ADVANCE Labs - Debiasing Word Embeddings Lab

The development of this document is partially funded by the National Science Foundation's Security and Trustworthy Cyberspace Education, (SaTC-EDU) program under Award No. 2114920.

1 Lab Overview

In this lab, you will be introduced to word embeddings which are commonly used in training AI/ML models to detect cyberbullying. In Lab 1 you were introduced to the steps involved in training a ML model for cyberbullying detection. One of those steps involved loading a pre-trained word embedding. You were briefly introduced to word embeddings in the hyperparameter tunning tasks of Lab 1 were you changed the dimensions of the embeddings among others. In this lab, you will learn that word embeddings contain biases. When biased word embeddings are used in training ML models, the model will propagate such biases and may lead to undesired outcomes. You will also learn that word embedding contain information that can be used to reduce bias.

In this lab, students will be given a starter code. Their task is to follow the instructions in the Jupyter notebook, complete simple coding challenges, test implemented code and write a detailed report.

2 Lab Environment

This ADVANCE lab has been designed as a Jupyter notebook. ADVANCE labs have been tested on the Google Colab platform. You should use Google Colab since it has nearly all software packages preinstalled, is free to use, and provides free GPUs. You can also download the Jupyter notebook from the lab website and run it on your machine, in which case you will need to install the software packages yourself (you can find the list of packages on the ADVANCE website). However, most of the ADVANCE labs can be conducted on the cloud, and you can follow our instructions to create the lab environment on the cloud.

3 Lab Tasks

3.1 Getting Familiar with Jupyter Notebook

The main objective of this lab is to learn how to perform adversarial attack on models trained to detect cyberbullying. Before proceeding to that, let us get familiar with the Jupyter notebook environment.

Jupyter notebooks have a Text area and a Code cell. The Text area is where you'll find instructions and notes about the lab tasks and the Code cell is where code written. You won't be interacting with code in this lab, however, you will be executing code.

In this lab, you will be executing code cells and selecting values from a drop-down menu in code cells. There are two tasks to be completed indicated. Two tasks are to be completed in the notebook provided, indicated by "Task #" where # is the task number. You must complete these two tasks, plus the discussion at the end of this document, to complete this lab successfully. To execute a code cell, click the play button on the left of the cell. An example of a code cell and a play button is shown in Figure 1. If you mistakenly click on the "Show code" button, don't worry about it. It only expands the code cell and makes the code visible.

Figure 1: Example of a code cell and play button.

There are code cells that contain a text box, the text box will appear when you run click the play button on a cell. When the text box becomes visible enter a text and hit enter/return on your keyboard to submit your input, upon submission you will get an output. Figure 2 shows an example of such a text box.

Figure 2: Example of a text box

Get started. After going through this manual, go through the notebook, follow the instructions and run the code cells.

3.2 Debiasing Word Embeddings

You will debias gender bias contained in word embeddings in the lab. We will make making use of a pre-trained GloVe word embeddings. To debias word embeddings, a gender direction is determined, gender neutral words neutralized in the gender direction, then gender specific words are equalized. The ultimate goal is to debias gender specific words so that they are equidistant to gender neutral words. You can access the lab by clicking here.

3.2.1 Word Representations

Word representation aims to represent words as vectors. A simple way of representing words is through one-hot representation. If we have a vocabulary of words in a list $V = [a, apple, man, orange, woman]$ and $|V| = 5$ is the size of the vocabulary. Then each of the words in the vocabulary can be represented as a one-hot vector as follows:

$$\vec{O}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \vec{O}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \vec{O}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \vec{O}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

where $\vec{O}_i$ represents the one-hot vector (a list of numbers) of word i at index i (starting at index 0) in our list of vocabulary. In one-hot encoding, a 1 is placed in the list at the position that corresponds to the position of a word in the vocabulary list. For instance, $apple$ is at position 1 in the vocabulary list if we start counting from 0, the one-hot list of $apple$ will have a 1 at position 1 and zero everywhere else as shown in

$\vec{O_1}$. The disadvantage of one-hot representation is that it does not capture the level of similarity between words. The distance of $\vec{O_i}$ is the same as $\vec{O_{i+1}}$. This is not the case for word embedding representation. **A vector is a list of numbers**.

3.2.2 Word Embedding

A word embedding is a representation that also represents words as vectors. Words are represented as a d-dimensional vector in high dimensional space as shown in Fig 3, d can be any number but most commonly 50, 100 or 300. The vectors are meaningful, words with similar semantic meaning tend to have vectors that are close to each other. A word embedding is trained on word co-occurrence in text corpora using a neural network [Mik+13]. During training an embedding matrix E is learned as shown in Fig 4. You can think of it as a look up table.

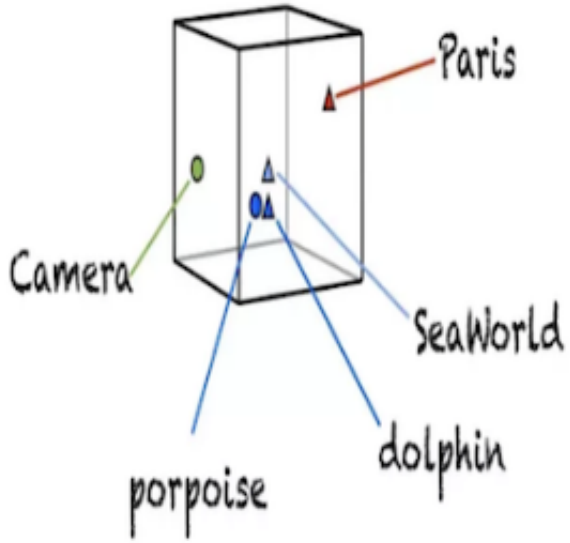

Figure 3: Word embeddings in high dimensional space

From Fig 4, columns represent embedding of words. The column numbers of orange are its embedding vector. Ploting the high dimensional embeddings using t-NSE would show that apple and orange are similar by being in the same cluster as they are both fruits. Embeddings of man and woman will be in the same cluster as they word gender.

3.2.3 Properties of Word Embedding

In addition to words with similar meaning being close to each other, word embeddings have been show to represent relationships between words [Bol+16]. For example, given an analogy "man is to king as woman is to x" (represented as $man : king :: woman : x$), performing a simple arithmetic of the embedding vectors $\vec{man}$, $\vec{king}$, and $\vec{woman}$ finds that $x = queen$ because $\vec{man} - \vec{woman} \approx \vec{king} - \vec{queen}$. To find $\vec{x}$, we need to find the word that its similarity is close to $\vec{king} - \vec{man} + \vec{woman}$. We use $\vec{word}$ to indicate the vector of "word".

apple	man	orange	woman
-1	1	-1.1	1.1
0.02	0.5	0.03	0.6
0.04	0.7	0.05	0.8
.	.	.	.
.	.	.	.
.	.	.	.
0.07	-0.02	0.06	-0.01

50

$|V| = 4$

Figure 4: Embedding matrix E. The first row is for clarity.

3.2.4 Cosine Similarity

Similarity between two word vectors u and v can be measured as follows:

$$CosineSimilarity(u, v) = \frac{u \cdot v}{||u||_2||v||_2} = cos(\theta) \tag{1}$$

$$||x||_2 = \sqrt{x_i^2 + ... + x_n^2}$$

where $u \cdot v$ is the inner product of the two vectors, $||u||_2$ is the length of the vector u, θ is the angle between u and v. The value is usually between -1 and 1. The similarity value is dependent on the angle between u and v. A cosine similarity value of 1 means that u and v are very similar. If the cosine similarity value is small, it means u and v are dissimilar. A value of -1 means exactly opposite.

3.2.5 Pre-trained Word Embeddings

Pre-trained word embeddings are word embeddings that are already trained using billions of text from a corpus. The pre-trained embedding used in this lab is the 50-dimensional GloVe embeddings.

3.2.6 Using Word Embeddings

We used pre-trained word embedding in Lab one to train our cyberharassment model as shown in Fig 5. What we did in lab one is known as transfer learning. In transfer learning you transfer knowledge from one task to a new task when you don't have sufficient traning data for the new task. The word embedding we used in lab 1 has been trained on using a large text corpus.

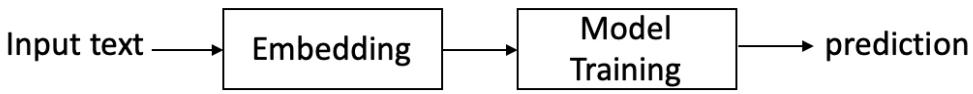

Figure 5: Using pre-trained word embedding

3.2.7 Debiasing Algorithm

Word embeddings can propagate gender, ethnicity, sexual orientation and other biases contained in the text used to train the model under training. For example, if we used a biased word embedding in training our cyberharassment model in lab 1, the trained model may be biased towards certain groups and may produce undesired outputs. The trained system can offensively answer "x = homemaker" to the analogy "man is to computer programmer as woman is to x". Because $\vec{man} - \vec{woman} \approx \vec{computerprogrammer} - \vec{homemaker}$. The same system can also answer $nurse$ to the analogy "father is to $doctor$ as $mother$ is to x".

To debias word embeddings, a distinction between gender specific and gender neutral words needs to be made. Gender specific words such $brother$, $sister$, $businesswoman$, $businessman$, etc are words that are associated to gender dictionary definition. Gender neutral words such as $flight\ attendant$ and $shoe$ are words that are not gender specific, they are the compliments of gender specific words.

The gender specific words are used to learn a gender direction or subspace in the embedding. Then the debiasing algorithm removes the bias only from the gender neutral words.

The first step in addressing bias in word embeddings is to identify a bias direction that captures bias. This direction can be easily found by computing the vector difference between gender specific words. For example, the bias direction can be found by: $g = \vec{she} - \vec{he}$ or $g = \vec{her} - \vec{his}$, $g = \vec{she} - \vec{he}$ etc.

The second step is neutralize, neutralize ensures that gender neutral words are zero in the gender direction. Neutralize will remove bias from neutral words by reducing some values from the vector (i.e number list) of the neutral word such as "babysit". This is the projection of the word onto the non-bias direction.

The last step is to Equalize, equalize ensures that any neutral word is equidistant (equal distance) to all words in an equality set (pair of gender specific words) such as $\{grandmother, grandfather\}$ and $\{guy, gal\}$. For each set of words, equalize will equate each word vector to simply their average, then adjusts the vectors so that they are of unit length.

4 Tasks

4.1 Task 1: Similarity

1) Which of the outputs are very similar or dissimilar? Give a reason for each of your answer. 2) What can you observe about the last output?

4.2 Task 2: Word analogy

1) What do you observe? Were there any cultural or gender stereotypes? List them. 2) Does the algorithm always give the right answer? List the incorrect analogies and what the correct analogy is if any.

4.3 Task 3: Occupational stereotype

1) Does the GloVe word embeddings propagate bias? why? 2) From the list associated with she, list those that reflect gender stereotype. 3) Compare your list from 2 to the occupations closest to he. What are your conclusions? Exclude businesswoman from your list.

4.4 Task 4a: Similarity between gender direction and names

Run the indicated code cell in the lab notebook to computes the similarity between the gender embedding and the embedding vectors of male and female names. What can you observe about the names?

4.5 Task 4b: Direct and indirect bias

Quantify direct and indirect biases between words and the gender embedding by running the indicated cell in the lab notebook. What is your observation?

4.6 Task 4c: Neutralize gender neutral words

Run the indicated cell in the notebook to neutralize the gender neutral word "babysit". What is your observation?

4.7 Task 5: Equalize

Looking at the equalization output after running the indicated cell, what can you observe?.

5 Submission Instructions

You need to submit a detailed lab report, with screenshots, to describe what you have observed. You also need to explain the observations that are interesting or surprising. Please also list important code snippets.

References

[Mik+13] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[Bol+16] Tolga Bolukbasi et al. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings". In: *Advances in neural information processing systems* 29 (2016).