -1.611] | [0.06, 0.2, 1.37] |
| margin | [-0.172, 0.498, -1.368] | [0.225, 1.222, -1.875] | [0.4, 0.72, 0.51] |
| smarty | [1.733, 2.613, -0.896] | [1.241, 1.488, -0.884] | [0.49, 1.12, 0.01] |
| taxi | [-0.124, -0.359, -1.257] | [0.015, 1.072, -1.197] | [0.14, 1.43, 0.06] |

Table 4.6: Examples of the results of our mapping model $f$ (NN). We take word embeddings Google-wv as input to the model, and compare the predictions (General-EPA) with ground truths (Warriner-EPA).

| Method | | LookUp | | | General | | | GitHub | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dimension | | E | P | A | E | P | A | E | P | A |
| **Part** 371 words | ACC | 330 | 308 | 315 | 352 | <u>354</u> | 342 | <u>358</u> | 353 | <u>358</u> |
| | MAE | 0.764 | 0.970 | 0.970 | 0.660 | 0.746 | 0.672 | <u>0.573</u> | <u>0.660</u> | <u>0.580</u> |
| | RMSE | 0.962 | 1.175 | 1.281 | 0.835 | 0.893 | 0.944 | <u>0.757</u> | <u>0.833</u> | <u>0.813</u> |
| **All** 428 words | ACC | - | - | - | 407 | <u>409</u> | 397 | <u>412</u> | 407 | <u>413</u> |
| | MAE | - | - | - | 0.639 | 0.734 | 0.645 | <u>0.579</u> | <u>0.664</u> | <u>0.564</u> |
| | RMSE | - | - | - | 0.813 | 0.879 | 0.910 | <u>0.768</u> | <u>0.833</u> | <u>0.793</u> |

Table 4.7: Comparison of the performance of using three methods. There are two subsets in the table, *part* is the intersected concept vocabulary of Warriner-EPA, General-EPA, Github-EPA, and *all* is the intersected concept vocabulary of General-EPA and Github-EPA. We calculate the prediction accuracy (ACC), mean absolute error (MAE), and root mean square error (RMSE). ACC is calculated by counting the number of predictions that are within the range of (mean - std, mean + std) in Themis-EPA. The highest score are underlined in the table.

on Evaluation, Potency, and Evaluation in Github community, there are 411 concepts are predicted from the word vector Github-wv to the EPA model Github-EPA using the approach introduced in this thesis. We evaluate the prediction performance on the Github community using mean absolute error (MAE) rate and root mean squared error (RMSE) rate. We want to verify that with the help of word embeddings, our proposed method could capture subtle affective differences across subcultures. We compare the error rates of using three methods (Warriner-EPA, General-EPA and Github-EPA) to the collected EPA ratings in the Github community Themis-EPA:

- *LookUp*: use affect ratings in Warriner's dataset if the concept exists (Warriner-EPA).
- *General*: predict affective ratings with trained models, general word embeddings as input vectors (General-EPA).
- *Github*: predict affective ratings with trained models, the pre-trained and aligned(see Chapter 3) Github word embeddings as input vectors (Github-EPA).

As it shows in Table 4.7, our proposed method outperforms the other two methods. In general, our proposed method performs well on all three dimensions, with slightly higher MAE on potency. We think it is because the mapping model we utilized, neural network, does not perform very well (see Table 4.5b). Even though our model is trained the general

| **Word** | **Warriner-EPA** | **Themis-EPA** | **Github-EPA** | **General-EPA** |
|---|---|---|---|---|
| **Good Examples** | | | | |
| option | [1.75, 2.17, -2.09] | [0.52, 1.12, -1.42] | [0.56, 0.87, -1.43] | [1.16, 1.54, -1.48] |
| core | [1.53, 1.52, -2.02] | [0.4, 0.69, -1.55] | [0.18, 0.76, -1.58] | [1.0, 1.86, -1.86] |
| code | [-0.27, 0.06, -2.69] | [0.1, 0.64, -1.27] | [-0.07, 0.62, -1.52] | [0.4, 1.33, -1.5] |
| super | [2.55, 2.93, -0.62] | [0.66, 1.43, -1.08] | [0.68, 1.28, -1.37] | [1.8, 1.47, 0.59] |
| promise | [2.06, 2.32, -1.1] | [1.27, 1.54, -0.85] | [1.45, 1.66, -1.1] | [2.98, 2.08, 0.24] |
| introduce | [1.2, 2.83, -0.2] | [0.66, 0.95, -1.46] | [0.41, 1.12, -1.59] | [1.45, 1.72, -0.95] |
| variable | [0.47, 0.58, -1.45] | [0.39, 0.75, -1.57] | [0.05, 0.61, -1.64] | [0.02, 0.45, -1.73] |
| statement | [0.5, 2.9, -1.38] | [0.16, 0.62, -1.41] | [0.12, 0.88, -1.63] | [1.1, 1.64, -1.5] |
| skilled | [2.55, 3.65, 0.42] | [1.24, 1.66, -0.68] | [1.04, 1.6, -1.08] | [1.88, 2.06, -0.12] |
| contain | [0.12, 1.23, -2.35] | [0.21, 0.53, -1.46] | [0.21, 1.19, -1.45] | [0.23, 1.28, -1.38] |
| assignment | [0.59, 1.67, -0.34] | [0.23, 0.8, -1.44] | [-0.1, 0.88, -1.16] | [0.61, 1.71, -1.25] |
| solve | [1.66, 2.6, -0.48] | [1.0, 1.25, -0.99] | [0.91, 1.54, -1.4] | [1.61, 2.06, -1.26] |
| performance | [1.54, 2.99, 1.27] | [0.8, 1.52, -0.71] | [1.26, 1.41, -0.6] | [1.87, 2.21, -0.66] |
| **Not So Good Examples** | | | | |
| unfriendly | [-3.07, -0.47, -0.9] | [-2.63, -1.25, -2.8] | [0.34, 1.29, -1.12] | [-2.3, -1.24, -0.36] |
| unreliable | [-2.55, -0.79, -0.56] | [-2.91, -1.32, -3.26] | [-1.34, -0.62, -0.97] | [-2.36, -1.82, -0.76] |
| useless | [-2.48, -0.19, -0.42] | [-2.68, -1.78, -3.18] | [-1.89, -0.98, -1.15] | [-2.27, -1.32, -1.13] |
| quiet | [1.86, 1.4, -3.81] | [-0.08, -0.31, -3.33] | [-0.03, 1.5, -1.39] | [1.51, 2.09, -2.06] |
| familiar | [1.53, 3.35, -3.05] | [0.66, 0.72, -1.63] | [1.67, 1.99, 0.16] | [1.91, 2.39, -1.89] |
| regular | [1.41, 2.7, -1.45] | [-0.22, 0.1, -2.02] | [1.34, 1.93, -1.98] | [0.38, 1.59, -2.07] |
| concerned | [-0.34, -0.25, -1.03] | [-0.59, -0.3, -2.33] | [-1.01, 0.62, -0.73] | [-0.2, 1.3, -0.82] |

Table 4.8: Examples of predicted Github concepts' EPA in detail.

culture using Google news word vectors, it receives better MAE and RSME scores on Github lexicon on Evaluation and Activity.

To further demonstrate the necessity of using our proposed method for affective lexicon induction, instead of training models on Themis-EPA directly (denoted as hidden model), we evaluate the performances with Table 4.5a and Figure 4.11. Testing on 434 labeled concepts, the MAE values on evaluation-potency-activity space of using our approach are 0.58, 0.66, and 0.56 respectively. If training with 300 labeled concepts with 5-fold cross-validation, the MAE values of training hidden function $f'$ are 0.59, 0.65, and 0.57, which is close to our proposed approach. Therefore, using our method can save the cost of labeling 300 concepts manually. One shortcoming of training on Themis-EPA dataset directly is that concepts in Themis-EPA dataset are mainly in the topic of software and programming. We cannot ensure the performance when applying to a different topic. However, we think our proposed method can do well because it learns from the whole space. Comparing the results with Table 4.5a, our proposed approximation method $f(g)$ is more feasible than training the $f'$ model directly.

We list part of our prediction with collected data in Table 4.8. Most verbs and nouns could get good predictions, for example, option, core, introduce, solve, etc. The EPA ratings obtained from our approach are very close to those collected from Github community members. However, adjectives are biased greatly, for example, quiet, unfriendly, and concerned. The word *unfriendly* is seen as very bad and powerless both in the general culture and the Github subculture. But the predicted result from our model shows slightly good and very powerful, which does not agree with the ground truths.

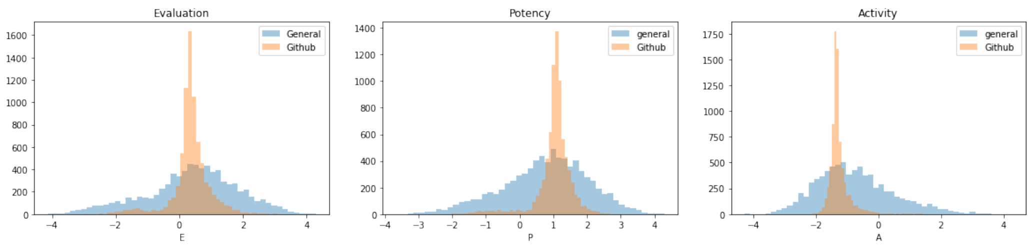


Figure 4.12: Comparison of EPA distribution between general culture Themis-EPA and Github community Github-EPA. The evaluation, potency, and activity ratings in Github subculure are more concentrated with smaller standard deviations, comparing with the general culture.

With the induced EPA lexicon on Github community, we can see how affective meanings are different across subcultures. We give some examples in Table 4.9. Most nouns

| **Word** | **Warriner-EPA** | **Github-EPA** | **Absolute Difference** |
|---|---|---|---|
| | | **Verb** | |
| ensure | [0.964, 1.618, -1.563] | [1.066, 1.674, -1.527] | [0.1, 0.06, 0.04] |
| assign | [-0.172, 1.314, -1.368] | [0.011, 1.359, -1.241] | [0.18, 0.04, 0.13] |
| handle | [0.988, 1.673, -1.688] | [0.662, 1.691, -1.661] | [0.33, 0.02, 0.03] |
| revise | [0.432, 1.604, -1.632] | [0.572, 1.641, -1.428] | [0.14, 0.04, 0.2] |
| regulate | [0.313, 1.106, -1.035] | [0.349, 1.144, -1.38] | [0.04, 0.04, 0.34] |
| assist | [1.248, 2.24, -1.299] | [1.377, 1.896, -1.371] | [0.13, 0.34, 0.07] |
| | | **Noun** | |
| horse | [1.366, 1.272, -0.743] | [1.236, 1.265, -0.82] | [0.13, 0.01, 0.08] |
| booklet | [0.739, 1.148, -1.591] | [0.643, 1.047, -1.562] | [0.1, 0.1, 0.03] |
| four | [0.432, 1.189, -1.813] | [0.464, 1.189, -1.595] | [0.03, 0.0, 0.22] |
| academy | [0.503, 0.816, -1.285] | [0.421, 1.035, -1.328] | [0.08, 0.22, 0.04] |
| vendor | [0.467, 0.982, -1.216] | [0.717, 0.962, -1.137] | [0.25, 0.02, 0.08] |
| sow | [0.254, 1.314, -1.368] | [0.293, 1.067, -1.305] | [0.04, 0.25, 0.06] |
| | | **Adjective** | |
| unavailable | [-1.627, -1.549, -1.174] | [-1.616, -1.339, -1.045] | [0.01, 0.21, 0.13] |
| dense | [-0.254, 0.401, -1.535] | [-0.368, 0.391, -1.406] | [0.11, 0.01, 0.13] |
| similar | [1.142, 1.396, -1.758] | [1.005, 1.43, -1.598] | [0.14, 0.03, 0.16] |
| conducive | [0.586, 1.369, -1.174] | [0.516, 1.284, -1.356] | [0.07, 0.08, 0.18] |
| unavailable | [-1.627, -1.549, -1.174] | [-1.616, -1.339, -1.045] | [0.01, 0.21, 0.13] |
| compatible | [1.65, 2.074, -0.91] | [1.493, 1.842, -1.087] | [0.16, 0.23, 0.18] |
| | | **Emotion** | |
| amazed | [3.141, 1.825, 1.744] | [0.747, 1.411, -1.208] | [2.39, 0.41, 2.95] |
| calm | [2.36, 3.664, -4.203] | [0.409, 1.139, -1.359] | [1.95, 2.53, 2.84] |
| confused | [-2.112, -0.913, -0.577] | [-2.46, -1.19, -0.015] | [0.35, 0.28, 0.56] |
| happy | [4.229, 3.346, 1.883] | [2.815, 2.672, -0.387] | [1.41, 0.67, 2.27] |
| glad | [3.141, 3.056, -1.368] | [3.02, 2.358, -0.199] | [0.12, 0.7, 1.17] |
| anxious | [-1.295, -0.885, 2.091] | [0.33, 1.068, -1.187] | [1.62, 1.95, 3.28] |
| amused | [2.549, 1.576, -0.59] | [0.407, 1.208, -1.284] | [2.14, 0.37, 0.69] |
| afraid | [-3.129, -2.876, 0.59] | [-0.906, -0.031, -0.338] | [2.22, 2.84, 0.93] |

Table 4.9: Detailed comparison of EPA lexicon between general culture and Github community.

and verbs agree on the polarity in both cultures and have slight differences regarding affective dimension strength. However, we find words representing human sentiments have significant differences across the two subcultures. The word *happy* is (4.2, 3.3, 1.8) in general culture context, however, it is (2.8, 2.6, -0.4) as predicted in Github community. Similar to the word *glad*, which is (3.1, 3.0, -1.4) in general environment while (3.0, 2.3, -0.2) in Github. Generally, sentiment words in Github are not as active as they are in the general environment. The word *afraid* tends to be more neutral because its EPA reduces from (-3.1, -2.8, 0.6) to (-0.9, -0, -0.3). To verify our hypothesis, we plot the distributions of each affective dimension (see Fig 4.12). It is interesting that words and phrases in Github are more neutral and with a lower standard deviation. We conclude two possible reasons for this. Firstly, people treat the Github community as a workplace, instead of social media. Therefore, they do not express much sentiment in the texts and try to be neutral in general. Secondly, people in Github subculture are mainly programmers who know about software/hardware, and they are talking about those topics most of the time. The composition of members and topics might have an impact on the affective space.

## 4.6 Conclusion

In this chapter, we introduce a new method of label induction using neural networks. From the previous work of Alhothali et al. [2], we implemented the EPA label prediction with graph propagation at first. We also compare it with using supervised learning methods, supported vector machine and artificial neural network. The results show that well-tuned neural network could receive the lowest prediction error in general. We also explore the impacts of different seed words selection strategy and label uniformization, which are not thoroughly investigated by any researchers before. The best model trained could hit no more than 0.7 MAE on average, which is much better results compared to previous work (MAE $<$ 1.1) [1, 2].

By the well-tuned neural network model, we further predict words EPA labels for Github community. According to most studies, EPA distributions show varieties between cultures to some extent. Taking the Github word vectors obtained in Chapter 3, we predict and extend the EPA lexicon for Github community. To evaluate the performance, we compare the prediction with data collected online. The result shows our prediction is very close to empirical studies with (0.5, 0.6, 0.5) MAE value on Evaluation, Potency and Activity dimensions.

We give further comparisons and list the differences in people's cognitions between cultures. It shows obvious differences, especially regarding words that represent sentiments. We find that in general affective meanings of nouns and verbs are highly agreed across cultures. However, people show different understandings and reactions regarding words related to sentiments and emotions. It is probably related to the nature of collaboration on Github community.

# Chapter 5

# State Prediction

## 5.1 Introduction

Natural language texts are meant to express or impact individuals' sentiments. Studies have shown that the underlying sentiments expressed or triggered by sentences are highly related to conversational contexts. People speak and react differently to groups of people under various circumstances. For example, people tend to be frank and direct talking with families and friends and feel the most comfortable. However, we try to be careful and sometimes a bit nervous communicating with workmates. Another example is that objective statements present factual information (e.g., a baby dies after being left in the car for over 8 hours) are taken more seriously posted in the newspaper than social media.

In this chapter, we focus on analyzing individuals' sentimental reactions triggered by sentences in different context settings. Using the extended Evaluation-Potency-Activity lexicon obtained in Chapter 4, we show the feasibility of subculture sentiment analysis without manually collected sentiment lexicon. Sentences are decomposed into a subject-verb-object (SVO) model, and we compute the predicted sentiment towards each of the components. The research work of Affect Control Theory (ACT) provides a computational regression model of EPA to handle affect state prediction of contextualized concepts. One drawback of ACT is that the regression model can change under different cultural contexts. Therefore, we would not use ACT in this thesis, mainly because of the lack of the subculture's regression models. Instead, we propose to use the Long Short-Term Memory (LSTM) network to infer the affective states. Performance evaluation is conducted using a news-headlines dataset which has a subject, an act and an object annotated by participants on Mechanical Turk [1]. The result shows that LSTM using word embedding representation or inducted EPA lexicon performs as well as using the original ACT model. We also experiment on applying the model to the Github corpus. Due to the lack of human annotated affective states on Github, we could not evaluate our prediction. However, the results indicate there is less controversy in the Github community.

## 5.2 Background

### 5.2.1 Related Work

Sentiments have been studied extensively in many fields like computer science, psychology, and sociology in recent years. Although there is psychological evidence showing sentiments are perceived in more than one dimension, most proposed theories adopt discrete appraisal theory by representing sentiments with discrete labels, such as positive, negative, neutral, or happy, sad. Some commonly used features in sentiment analysis include part-of-speech tagging, n-grams, term weighting, sentiment lexicons, and syntactic dependencies [71]. Unsupervised learning approaches usually calculate the difference between the point-wise mutual information (PMI) and determine the semantic orientation of sentence or document. Advanced approaches build more structured models by taking semantic taxonomy, the interaction of words or bag-of-words techniques into consideration. A framework is proposed by Coecke et al. which constructs a sentence vector as a function of its word vectors [14]. There are several recent studies tackled to predict readers' sentiments instead of from the writers' perspective.

Ahothali and Hoey explore predicting sentiments for each news headline using an augmented EPA lexicon and ACT equations [1]. They decompose sentence into subject-verb-object and associate subject with the actor, verb with the behavior and object with the object in ACT. The collected news headline dataset consists of 2,080 headlines from a group of news websites from the period of 1999 to 2014. Recruited participants from Mechanical Turk are asked to locate the subject, behavior, and object of each sentence and indicate their sentiments in the EPA format. Their approach yields a precision between 71% and 82% on the news headline dataset, which outperforms most supervised learning models.

MingLei Li et al. further propose to use LSTM learning method using word embeddings to predict the affective states of a described event [54]. They claim the method outperform the linear model in the ACT and most importantly, there is no need to construct affect lexicons manually. They perform the research on a corpus of 515 sentences in SVO forms with a vocabulary of 106, which are annotated by 25 females and 25 males. The best

LSTM model in their work achieves around 0.3 error rate on three affective dimensions. This paper inspires us to apply supervised learning models instead of using ACT equations.

### 5.2.2 LSTM

Humans' thinking does not start from scratch every second, but instead, it is based on understanding of previous words and the context. Traditional machine learning models and neural networks could not do this as they lack of memory mechanisms. Recurrent neural networks address this issue by adding loops in the network architecture, which allows the output information from previous input to persist in future incoming data. If we unroll the loop, a recurrent neural network can be thought of as multiple copies of the same network. The chain-like nature of recurrent neural network reveals the relationship to sequences and lists, which makes it receive tremendous success in various natural language processing tasks, including speech recognition, translation, sentiment analysis, etc.

One major problem in recurrent neural networks is the gap between relevant information and the point where it is needed. Sometimes the gap is small, for example, if we are trying to predict the last word in "the clouds are in the *sky*". However, there are also cases where we need more context, for example, if trying to predict the last word in the text "I grew up in France... I speak fluent *French*". In this case, the gap becomes very large and recurrent neural networks become unable to connect that information in practice. Long Short-Term Memory networks (LSTM) is an extension of recurrent neural network and capable of learning long-term dependencies [41, 26]. It keeps the chain-like structure, but the repeating module has a unique structure called gates.

Gates are designed to let information through optionally. They are composed of sigmoid neural net layers and a pointwise multiplication operation. A LSTM cell at position $t$ consists of four parts: an input gate vector $i_t$, a forget gate vector $f_t$, an output gate vector $o_t$ and a cell state vector $c_t$. Furthermore, the output of each cell is denoted as $h_t$. An LSTM structure can be described by the following sets of equations:

$$\begin{gathered}
i_t = \sigma\ (U_i x_t + W_i h_{t-1} + b_i) \\
f_t = \sigma\ (U_f x_t + W_f h_{t-1} + b_f) \\
o_t = \sigma\ (U_o x_t + W_o h_{t-1} + b_o) \\
c_t = f_t \circ c_{t-1} + i_t \circ tanh(U_c x_t + W_c h_{t-1} + b_c) \\
h_t = o_t \circ tanh(c_t)
\end{gathered} \tag{5.1}$$

where $\sigma$ is the sigmoid activation function, $\circ$ is the element wise produce, $b_i, b_f, b_o, b_c$ are the bias. The cell produces current output $h_t$ based on the current input $x_t$ and previous output $h_{t-1}$. $U_i, U_f, U_o, U_c, W_i, W_f, W_o, W_c$ are model matrix that are learning during the training process.

Our proposed LSTM model consists of three modules, which takes the word vector representations of subject-verb-object extracted from sentences. We do a final classification on the last hidden layer to predict sentiments. We train the LSTM model for each sentiment state of each role in an event. Fig 5.1 shows the architecture of LSTM.

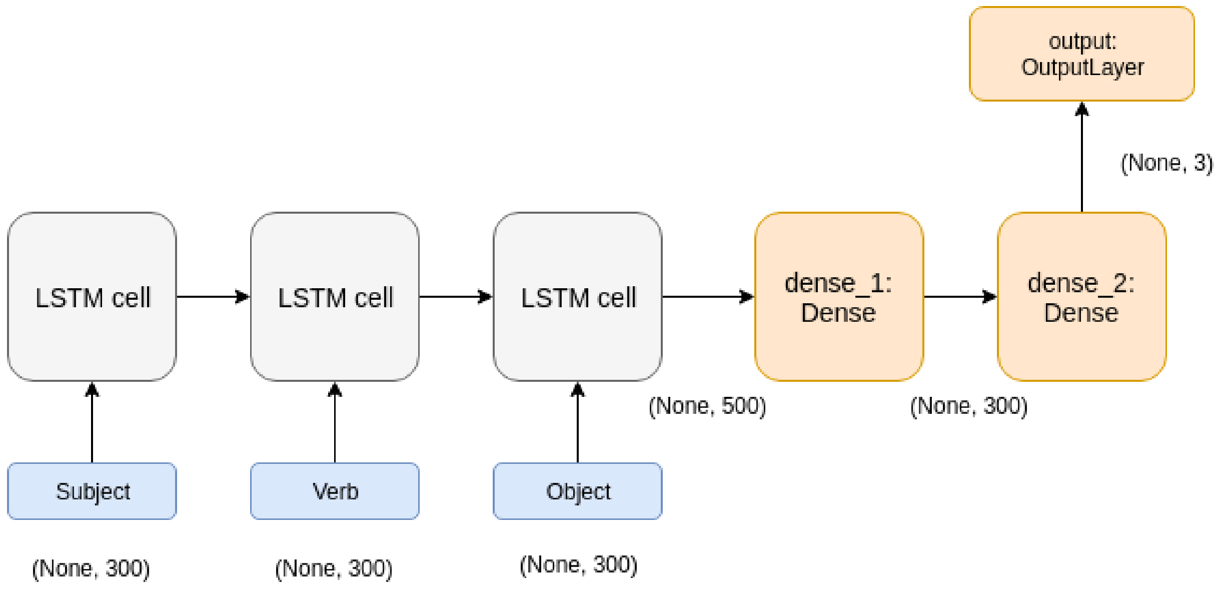


Figure 5.1: The structure of LSTM used to predict EPA states of SVO model. The architecture consistes three layers: the input SVO word embeddings layer, the LSTM layer, and the output affective state layer. There are two hidden dense layers connected between the outputs from LSTM layer and the final output.

## 5.3 Evaluation

To train and evaluate the performance of predicting subject and object's sentiments evoked from event-based sentences, we use a news headlines dataset collected by Ahothali [1]. The dataset consists 2,080 news headlines from a group of news websites and archives (BBC, CNN, Reuters, The Telegraph, Times) ranging from 1999 to 2014. Participants located in

North America are recruited with an approval rate above 90%. They are asked to locate the subject, behavior, and object of each sentence and rate each component's sentiment in the format of EPA. They have excluded ratings filled with blanks, zeros or similar values. The inappropriate subject, verb, and object from are also excluded.

We also try to use a Github comment dataset collected by Rishi [75]. This dataset consists of 834 pull requests from Github and a total of 3,000 comments. Each comment is labeled by four people regarding the sentiments expressed. The ten sentiments are *thanks, sorry, calm, nervous, careless, cautious, aggressive, defensive, happy*, and *angry*.

To evaluate the performance of our proposed LSTM method, we test the MAE error rate between prediction and ground truth. To obtain sentiment from Evaluation-Potency-Activity space, we calculate and sort Euclidean distances between predicted EPA position and ten anchor sentiment words mentioned above. Since word embeddings are used as LSTM model inputs, we map EPA to sentiments using the affective meanings in Warriner-EPA or Github-EPA, depending on which culture or subculture to analyze.

## 5.4 Results and Discussion

### 5.4.1 SVO EPA Prediction

Our proposed LSTM model is first evaluated in predicting the evoked sentiments towards news headlines by comparing the generated EPA to the ground truth. We extract subject-verb-object component from sentences using NLTK Stanford Parser and fill failed parsing values with human annotations collected by Ahothali [1]. The Stanford Parser constructs a probabilistic parse tree representing the structure of the sentences to identify grammatical components. Taking the word vector representations of subject, verb, and object, we train separate LSTM models to predict sentiments towards the event, subject, behavior, and object in the EPA dimensions. We set the dropout rate for LSTM layer to be 0.2 and add two dense layers before the final output layer. There are 1,535 news headlines in the labeled dataset, so we apply k-fold cross validation where k = 5. The results are shown in Table 5.1.

We further attempt to map predicted EPA values to ten basic sentiments mentioned above. Due to the lack of human-labeled sentiments evoked by news headlines, we do not have accurate evaluations. Instead, we calculate cosine distances on EPA dimension to infer sentiments, which allows us to get a sense of our models' performance. We utilize the Warriner-EPA as the sentiment dictionary. The sentiment that has the smallest cosine

| Identity | **Event** | | | **Subject** | | | **Behavior** | | | **Object** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dimension | E | P | A | E | P | A | E | P | A | E | P | A |
| **MAE** | 0.56 | 0.60 | 0.58 | 0.53 | 0.55 | 0.55 | 0.53 | 0.57 | 0.57 | 0.56 | 0.56 | 0.57 |
| **RMSE** | 0.76 | 0.81 | 0.80 | 0.74 | 0.77 | 0.72 | 0.79 | 0.82 | 0.79 | 0.77 | 0.77 | 0.77 |

(a) The prediction results of using our proposed LSTM approach.

| Identity | **Subject** | | | **Object** | | |
|---|---|---|---|---|---|---|
| Dimension | E | P | A | E | P | A |
| **MAE** | 1.11 | 1.25 | 1.27 | 1.09 | 1.26 | 1.27 |
| **RMSE** | 1.38 | 1.24 | 1.33 | 1.33 | 1.56 | 1.54 |

(b) The results from previous work. [1].

Table 5.1: Affective prediction of 1,535 event-based sentences. MAE=mean absolute error. RMSE=root mean squared error. LSTM model can achieve better results compared with using ACT [1].

distance to human annotated EPA ratings is taken as human annotated sentiment. The sentiment that has the smallest cosine distance to predicted EPA ratings using LSTM model is taken as model predicted sentiment. We select the top three out of ten sentiments and add one score if there is at least one element in common. Of the 1,532 news headlines, there are 1,441 and 1,429 hits for predicting the subject's and object's sentiments respectively. Here we list part of our predicted sentiments which look quite reasonable (see Table 5.2).

### 5.4.2 Sentiment Prediction on Github Comments

With the well-trained LSTM models that could predict EPA scores from subject-verb-object word embeddings in sentences, we further test on a Github comments dataset. Before we input data to our models, we first split and extract SVO components from Github comments. Using the Stanford Parser, there are only 976 sentences left, which is probably because the collected comments are not well formatted, which increases the difficulty of grammatical components extraction. We conclude three reasons for the Parser performs poorly on the Github comments dataset.

- Github comments are very casual sentences and lack of enough context.
  For example: "Done. Renamed *timestamp* to *freeze-date*", "ah yeah I missed that one, typo. fixed", and ":) Ya probably not.". In these examples, there are not

| **Headline** | **Ground Truth** | | **Model Prediction** | |
|---|---|---|---|---|
| | ETS | ETO | ETS | ETO |
| Bomb *kills* 18 on military bus in Iran | careless | cautious | careless | sorry |
| Massive earthquake *strikes* Chile | aggressive | aggressive | sorry | nervous |
| House speaker *offers* comprehensive pension fix | happy | happy | happy | cautious |
| Behind the Masks in Ukraine, Many *Faces* of Rebellion | aggressive | aggressive | aggressive | aggressive |
| Stomp the Yard" has *winning* moves in its weekend debut | happy | nervous | happy | cautious |

Table 5.2: LSTM model's results on predicting basic sentiments. ETS and ETO are sentiments towards subject and object respectively. SVO elements extracted by parser are denoted as: subject, *behavior*, and object.

necessarily to be a subject, verb or object and group members can understand clearly. However, programs fail to analyze the structure.

- Abbreviations and terminologies are commonly used in the Github community. For example, "I haven't really grasped that code, but comparing the diff where that code was introduced, maybe should it use?" and "Nope, I don't have this in my repo local repo. So it isn't my code at all". Word *diff* and *repo* are abbreviations for "difference" and "repository". It is hard for the program to tell the true meanings in the Github culture.

- A few comments are not collected properly in the raw data. We are not sure about the reason, but it is obvious that long sentences are partly cut off. For example, "As this enum is public, it might make sense to give it a more C# friendly name... How do you feel about,, ...", and "Bug, as it's... again instead of ... Fixing this with my refactor and adding a unit test because I am getting afraid :)" There is no way for the parser to understand the sentence structure correctly.

Even though the SVO Parser does not work well on the Github comments data, we still try to evaluate the performance of predicting evoked sentiments. Of the 976 comments, we pick the top three out of ten sentiments as our prediction and add one score if any predicted sentiment appears in the ground truth set. For example, if the top three sentiments predicted by the model are *calm, sorry, happy*, and the human annotation is *sorry*, we count it as a successful prediction (hit). Otherwise, it is not a hit if the human annotation is *sad*. There are 215 hits in the result if selecting the top three predicted sentiments. Moreover, there are 466 hits if increasing the threshold to top four. Here we show part of our results in Table 5.3. Even though the results are not as good as using state-of-art methods [76, 75], there are some reasonable predictions.

| **Comments** | **Human** | **Prediction** |
|---|---|---|
| The easiest fix would be to just change that line to since the only reason it's there is to define something the windows headers don't. | calm, cautious | sorry, cautious |
| Indeed. I think the "extra" is a typo – sort of redundant. I'll fix this in the docs. | calm, sorry | sorry, thanks, cautious |
| Ouch! My bad on this one. I reverted this back to the original implementation. | careless, sorry | sorry, thanks, cautious |
| I don't think this is right. | aggressive, cautious | sorry, cautious, careless |
| I like the name change. is more granular and clear to me | calm | thanks, calm |
| I've got an OSX machine here so I'll be sure to test any changes I make from hereon out. Though I think the C code is safe now :) | calm, cautious, thanks | thanks, happy |
| Separate non-related issue, isn't it? | cautious, defensive | cautious, nervous |

Table 5.3: Apply LSTM model to predict Github comments sentiments. The SVO parser could extract from 976 Github comments successfully. We use the LSTM model to predict sentiments triggered by those sentences.

## 5.5 Conclusion

In this chapter, we show an application of using our proposed EPA induction method to predict individuals' sentiments. Different from previous work, we train an LSTM model instead of using the ACT, mainly because of the lack of experimental regression parameters in a given subculture. We show the average error rate of training LSTM models is between 0.55 and 0.60. Moreover, we can predict discrete sentiments with accuracy over

93%. We further test the models on a Github comments dataset. Even though the accuracy is not very satisfying, we believe it could be a promising direction to go. There are mainly three reasons that hurt the prediction performance. Firstly, Github sentences lack enough context compared to news headlines. Secondly, there are many abbreviations and terminologies used in Github community, and our model cannot distinguish polysemy in fine-grained level. Thirdly, the data that we use is not well-structured. Moreover, we find the Stanford Parser fail to extract more than a half of the collected Github comments. Therefore, we can consider ask human to annotate SVO identities to improve the data quality. Also, we can set up rules to remove meaningless sentences.

We also want to compare individuals' sentiments triggered by sentences between general culture and Github community. We believe this could provide evidence of affect divergence across culture and subculture. For example, do people feel the same when someone says encouraging things? Also, do people express negative sentiments in the same way even if in different environment settings? We could explore more in this direction with carefully constructed sentence lists in the future.

# Chapter 6

# Conclusion

Human sentiments are complicated states of mind deriving from experiences, circumstances, behaviors and so on. In recent years, sentiment analysis has received much attention in many fields, as it plays a significant role in a variety of real-world problems. Most researchers treat sentiments as binary scores or distinct labels. However, we find multi-dimensional space to be a comprehensive and universal representation of human sentiments. Affect control theory models sentiments in Evaluation-Potency-Activity space, which provides a practical method to do identity, behavior and sentiment reasoning and analysis. Most of the proposed machine learning approaches in the sentiment analysis area rely heavily on human annotations. Affect control theory researchers spend a long time collecting human's EPA ratings for thousands of concepts in several regions and countries. They demonstrate that the semantic differentials of a three-number profile indicate affective meanings related to a particular culture. Therefore, the collected human annotations become unreliable if investigating in a subculture or online culture.

In this thesis, we propose a framework to solve the scalability issue. We show that our automatic affective meanings recognizer could capture subtle EPA profile changes by taking pre-trained word embeddings. We further show the possibility of predicting sentiments evoked from sentences in different cultural contexts using our expanded EPA lexicons. Finally, the experiment results support our hypothesis, that cultural divergence in affective dimensions have been embedded in distributed word representations, and some affect meanings could be much different across subcultural environments. Basically, our framework has three steps to obtain the induced affect lexicon for any given culture:

1. Collect a text corpus for the desired culture (subculture), community, region or country.

2. Train word embeddings using the collected corpus and align the high dimensional space to general culture space.

3. Input pre-trained word vectors to mapping models and obtain output affective ratings.

Comparing to recruiting a group of participants and collecting individual ratings for each concept, our method is easy to carry on and close to the ground truths. We test the average error rate of using human annotations, predictions using general word representations and predictions using specific corpus's word representations to evaluate the performance. The results show our method yields the highest accuracy. We conclude three reasons for the biases of our prediction:

1. Information loss when aligning word vector spaces.
   We build models to align high dimensional spaces using two methods: Singular Value Decomposition (SVD) and Multilayer Perceptron (MLP). MLP model yields the higher accuracy in general, however, there is at least 0.5 error rate in the best case. Therefore there is a certain percentage of unavoidable information loss when doing the space transformation.

2. Information loss when mapping from word vector space to affect space.
   We attempt to model the mapping function using three methods: Graph-based semi-supervised learning (GP), Supported Vector Regression (SVR) as well as the Neural Network (NN). In general, the neural network models get the highest accuracy with the limited training data. Again, there is unavoidable information loss when doing the mapping.

3. The nature of induction and approximation. Basically, our proposed method is trying to find the approximation of mapping function from word vectors to affective ratings under a given context. We use the general word vectors and human labeled affect meanings to measure the shifts. We doubt if the bias of such approximation could be eliminated.

## 6.1 Future Work

In this thesis, we only explore a little of using the induced EPA lexicon to predict sentiments triggered by sentence due to the lack of a comparable sentence dataset. A more comprehensive design and study could be further conducted to evaluate predicted affect meanings' performance on practical problems. Moreover, we can integrate our results into the THEMIS project and explore how sentimental difference plays a role in group cooperations.

Concerning the word vector space alignment method, there are more things could be done to clean the data. For example, comments like "fix bugs..." and "typo :(" could be filtered out as they provide little useful information. Users' comments under the same pull requests could be grouped as they share the same context.

With regard to the affect lexicon expansion method, there are several approaches can be applied to enhancing the quality of the generated lexicon. For example, phrases and multi-word terms could not be accepted by the current framework. We can think of some improvement methods to obtain their word vector positions and predict affect meanings.

In general, our proposed framework in this thesis provides a new direction to obtain affect meanings lexicon in sentiment analysis. Also, we contribute toward an enhanced understanding of the semantic differential, affect differences, and cultural divergence. There are multiple directions can be explored in the future, and many projects could be done using the proposed work.

# References

[1] Areej Alhothali and Jesse Hoey. Good news or bad news: using affect control theory to analyze readers' reaction towards news articles. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1548–1558, 2015.

[2] Areej Alhothali and Jesse Hoey. Semi-supervised affective meaning lexicon expansion using semantic and distributed word representations. *arXiv preprint arXiv:1703.09825*, 2017.

[3] Cecilia Ovesdotter Alm and Richard Sproat. Emotional sequencing and development in fairy tales. In *International Conference on Affective Computing and Intelligent Interaction*, pages 668–674. Springer, 2005.

[4] Scott Ambler. *Agile modeling: effective practices for extreme programming and the unified process.* John Wiley & Sons, 2002.

[5] Silvio Amir, Ramón Astudillo, Wang Ling, Bruno Martins, Mario J Silva, and Isabel Trancoso. Inesc-id: A regression model for large scale twitter sentiment lexicon induction. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 613–618, 2015.

[6] Amir Bakarov. A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536*, 2018.

[7] Debasish Basak, Srimanta Pal, and Dipak Chandra Patranabis. Support vector regression. *Neural Information Processing-Letters and Reviews*, 11(10):203–224, 2007.

[8] Plaban Kumar Bhowmick. Reader perspective emotion analysis in text through ensemble based multi-label classification framework. *Computer and Information Science*, 2(4):64, 2009.

[9] Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A Reis, and Jeff Reynar. Building a sentiment summarizer for local service reviews. In *WWW workshop on NLP in the information explosion era*, volume 14, pages 339–348, 2008.

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

[11] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[12] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[13] Gang Chen, Yangqiu Song, Fei Wang, and Changshui Zhang. Semi-supervised multi-label learning by solving a sylvester equation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 410–419. SIAM, 2008.

[14] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*, 2010.

[15] Wikimedia Commons. File:example of unlabeled data in semisupervised learning.png — wikimedia commons, the free media repository, 2016. [Online; accessed 4-October-2018].

[16] Samuel I Daitch, Jonathan A Kelner, and Daniel A Spielman. Fitting a graph to vector data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 201–208. ACM, 2009.

[17] Ann Devitt and Khurshid Ahmad. Sentiment polarity identification in financial news: A cohesion-based approach. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 984–991, 2007.

[18] Paul Ekman. Are there basic emotions? 1992.

[19] Siamak Faridani. Using canonical correlation analysis for generalized sentiment analysis, product recommendation and search. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 355–358. ACM, 2011.

[20] Ethan Fast, Binbin Chen, and Michael S Bernstein. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4647–4657. ACM, 2016.

[21] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

[22] Barbara L Fredrickson. The role of positive emotions in positive psychology: The broaden-and-build theory of positive emotions. *American psychologist*, 56(3):218, 2001.

[23] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[24] Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644, 2018.

[25] Aparna Garimella, Rada Mihalcea, and James Pennebaker. Identifying cross-cultural differences in word usage. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 674–683, 2016.

[26] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[27] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[28] Jamie Guillory, Jason Spiegel, Molly Drislane, Benjamin Weiss, Walter Donner, and Jeffrey Hancock. Upset now?: emotion contagion in distributed groups. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 745–748. ACM, 2011.

[29] Emitza Guzman, David Azócar, and Yang Li. Sentiment analysis of commit comments in github: an empirical study. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 352–355. ACM, 2014.

[30] Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. Learning bilingual lexicons from monolingual corpora. *Proceedings of ACL-08: Hlt*, pages 771–779, 2008.

[31] William L Hamilton, Jure Leskovec, and Dan Jurafsky. Cultural shift or linguistic drift? comparing two computational measures of semantic change. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 2116. NIH Public Access, 2016.

[32] William L Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096*, 2016.

[33] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

[34] Vasileios Hatzivassiloglou and Kathleen R McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*, pages 174–181. Association for Computational Linguistics, 1997.

[35] Carleen Hawn. Take two aspirin and tweet me in the morning: how twitter, facebook, and other social media are reshaping health care. *Health affairs*, 28(2):361–368, 2009.

[36] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[37] David R Heise. Affect control theory: Concepts and model. *Journal of Mathematical Sociology*, 13(1-2):1–33, 1987.

[38] David R Heise. Understanding social interaction with affect control theory. *New directions in contemporary sociological theory*, pages 17–40, 2002.

[39] David R Heise. *Surveying cultures: Discovering shared conceptions and sentiments.* John Wiley & Sons, 2010.

[40] David R Heise. Cultural variations in sentiments, 2014.

[41] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[42] Stanisław Jastrzebski, Damian Leśniak, and Wojciech Marian Czarnecki. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *arXiv preprint arXiv:1702.02170*, 2017.

[43] Kenneth Joseph and Kathleen M Carley. Relating semantic similarity and semantic association to how humans label other people. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 1–10, 2016.

[44] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[45] Jaap Kamps, Maarten Marx, Robert J Mokken, Maarten De Rijke, et al. Using wordnet to measure semantic orientations of adjectives. In *LREC*, volume 4, pages 1115–1118. Citeseer, 2004.

[46] Philipp Koehn and Kevin Knight. Estimating word translation probabilities from unrelated monolingual corpora using the em algorithm. In *AAAI/IAAI*, pages 711–715, 2000.

[47] Philipp Koehn and Kevin Knight. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition-Volume 9*, pages 9–16. Association for Computational Linguistics, 2002.

[48] Zornitsa Kozareva, Borja Navarro, Sonia Vázquez, and Andrés Montoyo. Ua-zbsa: a headline emotion classification through web information. In *Proceedings of the 4th international workshop on semantic evaluations*, pages 334–337. Association for Computational Linguistics, 2007.

[49] Austin C Kozlowski, Matt Taddy, and James A Evans. The geometry of culture: Analyzing meaning through word embeddings. *arXiv preprint arXiv:1803.09288*, 2018.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[51] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635. International World Wide Web Conferences Steering Committee, 2015.

[52] Alberto Lavelli, Fabrizio Sebastiani, and Roberto Zanoli. Distributional term representations: an experimental comparison. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 615–624. ACM, 2004.

[53] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[54] Minglei Li, Qin Lu, Yunfei Long, and Lin Gui. Affective state prediction of contextualized concepts. In *IJCAI 2017 Workshop on Artificial Intelligence in Affective Computing*, pages 45–57, 2017.

[55] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, Michele Lanza, and Rocco Oliveto. Sentiment analysis for software engineering: How far can we go? In *Proceedings of 40th International Conference on Software Engineering*, 2018.

[56] Kevin Hsin-Yih Lin, Changhua Yang, and Hsin-Hsi Chen. Emotion classification of online news articles from the reader's perspective. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 220–226. IEEE Computer Society, 2008.

[57] Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer, 2012.

[58] Wei Liu, Junfeng He, and Shih-Fu Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 679–686, 2010.

[59] Batja Mesquita and Nico H Frijda. Cultural variations in emotions: a review. *Psychological bulletin*, 112(2):179, 1992.

[60] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[61] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[62] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[63] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[64] Tom M Mitchell. Artificial neural networks. *Machine learning*, 45:81–127, 1997.

[65] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. Curating github for engineered software projects. *Empirical Software Engineering*, 22(6):3219–3253, 2017.

[66] Alessandro Murgia, Parastou Tourani, Bram Adams, and Marco Ortu. Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In *Proceedings of the 11th working conference on mining software repositories*, pages 262–271. ACM, 2014.

[67] Nicolas Nicolov, William Allen Tuohig, and Richard Hansen Wolniewicz. Automatic sentiment analysis of surveys, December 10 2009. US Patent App. 12/481,398.

[68] Charles E Osgood. The nature and measurement of meaning. *Psychological bulletin*, 49(3):197, 1952.

[69] Charles E Osgood. Dimensionality of the semantic space for communication via facial expressions. *Scandinavian journal of Psychology*, 7(1):1–30, 1966.

[70] Charles Egerton Osgood, William H May, Murray Samuel Miron, and Murray S Miron. *Cross-cultural universals of affective meaning*, volume 1. University of Illinois Press, 1975.

[71] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[72] W Gerrod Parrott. *Emotions in social psychology: Essential readings.* Psychology Press, 2001.

[73] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. Security and emotion: sentiment analysis of security discussions on github. In *Proceedings of the 11th working conference on mining software repositories*, pages 348–351. ACM, 2014.

[74] Delip Rao and Deepak Ravichandran. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for*

*Computational Linguistics*, pages 675–682. Association for Computational Linguistics, 2009.

[75] Deepak Rishi. Affective sentiment and emotional analysis of pull request comments on github. Master's thesis, University of Waterloo, 2017.

[76] Deepak Rishi, Jesse Hoey, Mei Naggappan, Kimberly B Rogers, and Tobias Schroder. Emotion and interaction processes in a collaborative online network.

[77] Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. Ultradense word embeddings by orthogonal transformation. *arXiv preprint arXiv:1602.07572*, 2016.

[78] Masoud Jalili Sabet, Heshaam Faili, and Gholamreza Haffari. Improving word alignment of rare words with word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3209–3215, 2016.

[79] Lawrence K Saul and Sam T Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of machine learning research*, 4(Jun):119–155, 2003.

[80] Robert J Schalkoff. *Artificial neural networks*, volume 1. McGraw-Hill New York, 1997.

[81] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[82] Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*, 2017.

[83] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[84] Philip J Stone, Dexter C Dunphy, and Marshall S Smith. The general inquirer: A computer approach to content analysis. 1966.

[85] Md Arafat Sultan, Steven Bethard, and Tamara Sumner. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230, 2014.

[86] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction.* MIT press, 1998.

[87] James Chi-shun Tsiao, David Yinkai Chao, and Peter P Tong. Natural-language voice-activated personal assistant, May 8 2007. US Patent 7,216,080.

[88] Peter D Turney, Michael L Littman, Jeffrey Bigham, and Victor Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. *arXiv preprint cs/0309035*, 2003.

[89] Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 777–785. Association for Computational Linguistics, 2010.

[90] Hanna M Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM, 2006.

[91] Jun Wang, Tony Jebara, and Shih-Fu Chang. Graph transduction via alternating minimization. In *Proceedings of the 25th international conference on Machine learning*, pages 1144–1151. ACM, 2008.

[92] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1591–1601, 2014.

[93] Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 45(4):1191–1207, 2013.

[94] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics, 2005.

[95] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 267–272, 2015.

[96] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4, 2006.

[97] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, language technologies institute, school of computer science Pittsburgh, PA, 2005.

[98] Alexey Zobnin. Rotations and interpretability of word embeddings: the case of the russian language. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 116–128. Springer, 2017.

# Glossary

**alignment-model** A statistical function that do word embedding hyperspace transformation. 12

**General-EPA** Affective ratings for the general culture generated by our proposed method. xi, 42, 51–54

**Github-comments** A dataset consisting of pull requests, issue comments and commit messages from 4,124 Github repositories. 17, 18, 81

**Github-EPA** Affective ratings for the Github subculture generated by our proposed method. xi, xiv, 30, 42, 49, 53–56, 63

**Github-wv** Word embeddings trained on Github-comments without alignment. xi, xiv, 17, 18, 20–22, 26, 38, 49, 50, 53

**Github-wv-aligned** Word embeddings trained on Github-comments and aligned to the Google-wv hyperspace. x, 18, 19, 26, 27, 30, 42

**Google-wv** Word embeddings trained on Google News corpus [1]. x, xi, 17–19, 26, 27, 30, 38, 42, 49, 50, 52, 81

**mapping-model** A machine learning model that can measure affective meanings from word embeddings. 30

**Themis-EPA** A dataset collected in THEMIS.COG project [2] consisting of affective ratings in evaluation, potency, and activity dimensions (EPA) for 587 concepts. xi, xiv, 30, 40, 42, 49–51, 53–55

[1]https://code.google.com/archive/p/word2vec/
[2]https://themis-cog.github.io/

**Warriner-EPA** A dataset collected by Warriner et al. consisting of affective norms of valuence, arousal, and dominance (VAD) for 13,915 concepts [93]. xi, 19, 30, 38, 40, 42, 49–54, 56, 63