ADVANCE Labs - Debiasing Word Embeddings Lab

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 Prerequisite - Linear Algebra

This lab requires some background knowledge of linear algebra. Re-familiarize yourself with vectors, vector addition and subtraction, vector multiplication by scalar, and unit vectors, dot product and vector length, angle between vectors, vector projection, and eigenvalues and eigenvectors.

4 Lab Tasks

4.1 Getting Familiar with Jupyter Notebook

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

Jupyter notebooks have a Text area and a Code area. The Text area is where you'll find instructions and notes about the lab tasks. The Code area is where you'll write and run code. Packages are installed using `pip` and need to be preceded with a `!` symbol.

You will complete some simple coding challenges and tasks in this lab. You will be asked to complete and execute sections of a code. After execution, you are expected to add a screenshot of your output in your report. In the notebook provided, eight tasks are to be completed, indicated by "Task #" where # is the task number. You must complete these eight tasks plus the discussion at the end of this document to complete this lab successfully. Following each task is a code cell with some part of the code replaced with ***None*** statements. You are to understand the code and replace all the ***None*** statements within the designated block with the correct statement unless stated otherwise. The designated block starts with a comment (Start code here) and ends with a comment (End code here). In the designated block, you are only required to replace the ***None*** statements with the correct statements (variable(s) or formulas). For example, in the designated block in the code cell below, you must replace ***None*** with the correct statement.

```
# Start code here #
...
total_number_of_examples = None
...
# End code here #
```

Get started. Run the cell under "Required modules" to load the required frameworks for this lab.

4.2 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.

4.2.1 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 $V = [a, apple, man, orange, woman]$ 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 of word i at index i (starting at index 0) in our list of vocabulary. The disadvantage of one-hot representation is that it does not capture the level of similarity between words. The euclidean distance of $\vec{O}_i$ is the same as $\vec{O}_{i+1}$ and their inner product is zero. This is not the case for word embedding representation.

4.2.2 Word Embedding

A word embedding is a representation that also represents words as vectors. Words are represented as a d-dimensional vector $\vec{w} \in \mathbb{R}^d$ in high dimensional space as shown in Fig 1. 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 2. You can think of it as a look up table.

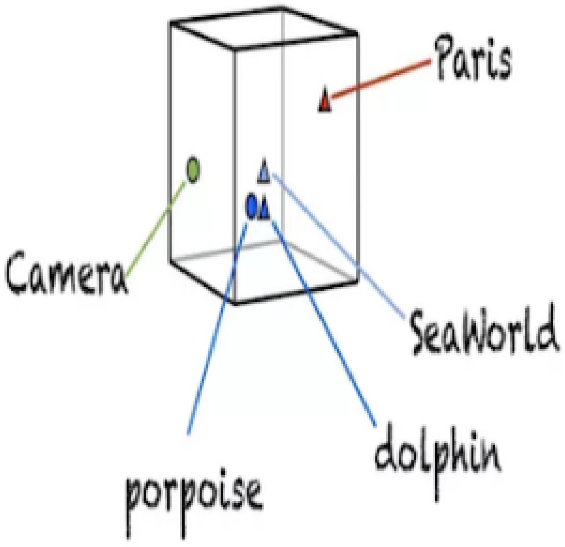
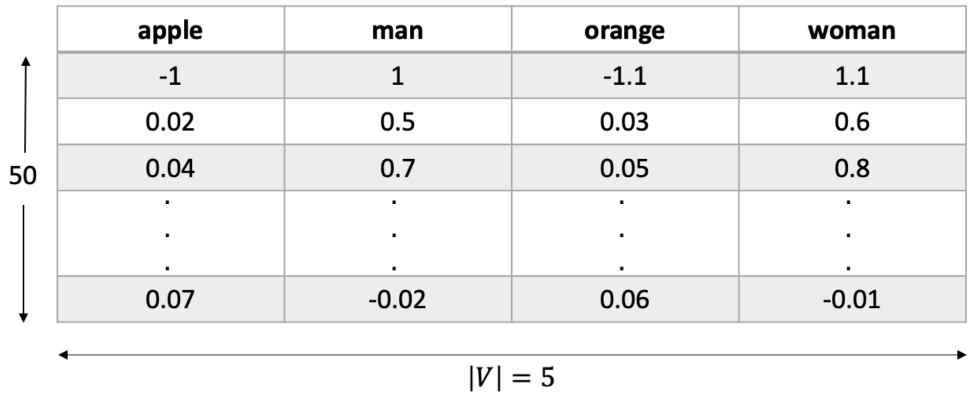


Figure 1: Word embeddings in high dimensional space

From Fig 2, columns represent embedding of words. The embedding of orange can be extracted by computing $E \cdot \vec{O}_2$. However, this computation is inefficient. Modern machine learning frameworks such as PyTorch have an Embedding layer that efficiently extracts the embedding of words for us. 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.



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

$|V| = 5$

Figure 2: Embedding matrix E. The first row is for clarity.

4.2.3 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 $\vec{king} - \vec{man} + \vec{woman}$.

4.2.4 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 dot product of the two vectors, $||u||_2$ is the Euclidean norm or length of the vector u, θ is the angle between u and v. Its value ranges 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 indicates exactly opposite. Fig 3 A shows that u and v are similar if for example u is England and v is London, Fig 3 B show that u and v are not similar if for example u is dog and v is plane , and Fig 3 C show that the two vectors are in opposite direction (example u is $\vec{spain} - \vec{madrid}$ and v is $\vec{lisbon} - \vec{portugal}$) but are similar.

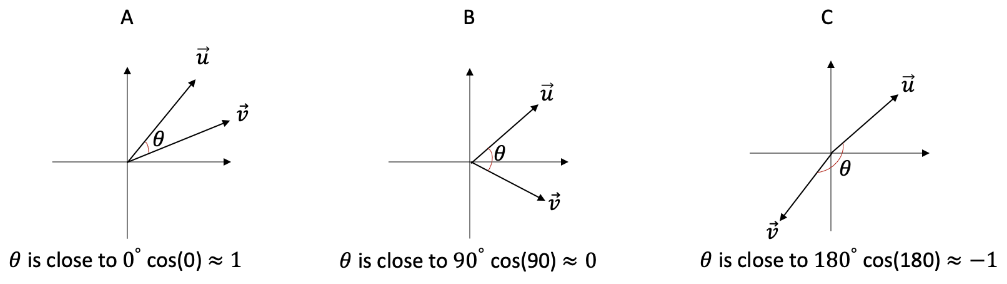

A
B
C
$\vec{u}$
$\vec{v}$
θ
θ is close to 0° cos(0) ≈ 1
θ is close to 90° cos(90) ≈ 0
θ is close to 180° cos(180) ≈ −1

Figure 3: The measure of the similarity between two vectors $\vec{u}$ and $\vec{v}$ is the cosine of the angle between them

4.2.5 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.

4.2.6 Using Word Embeddings

We used pre-trained word embedding in Lab one to train our cyberharassment model as shown in Fig 4. 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.

4.2.7 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

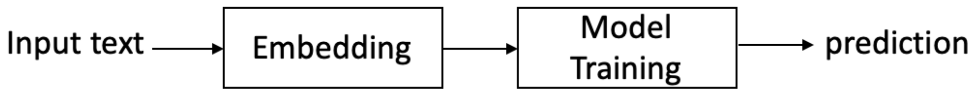


Figure 4: Using pre-trained word embedding

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 (or subspace) 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. A more accurate method is to use compute the principal components of 10 gender pair difference and take the top principal component as the gender direction g.

The second step is neutralize, neutralize ensures that gender neutral words are zero in the gender direction. So, in our 50-dimensional embeddings example, the 50-dimensional space can be split into 49-dimensional non-bias direction ($\vec{g}_{\perp}$) and bias direction g. The non-bias direction is orthogonal (at $90\,\deg$) to the bias direction. Neutralize will remove bias from neutral words by reducing some values from the components of the neutral word such as "babysit". This is the projection of the word onto the non-bias direction. The following equation neutralizes $\vec{v}$:

$$v^{bias\_component} = \frac{e \cdot g}{||g||_2^2} * g \tag{2}$$

$$v^{debiased} = v - v^{bias\_component} \tag{3}$$

Equation 3 means the projection of v onto the orthogonal subspace.

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. Given two gender specific word pairs w_1 and w_2 to debias, and their embeddings e_{w_1} and e_{w_2}, equalization can be achieved with the following equations:

$$\mu = \frac{e_{w_1} + e_{w_2}}{2} \tag{4}$$

$$\mu_B = \frac{\mu \cdot g}{||g||_2^2} * g \tag{5}$$

$$v = \mu - \mu_B \tag{6}$$

$$e_{w_1B} = \frac{e_{w_1} \cdot g}{||g||_2^2} * g \tag{7}$$

$$e_{w_2B} = \frac{e_{w_2} \cdot g}{||g||_2^2} * g \tag{8}$$

$$e_{w_1B}^{new} = \sqrt{|1 - ||v||_2^2|} * \frac{e_{w_1B} - \mu_B}{||(e_{w_1} - v) - \mu_B||_2} \tag{9}$$

$$e_{w_2B}^{new} = \sqrt{|1 - ||v||_2^2|} * \frac{e_{w_2B} - \mu_B}{||(e_{w_2} - v) - \mu_B||_2} \tag{10}$$

$$e_1 = v + e_{w_1B}^{new} \tag{11}$$

$$e_2 = v + e_{w_2B}^{new} \tag{12}$$

5 Tasks

5.1 Task 1a: Implement cosine similarity

Implement the cosine similarity using equation 1.

5.2 Task 1b: Try your own input

Test your cosine similarity by running the indicated cells and using your own words.

5.3 Task 2a: Implement word analogy

Implement the answer_analogy() function as indicated in the notebook.

5.4 Task 2b: Test word analogy and report your observation

After testing you word analogy implementation, what are your observations? What do you observe about the last two outputs?.

5.5 Task 2c: Try your own analogies

Find two word analogies that works and one that doesn't. Choose your words, complete the code and report your inputs and outputs.

5.6 Task 3a: Implement occupation stereotypes

Complete the get_occupation_stereotypes() function as indicated in the notebook.

5.7 Task 3b: Observation on occupation stereotypes

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.

5.8 Task 4a: Gender embedding and male and female names

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

5.9 Task 4b: Quantify direct and indirect bias

Quantify bias between words and the gender embedding by running the cell indicated in the notebook. What is your observation?

5.10 Task 4c: Implement neutralize

Implement neutralization using equations 2 - 3.

5.11 Task 4d: Test neutralize

Test your neutralization implementation. What do you observe?.

5.12 Task 5a: Implement equalization

Implement equalization using equations 4 -12

5.13 Task 5b: Test equalization

Test your equalization implementation.

5.14 Task 5c: Equalization observation

Looking at the output of your equalization implementation test above, what can you observe?.

6 Submission Instructions

You need to submit a detailed lab report, with screenshots, to describe what you have done and 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).