Word Embedding Fairness Evaluation

With word embeddings being such a crucial component of NLP, the reported social biases resulting from the training corpora could limit their application. The framework introduced here intends to measure the fairness in word embeddings to better understand these potential biases.

comments

By Pablo Badilla and Felipe Bravo-Marquez.

About WEFE

Word embeddings are dense vector representations of words trained from document corpora. They have become a core component of natural language processing (NLP) downstream systems because of their ability to efficiently capture semantic and syntactic relationships between words. A widely reported shortcoming of word embeddings is that they are prone to inherit stereotypical social biases exhibited in the corpora on which they are trained.

The problem of how to quantify the mentioned biases is currently an active area of research, and several different *fairness metrics* have been proposed in the literature in the past few years.

Although all metrics have a similar objective, the relationship between them is by no means clear. Two issues that prevent a clean comparison is that they operate with different inputs (pairs of words, sets of words, multiple sets of words, and so on) and that their outputs are incompatible with each other (reals, positive numbers, range, etc.). This leads to a lack of consistency between them, which causes several problems when trying to compare and validate their results.

We propose the **Word Embedding Fairness Evaluation** (WEFE) as a framework for measuring fairness in word embeddings, and we released its implementation as an open-source library.

Framework

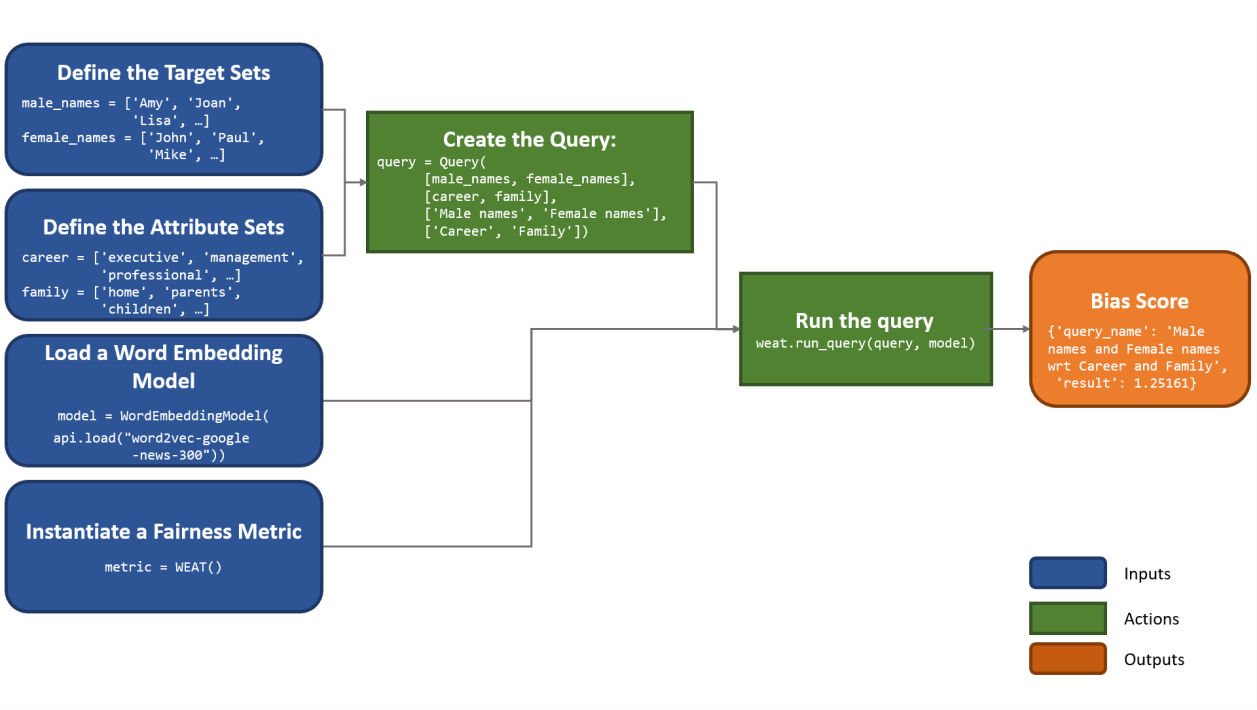
We propose an abstract view of a *fairness metric* as a function that receives *queries* as input, with each query formed by a *target* and *attribute* words. The *target words* describe the social groups in which fairness is intended to be measured (e.g., women, white people, Muslims), and the *attribute words* describe traits or attitudes by which a bias towards one of the social groups may be exhibited (e.g., pleasant vs. unpleasant terms). For more details on the framework, you can read our recently accepted paper in *IJCAI-PRICAI* [1].

WEFE implements the following metrics:

- Word Embedding Association Test (WEAT)
- Relative Norm Distance (RND)
- Relative Negative Sentiment Bias (RNSB)
- Mean Average Cosine (MAC)

Standard usage pattern of WEFE

The standard process for measuring bias using WEFE is shown in the following diagram:



Installation

There are two different ways to install WEFE:

```
pip install wefe
```

or

```
conda install -c pbadilla wefe
```

Running a Query

In the following code, we measure the *gender bias* of *word2vec* using:

- A query that studies the relationship between male names and career-related words, and female names and family-related words, which we call *"Male names and Female names wrt Career and Family"*. The words used in the query are detailed in the code.
- The Word Embedding Association Test (WEAT) metric. Proposed by Caliskan et al. 2017, WEAT receives two sets $\mathbf{T_1}$ and $\mathbf{T_2}$ of target words, and two sets $\mathbf{A_1}$ and $\mathbf{A_2}$ of attribute

words. Thus, it always expects a query of the form $\mathbf{Q} = (\mathbf{T_1}, \mathbf{T_2}, \mathbf{A_1}, \mathbf{A_2})$. Its objective is to quantify the strength of association of both pair of sets through a permutation test.

Given a word embedding $\mathbf{w}$, WEAT defines first the measure $\mathbf{d}(\mathbf{w}, \mathbf{A_1}, \mathbf{A_2}) = \left(\text{mean}_{\mathbf{x} \in \mathbf{A_1}} \cos(\mathbf{w}, \mathbf{x})\right) - \left(\text{mean}_{\mathbf{x} \in \mathbf{A_2}} \cos(\mathbf{w}, \mathbf{x})\right)$ where $\cos(\mathbf{w}, \mathbf{x})$ is the cosine similarity of the word embedding vectors.

Then for a query $\mathbf{Q} = (\mathbf{T_1}, \mathbf{T_2}, \mathbf{A_1}, \mathbf{A_2})$ the WEAT metric is defined over the embeddings of the query word sets as:

$$\mathbf{F}_{\text{WEAT}}(\mathbf{M}, \mathbf{Q}) = \sum_{\mathbf{w} \in \mathbf{T_1}} \mathbf{d}(\mathbf{w}, \mathbf{A_1}, \mathbf{A_2}) - \sum_{\mathbf{w} \in \mathbf{T_2}} \mathbf{d}(\mathbf{w}, \mathbf{A_1}, \mathbf{A_2})$$

The idea is that the more positive the value given by $\mathbf{F}_{\text{WEAT}}$, the more the target $\mathbf{T_1}$ will be related to attribute $\mathbf{A_1}$ and target $\mathbf{T_2}$ to attribute $\mathbf{A_2}$. On the other hand, the more negative the value, the more target $\mathbf{T_1}$ will be related to attribute $\mathbf{A_2}$ and target $\mathbf{T_2}$ to attribute $\mathbf{A_1}$. Commonly these values are between $\pm \mathbf{0.5}$ and $\pm \mathbf{2.0}$. The ideal score is $\mathbf{0}$.

1. We first load a word embedding model using the gensim API.

```
from wefe import WordEmbeddingModel, Query, WEAT
import gensim.downloader as api

word2vec_model = WordEmbeddingModel(api.load('word2vec-google-news-300'),
                                    'word2vec-google-news-300')
```

2. Then, we create the Query object using the target words (*Male names* and *Female names*) and two attribute words sets (*Career* and *Family* terms).

```
# target sets (sets of popular names in the US)
male_names = ['John', 'Paul', 'Mike', 'Kevin', 'Steve', 'Greg', 'Jeff', 'Bill'
female_names = ['Amy', 'Joan', 'Lisa', 'Sarah', 'Diana', 'Kate', 'Ann', 'Donna

#attribute sets
career = ['executive', 'management', 'professional', 'corporation',
          'salary', 'office', 'business', 'career']
family = ['home', 'parents', 'children', 'family', 'cousins', 'marriage',
          'wedding', 'relatives']

gender_occupation_query = Query([male_names, female_names],
                                [career, family],
                                ['Male names', 'Female names'],
                                ['Career', 'Family'])
```

3. Finally, we run the Query using WEAT as the metric.

```
weat = WEAT()
results = weat.run_query(gender_occupation_query, word2vec_model)

print(results
```

```
{'query_name': 'Male names and Female names wrt Career and Family',
 'result': 1.251}
```

As we can see, the execution returns a *dict* with the name of the executed query and its score. The score being positive and higher than one indicates that *word2vec* exhibits a moderately strong relationship between men's names and careers and women's names and family.

Running multiple Queries

In WEFE, we can easily test multiple queries in one single call:

1. Create the queries:

```
from wefe.utils import run_queries
from wefe.datasets import load_weat

# Load the sets used in the weat case study
weat_wordsets = load_weat()

gender_math_arts_query = Query(
    [male_names, female_names],
    [weat_wordsets['math'], weat_wordsets['arts']],
    ['Male names', 'Female names'],
    ['Math', 'Arts'])

gender_science_arts_query = Query(
    [male_names, female_names],
    [weat_wordsets['science'], weat_wordsets['arts_2']],
    ['Male names', 'Female names'],
    ['Science', 'Arts'])
```

2. Add the queries to an array:

```
queries = [
    gender_occupation_query,
    gender_math_arts_query,
    gender_science_arts_query,
]
```

3. Run the queries using WEAT:

```
weat_results = run_queries(WEAT, queries, [word2vec_model])
weat_results
```

Note that these results are returned as DataFrame objects.

model_name	Male names and Female names wrt Career and Family	Male names and Female names wrt Math and Arts	Male names and Female names wrt Science and Arts
word2vec-google-news-300	1.25161	0.631749	0.535707

We can see that in all cases, male names are positively associated with career, science and math words, whereas female names are more associated with family and art terms.

While the above results give us an idea of the gender bias that word2vec exhibits, we would also like to know how these biases occur in other Word Embeddings models.

We run the same queries on two other embedding models: "glove-wiki" and "glove-twitter".

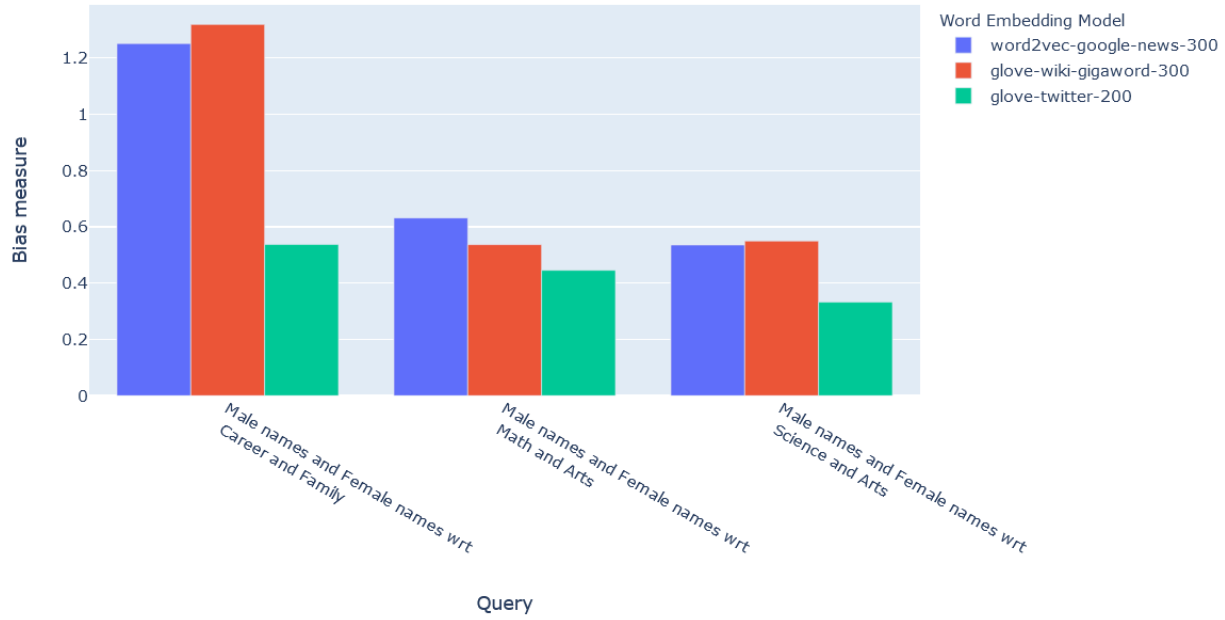
1. Load glove models and execute the queries again:

```
# load the models.
glove_wiki = WordEmbeddingModel(api.load('glove-wiki-gigaword-300'),
                                'glove-wiki-gigaword-300')
glove_twitter = WordEmbeddingModel(api.load('glove-twitter-200'),
                                'glove-twitter-200')

# add the models to an array.
models = [word2vec_model, glove_wiki, glove_twitter]

# run the queries.
results = run_queries(WEAT, queries, models)
results
```

Model Name	Male names and Female names wrt Career and Family	Male names and Female names wrt Math and Arts	Male names and Female names wrt Science and Arts
word2vec-google-news-300	1.25161	0.631749	0.535707
glove-wiki-gigaword-300	1.31949	0.536996	0.549819
glove-twitter-200	0.537437	0.445879	0.332440

2. We can also plot the results:

```
from wefe.utils import plot_queries_results
plot_queries_results(results)
```



Aggregating Results

The execution of run_queries in the previous step gave us various result scores. However, these do not tell us much about the overall fairness of the embedding models.

We would like to have some mechanism to aggregate these results into a single score.

To do this, when using run_queries, you can set the add_results parameter to True. This will activate the option to add the results by averaging the absolute values of the results and putting them in the last column of the result table.

It is also possible to ask the function run_queries to return only the aggregated results by setting the return_only_aggregation parameter to True.

```
weat_results = run_queries(WEAT,
                           queries,
                           models,
                           aggregate_results=True,
                           return_only_aggregation=True,
                           queries_set_name='Gender bias')
weat_results
```

model_name	WEAT: Gender bias average of abs values score
word2vec-google-news-300	0.806355
glove-wiki-gigaword-300	0.802102
glove-twitter-200	0.438586

The idea of this type of aggregation is to quantify the amount of bias of the embedding model according to various queries. In this case, we can see that glove-twitter has a lower amount of gender bias than the other models.

Rank Word Embeddings

Finally, we would like to rank these embedding models according to the overall amount of bias they contain. This can be done using the create_ranking function, which calculates a fairness ranking from one or more query result.

```
from wefe.utils import create_ranking

ranking = create_ranking([weat_results])
ranking
```

model_name	WEAT: Gender bias average of abs values score
word2vec-google-news-300	3
glove-wiki-gigaword-300	2
glove-twitter-200	1

You can see this tutorial code in this notebook and the complete reference documentation including a user guide, examples and replication of previous studies at the following link.

If you like the project, you are more than welcome to "star" it on Github.

[1] P. Badilla, F. Bravo-Marquez, and J. Pérez WEFE: The Word Embeddings Fairness Evaluation Framework In *Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI 2020)*, Yokohama, Japan.