Teaching AI when to care about gender

James Powell
Los Alamos National Laboratory

code{4}lib JOURNAL

Issue 54, 2022-08-29

Teaching AI when to care about gender

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) concerned with solving language tasks by modeling large amounts of textual data. Some NLP techniques use word embeddings which are semantic models where machine learning (ML) is used to learn to cluster semantically related words by learning about word co-occurrences in the original training text. Unfortunately, these models tend to reflect or even exaggerate biases that are present in the training corpus. Here we describe the Word Embedding Navigator (WEN), which is a tool for exploring word embedding models. We examine a specific potential use case for this tool: interactive discovery and neutralization of gender bias in word embedding models, and compare this human-in-the-loop approach to reducing bias in word embeddings with a debiasing post-processing technique.

by James Powell (0000-0002-3517-7485), Kari Sentz (0000-0002-1530-1952), Elizabeth Moyer (0000-0003-1604-6049), Martin Klein (0000-0003-0130-2097)

Introduction

In the 1800s, the author Mary Ann Evans wrote under the pseudonym George Eliot due to gender bias and widely embraced stereotypes about female authors. Due to the recent pandemic, online video conferencing replaced face-to-face interactions. In this isolated and impersonal space, many chose to overtly assert gender identity by specifying their preferred personal pronouns. Today we are now much more inclined to embrace our gender identity, both personally and professionally. But the passage of time does not erase the past, for which we have a vast digitized written record. So we face a new dilemma, as we become more dependent upon machine learning in our daily lives, we run the risk of unseen bias having unforeseen consequences. So we need to neutralize biases, such as those associated with gender that are perpetuated in machine learning models. To address this issue, the emerging consensus suggests we need to inspect and adjust for bias that has made its way into ML models. In other words, we ought to adjust our models, rather than our stories.

What are word embeddings?

Word embedding models encode semantic information about words that they learn based on the contexts of those words in unlabeled (raw) text. There are various ways to create word embeddings, but the results are typically a word matrix of 50-300 numeric values per word that encode information about the context of each word in the corpus. These values represent the location and magnitude of a vector describing each word, which can be used to measure proximity and distance among terms in the word embedding space. The two most common approaches for generating word embeddings are word2vec (Mikolov et al. 2013) and Global vectors for word representation (GloVE) (Pennington et al. 2014). GloVE generates word vectors from a large matrix that contains counts of word co-occurrences across the entire training corpus. Word2vec generates a model of co-occurrence patterns in text which is built up as it slides a fixed length word “window” over the sequence of text from the corpus. This creates a model that can predict a word or a sequence of words given another word, or words, as input. The model is encoded as a list of words in the corpus, and a sequence of numbers (a vector) that is an approximate numeric representation of word co-occurrences found in the corpus. Vectors that represent similar words will have endpoints that are closer to one another (Figure 1). These vectors can be evaluated to determine just how similar two words are by using a metric such as cosine distance, which is a simple geometric measurement of the angle between two word vectors. When this angle is small, it means that two words are semantically (or in some cases, syntactically) similar.

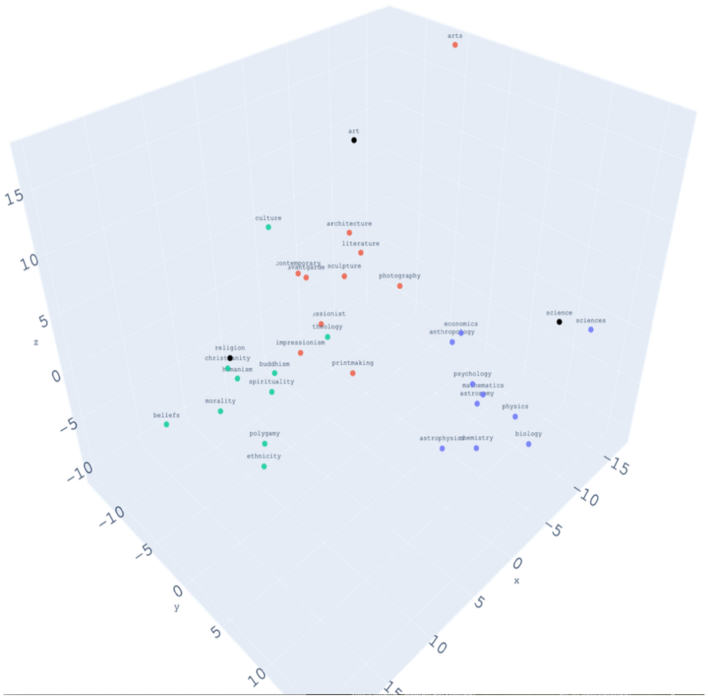
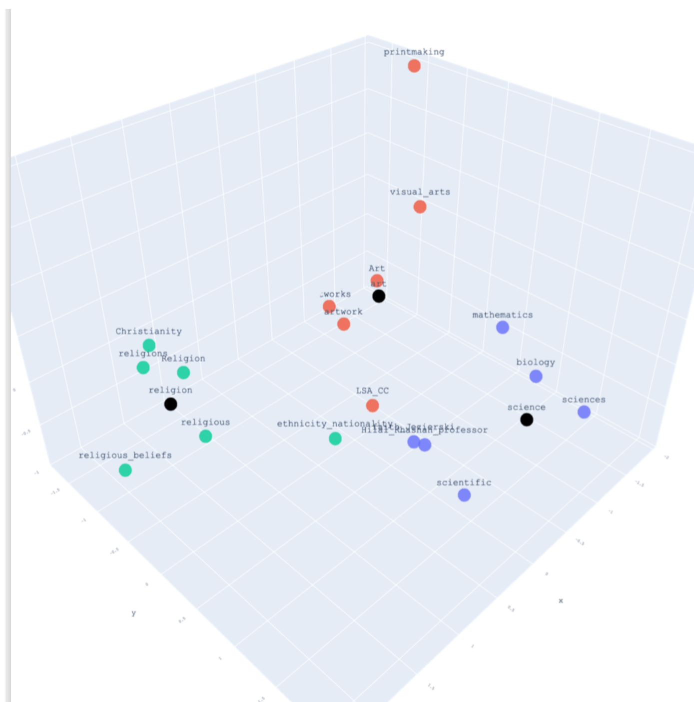


Figure 1. Principal component plots for the terms art, science, and religion showing their nearest neighbors in the New York Times corpus at left, in GoogleNews at right.

Word embeddings have characteristics which make them suitable for both intrinsic tasks such as measuring word similarity, and extrinsic tasks by serving as input to Machine Learning (ML) pipelines designed to solve NLP problems (Whitaker et al. 2019). Extrinsic NLP tasks include predicting whether a text segment is positive or negative (sentiment analysis), assigning a category to a document (such as a subject heading), or determining whether one sentence logically follows another (entailment). One way to solve these kinds of problems is via supervised ML. Supervised ML requires labeled training data, that is, explicit examples of data that represent learnable patterns. Labeled training data for automated sentiment analysis might consist of movie review texts labeled as positive or negative. Given this training data, an ML algorithm can learn the patterns that constitute positive and negative sentiment (a good or bad movie review). One type of ML algorithm is a neural network. Neural networks are written to learn patterns in data, rather than to solve a specific problem. Given large amounts of training data, a neural network can learn a model that can make predictions about new, previously unseen data. Word embeddings can be used to speed up training of neural networks when the training data contains text, because word embedding models already contain useful information about text, such as which words have similar meanings. Using a pretrained model to support training of another model is referred to as transfer learning, because it enables re-use of an existing ML model to facilitate training of another model for a different task. Transfer learning also speeds up training of new predictive models since the neural network no longer needs to relearn everything from scratch.

For many deep learning/NLP tasks, it is standard practice to download pre-trained models such as GoogleNews, Common Crawl, and wikipedia word embeddings models and use them for deep learning tasks. Since pretrained models are trained on extremely large corpora, they often contain superior word co-occurrence vectors to those that would be found in a smaller model. Large word embedding models work well if there is enough overlap between the vocabulary of the model and the vocabulary of the text of the local training corpus. If there is a significant mismatch, where there are a lot of vocabulary words not found in the embedding model, then large word embedding models are less beneficial. In those cases, researchers will train local word embedding models. This might be done for example in cases where an embedding model for the native language of the text does not exist, when the vocabulary used in the corpus is highly specialized, such as is the case with genre specific text or scientific publications, or if they want to work directly with a particular word embedding model or set of models, as is the case with diachronic text analysis. Diachronic text analysis involves using word embeddings that include a time dimension, allowing for measurement of change among words over time. The DUKWeb project (Tsakalidis et al. 2021) is a large-scale effort to generate pre-trained diachronic word embeddings for the contents of websites in the .uk subdomain retrieved from the Internet Archive, allowing researchers to study the changes in relationships and meaning among words in this corpus spanning 1996-2013.



When word embeddings go wrong

The Common Crawl [1] word embedding model is trained on one of the largest corpora that is publicly available. The corpus consists of 2.96 billion active and defunct web pages which have been collected since 2008. A 2022 study entitled "Based on billions of words on the internet, people = men" (Bailey et al. 2022), found that in a word embedding model trained on *Common Crawl*, the cosine distance between word embedding vectors for "men" and "people" was smaller than the cosine distance for embedding vectors for "women" and "people." Validation of this hypothesis inspired a second line of inquiry based on a follow-on hypothesis that, given the model's conflation of "people" with "men", gender stereotype associations in this model would be asymmetric. They also found evidence to support this. Their results showed that women were more likely to be associated with gender stereotypical language than men.

In 2016, a group of researchers wrote a landmark paper (Bolukbasi et al. 2016) about the prevalence of gender stereotypes in large word embedding models, and proposed techniques for addressing it. They first demonstrated this problem via analogies mapped to simple vector math, showing that many common gender stereotypes were present in word embeddings trained on the Google News corpus. They enlisted crowdsourcing to identify two sets of 100 pairs of words that could be used to identify definitional (e.g. she, grandmother) and stereotypical (she, secretary) gender associations. Next, they used ten of these gender pairs to identify what is in essence a direction for gender.

Their concept that gender might be associated with a direction in a word embedding model inspired them to propose several debiasing techniques for word embeddings. In one approach, they use gender-specific words to find a vector representing a gender direction common to the terms. Using this "gender" vector, they identify a second vector that is orthogonal to it. They then perform debiasing by projecting (move) gendered words onto this orthogonal vector.

This eliminates the gender direction from the terms, which they suggest results in the neutralization of gender in these word embedding vectors. This causes words that previously had a strong association with one gender to be equidistant from gender words. For example after the adjustment, "secretary" would be the same distance from "he" as it is from "she". Their second approach aims to preserve gender associations while reducing bias. For example, this approach ensures that "grandmother" retains a relationship to "female" gender, and "grandfather" to male gender. It then adjusts terms that might have a gender association learned from bias in the corpus. This might cause the terms "hiking" and "baking" to be adjusted so that they are equidistant from "grandmother" and "grandfather." This is a much simplified description of their work, and this paper is definitely worth a read in its entirety.

It is important to note the central role of human judgment in the research described above. The authors are careful to note that human judgment plays an important role in their approach to identifying bias and debiasing word embedding models. They broadly observe that "gender associations vary by culture and person." They also discuss at length the process of manually creating gender word pairs suggesting that "the choice of words is subjective and ideally should be customized to the application at hand." We will revisit these issues shortly.

Gender bias has historically had a significant impact on many aspects of life, so the identification of gender bias in data that is so central to many machine learning algorithms, and the idea that this bias could possibly be neutralized was an extremely important step for the field. This opened the door for the possibility that other forms of social bias (Garg et al. 2018) in this data could be identified and mitigated. But social biases are not the only problems that can manifest in word embeddings. Fixed word embedding models trained on large corpora will inevitably encounter words that have multiple meanings, or senses. This results in a diffuse representation that does not represent any sense of the word very well. For example, the word "bank" has several distinct uses, including "financial institution" and "the land adjacent to a river." If a training corpus includes references to both senses of "bank", then the resulting embedding vector will represent the merger of what are effectively two semantically distinct words represented by the same string of characters, which have different word neighbors and usages. In other cases, the quality of some word embedding vectors may suffer due to a paucity of examples in the text. Sometimes an embedding model may encode cultural biases, or be heavily influenced by false or misleading information sources, or it may skew toward particular opinions, political positions, or scientific disciplines.

Word embeddings may also have problems learning consistently meaningful representations for distinct scientific disciplines, particularly for terms that span multiple disciplines. A scientific corpus skewed towards physics and other hard sciences will generate better representations for terminology in those fields, but will offer little to no meaningful semantic representations for other fields such as sociology or linguistics. Nor would it be useful for identifying important co-occurrences that might be indicative of multidisciplinary efforts. This problem, perhaps obviously, is more likely to occur for word embeddings generated for a scientifically focused corpus, and it exemplifies the tradeoff between capturing a good representation for a technical vocabulary over modeling science more broadly or depending on a large-scale word embedding model.

Digital libraries provide wide ranging access to intellectual and cultural artifacts. Machine learning models can be used to support many features of digital libraries. They can be used to suggest topics for documents in a corpus, recommend or summarize content, perform automated text classification, and support query formulation and offer search term suggestions, to name a few. If any of these features depends on a model that has problems such as encoded bias, incomplete semantic representations, or term meanings affected by skewed points of view; it could potentially produce unpredictable or misleading or even offensive results. This could be particularly harmful if, for example, some type of bias affects the perception or distribution of historical or culturally sensitive content, or if it results in the inadvertent censoring of certain ideas due to current political sensibilities. It is also important to note that digital libraries can be the source of corpora used to produce word embedding models for various tasks, treating library collections as data (Padilla et al. 2019). Collection curators are in a particularly good position to identify and mitigate problems with word embeddings trained on their corpora if they can be empowered to do so.

The *Word Embeddings Navigator*

We believe that one of the best ways for a human to "explain" bias to a word embedding model is to enable them to inspect that model and give them the ability to iteratively perform targeted adjustments to vectors in that model. The goal of explanation "is to fill in the gap between his audience's knowledge or beliefs about some phenomena and what [they take] to be the actual state of affairs" [2]. With that goal in mind, we introduce the *Word Embeddings Navigator* (*WEN*), an interactive Web application for exploring and modifying word embedding models. *WEN* facilitates human-in-the-loop interaction with word embedding models to enable iterative query and adjustment of term vectors. Although there are other word embedding visualization tools, including Conceptvector (Park et al. 2017), Embedding Projector (Smilkov et al. 2016), vec2graph (Katricheva et al. 2019), and WordBias, a tool which was specifically designed to facilitate bias detection through visualizations (Ghai et al. 2021); we believe this is the first tool that combines visual word embedding exploration with the ability for a user to interactively adjust individual word vectors. In the following sections we will illustrate how this works, describe some of the technical details of the system, and evaluate how it performs in comparison to post-processing debiasing.

WEN was originally conceived of as a tool for exploring and adjusting temporally aligned word embedding models. For example, figure 2 shows a *WEN*-generated heatmap visualization of terms near the word "mars" in temporal snapshots of a scientific corpus. *WEN* was a product of an internally funded research project (Sentz et al. 2019). A broader goal of the project was to support discovery of latent knowledge stored in an embedding model (Tshitoyan et al. 2019). Latent knowledge exists in all corpora but for scientific corpora that span significant time periods within a particular field, there are often hints of discoveries that have yet to be formally declared, and these can be represented by similar contexts that occur at different points in time. The ability to change word embedding vectors was added when it became apparent that small, locally trained models failed to capture relationships that were immediately obvious to subject matter experts, but were not reflected in the source training data. Adding a mechanism to change individual embedding vectors fit nicely into the application since those changes would be made to the embedding model in memory, and thus manifested in subsequent user queries. We envisioned that users would make these adjustments based on specialized knowledge such as that possessed by a subject matter expert.

How *WEN* works

WEN is designed to allow for interactive exploration and iterative adjustment of word2vec word embedding models or other embedding models, such as GloVE embeddings, that can be converted to word2vec format. It emphasizes a four step approach: "search – navigate – visualize – adjust". All four phases are in service of the goal of surfacing quality issues in word embeddings and making it possible to address these issues directly and immediately within the model. We implemented a prototype of this concept, at the core of which is a capability we refer to as interactive refitting. The prototype consists of the following components:

- A search interface for providing one or more search terms
- A textual result interface that presents most similar terms from the word embedding model
- Links to explore these results via three Web visualization tools

- A mechanism allowing users to identify and specify target for adjustment via refitting
- A Web services wrapper for our implementations of the refitting objective
- An update function that modifies selected word2vec embedding vectors and stores them in the live model

The embedding navigator tool makes extensive use of the gensim [3] python library for natural language processing. An instance of the navigator can be configured to load an arbitrary number of related or unrelated word embeddings in word2vec binary or text format, only limited by system memory. At launch time, navigator loads a configuration file, and as a Flask application, it then iterates through file system references to the embedding files and loads them into memory. Since this tool allows users to update individual word embedding vectors, we load the full word embedding model, rather than just a dictionary of words and their vectors, which would have a smaller memory footprint but would not allow model updates. At run time, *WEN* parses its configuration file to discover which models to load, and there is additional information provided to support the user interface as illustrated by this JSON metadata excerpt:

```
{
  'name': 'NYT 1987-1991',
  'filename': 'NYT-1987-1991.model',
  'label': 'nyt-87-91'
}
```

The name is the full corpus name to be presented to the user, filename is the primary word2vec binary model file to be loaded at runtime, and the model label is used in the UI to provide short labels for interface form elements and various visualizations. A serialized model is loaded from the filesystem using a call to gensim's word2Vec, e.g.:

```
this_model = Word2Vec.load(models['filename'])
```

where models['filename'] corresponds to a file containing word2vec embeddings.

Metadata about the loaded models is utilized throughout the interface. Flask templates ensure that the user can query any or all of the embedding models depending on what aspect of the embeddings they are interested in. Since *WEN* was initially conceived of as a tool to explore temporal slices of word embeddings, there are some facilities for visualizing the behavior of a group of words associated with a query across a set of embeddings, usually a temporally aligned collection of models. A sankey diagram illustrates the changes in similarity to the target word in each instance of the loaded embeddings. This works for temporally aligned and unaligned word embeddings as well. It can also be used to view non-temporal shifts among a set of word embeddings.

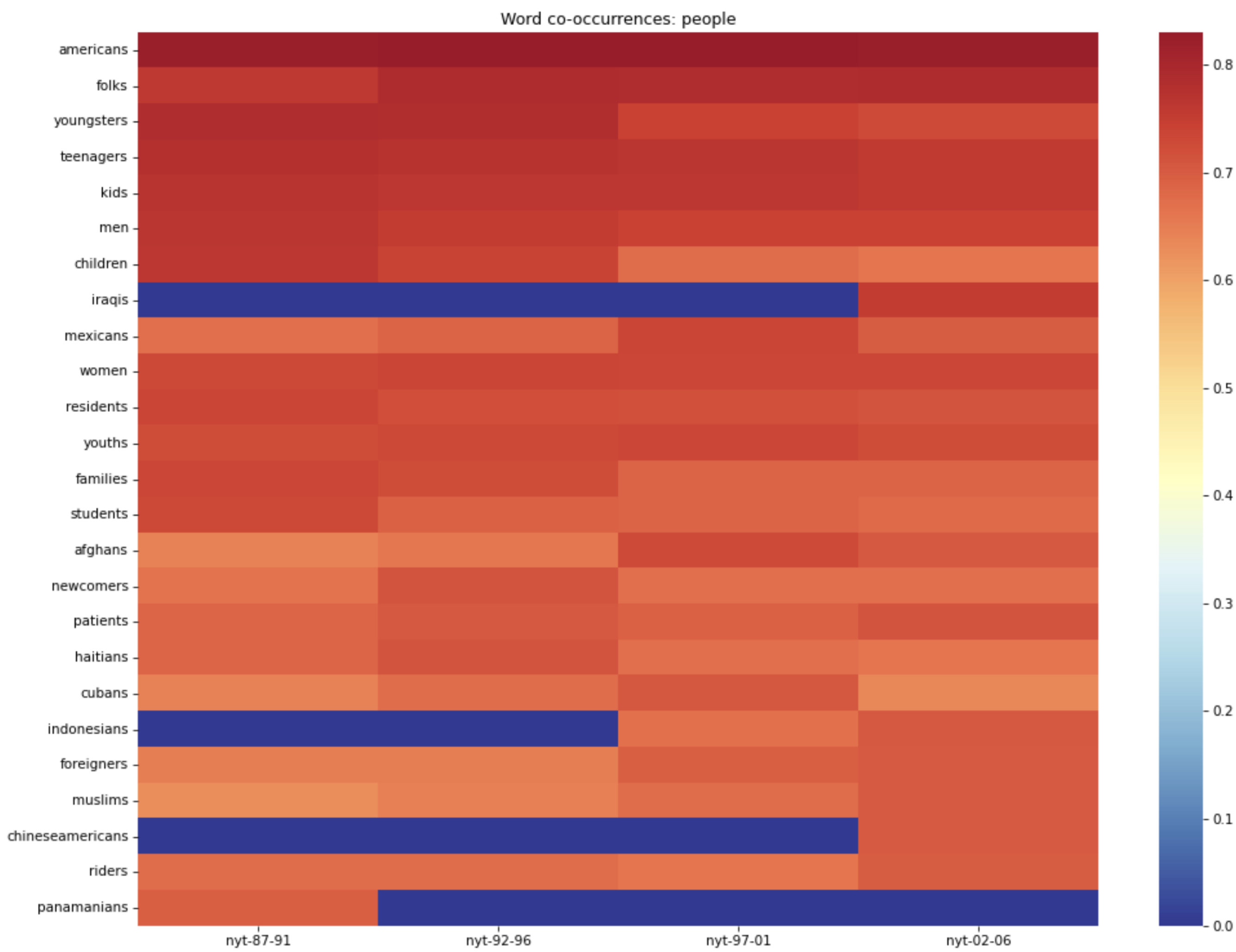


Figure 2. WEN-generated heatmap for terms most similar to “people” across New York Times word embedding models spanning 1987-2006.

WEN use cases

The search interface allows for several types of queries (Figure 3). The basic query is for a word or phrase in a single user selectable word embedding model. The query itself is performed using the *gensim* “most_similar” method of the word embedding model. It computes the cosine similarity of the query word’s vector to all other word vectors in the target embedding space, and returns a list of entries that are the most similar to the query term. Somewhat counter intuitively, the most_similar method can also be used to perform a query that involves subtracting the vector of one term from another and then finding the most similar entries to the resulting vector. We implement this as the difference query.

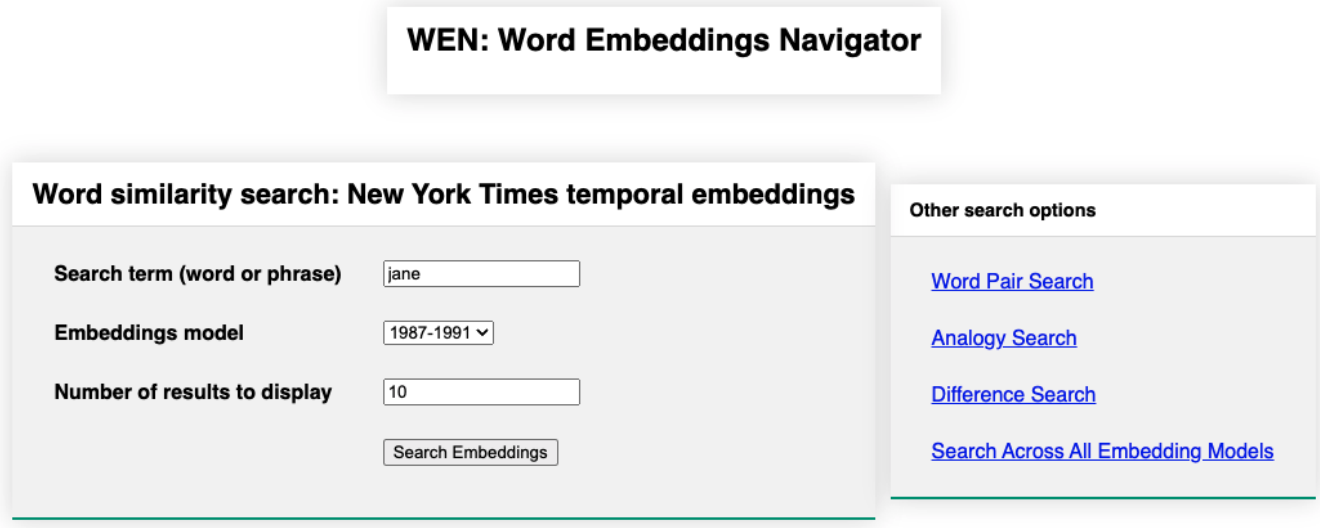


Figure 3. Search interface for an instance of the Word Embeddings Navigator exposing four temporally adjacent word embedding models for the New York Times annotated corpus (1987-1991, 1992-1996, 1997-2001, 2002-2006).

Embedding navigator supports several other methods for interactively querying an embedding model. The analogy search corresponds to a common means of evaluating word embedding models: X is to Y, as A is to B. The user provides three of the four elements of the analogy query. In its simplest form this sort of match would be performed using very basic vector arithmetic where the most similar left out term is the term closest to the vector Y-X +A. An important characteristic of the gensim library is that it includes implementations of many of the latest state of the art approaches to solving natural language processing tasks. For analogies, gensim’s word2vec class provides an improved method for the analogy task in the form of a method called most_similar_cosmul. This method uses a normalized multiplicative objective called 3COSMUL when calculating best matches for the left out analogy parameter. The exact method call looks like this:

```
this_model.w2v.most_similar_cosmul(
    positive=[term1, term2], negative=[term3], topn=10
)
```

Where X and A are the positive terms, Y is the negative term, and topn indicates the number of matches to return. When compared to simple vector math for solving analogy questions with word embeddings, 3COSMUL provides a 20% improvement (Levy et al. 2014) in correctly identifying analogy relationships.

The most_similar_cosmul method is also used to identify terms that are most similar to a pair of terms. Although the actual calculation is a bit more complex, in essence, the best matches are those which are closest to the normalized average of the two query terms provided by the user. This is indicated by the inclusion of a positive parameter, and omission of a negative parameter passed to the method.

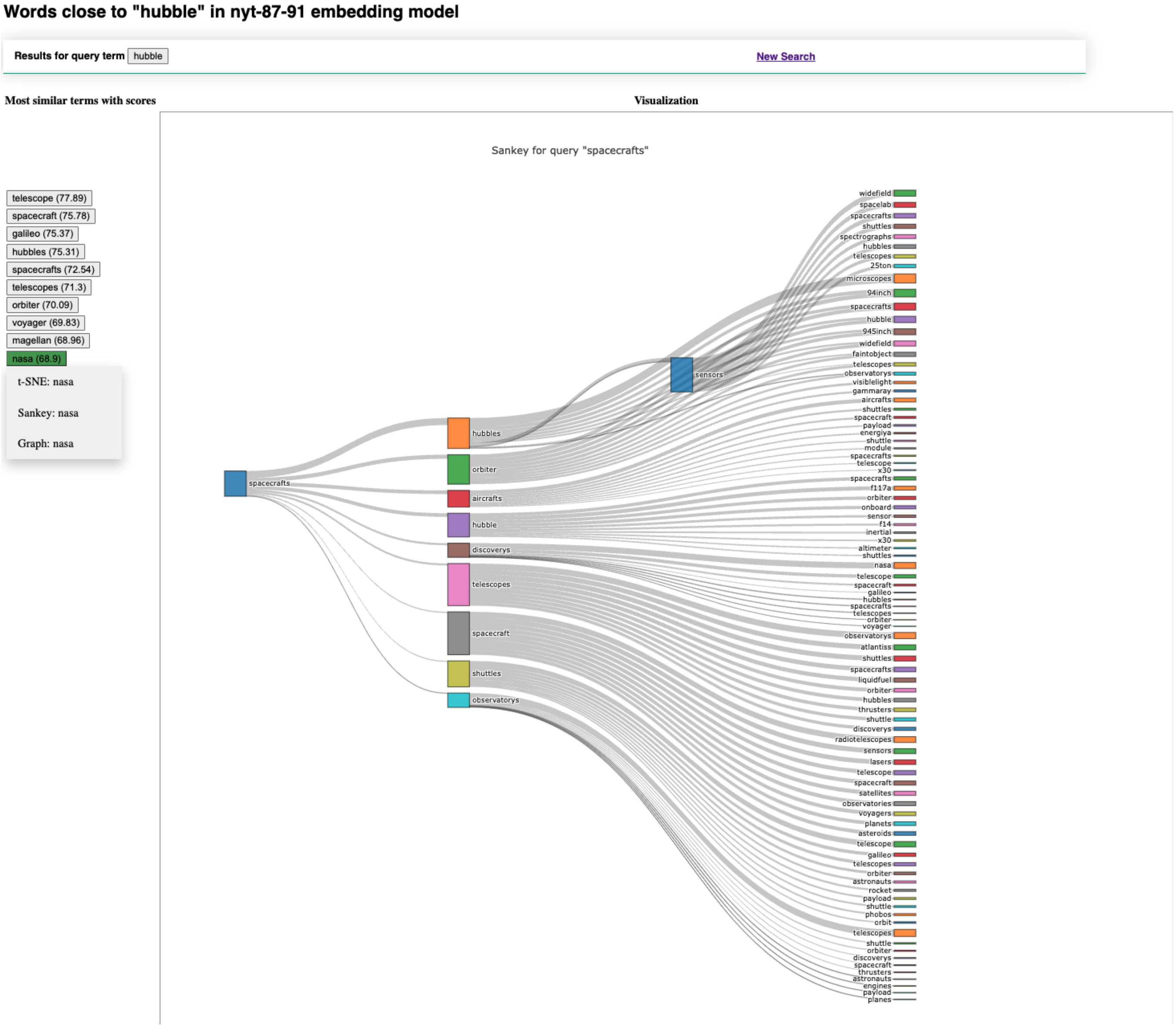


Figure 4. A sankey diagram for the term “spacecrafts” from the 1987-1991 instance of the NYT corpus.

These four query options form the core of the embedding navigator search functionality. Results are available in several forms and the user can specify the number of results they wish to see at query time. The basic results form is a sorted list of most similar terms together with their respective cosine similarity scores. From this listing users can perform several tasks. The list provides click to query access for each term in the results set. There are also three visualization options provided: a t-SNE plot of the most similar terms, a sankey flow diagram of first and second order results, and a graph, or network visualization, which also presents first and second order results. Since two of the visualizations are graph based and it may not be immediately obvious how this would be achieved, we explain it here. First order results are those which are in close proximity to the original query term. Second order similar terms are the set of results associated with each first order search result. Thus the farthest nodes in the network visualizations would generally be two degrees from the initial query term. The advantage of providing a network view is that it can reveal instances where a term has multiple connections among nodes in the first or second order result sets.

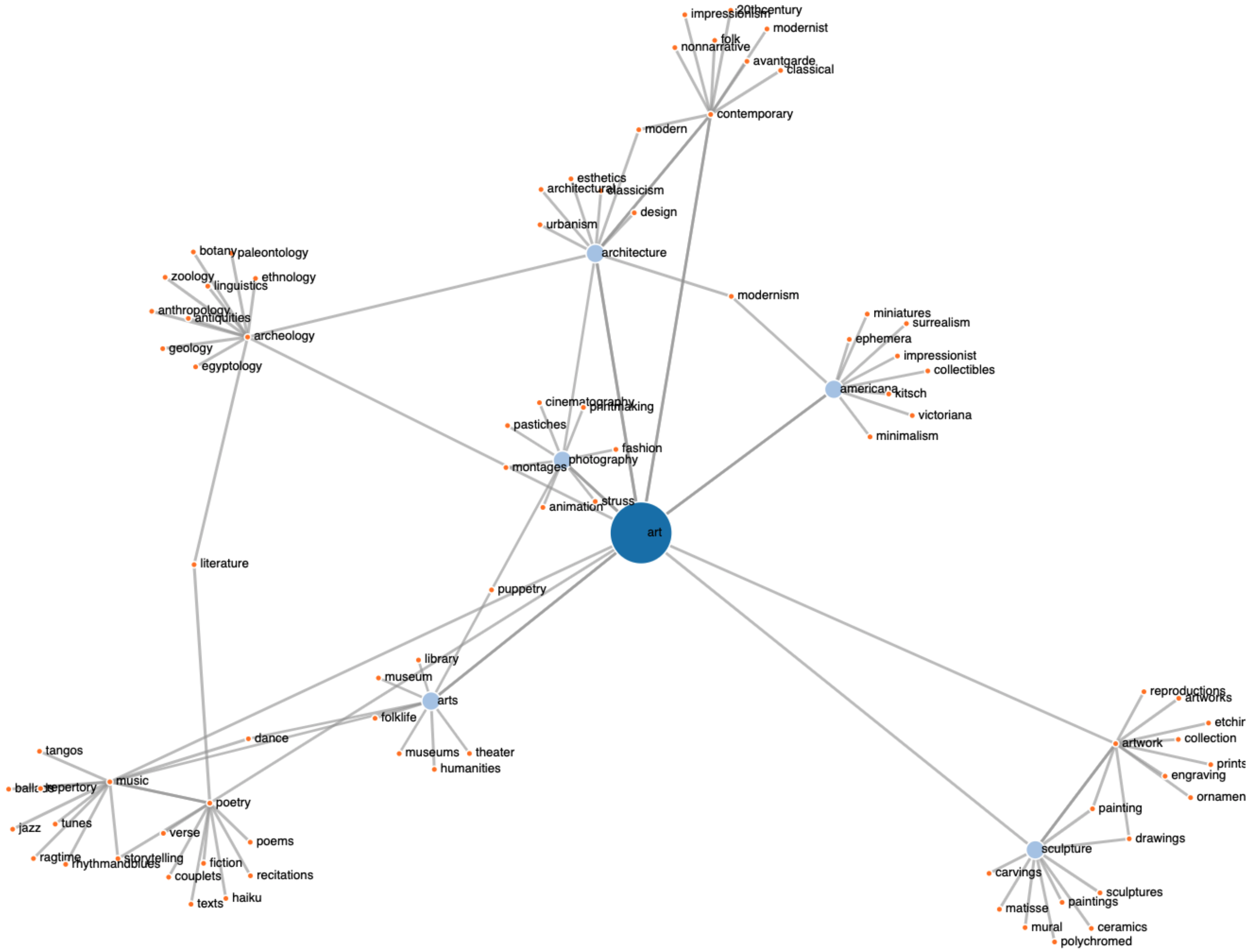


Figure 5. A force-directed graph of the term "art" together with its most similar terms in the embedding model.

t-SNE (Van der Maaten et al. 2008) is a technique for visualizing high dimensional data which preserves a reasonable degree of variation during dimensionality reduction. The true value of t-SNE is realized when visualizing both high dimensional data and a large number of data points. We use a javascript implementation of a t-SNE visualization tool which performs the dimensionality reduction in the browser. For small results sets, performance is adequate and we find that the plots it generates accurately reflect the relationships of the embedding vectors. While it is a tall order to expect good performance from browser-based dimensionality reduction of large amounts of high dimensional data, we include this capability because it provides a different perspective on a cluster of word embedding vectors. t-SNE remains a state of the art technique for visualizing high dimensional data.

As alluded to above, the other two visualizations provided are both based on network visualizations. The sankey diagram (Figure 4) shows the query term and two levels of results. The line connecting each result indicates how similar the term is to the term it is connected to. It is implemented using the javascript version of the plotly library [4], so it provides connection information when the user hovers over a result node, which is particularly useful when there is only a small amount of variation in the similarity scores. From the sankey diagram, the user can change the number of search results for the current query to see additional words in relation to the initial search term.

The user can also elect to view the result set as a graph (Figure 5). The graph implementation is a basic network diagram implemented in d3 [5]. The graph does not utilize similarity scores so edge width is uniform. This graph is intended to provide a comprehensive view of a term in relation to its nearest (most similar) neighbors in the embedding space. The graph uses a force directed layout, which is a layout model that tries to minimize overlap of nodes. The user can interact with nodes in the graph to further reduce overlap in a given region of the graph by selecting and dragging a node. A major advantage of the graph view is that recurring nodes are connected wherever they occur in the result set so that the user can see connections such as cycles and clusters that occur when the same term occurs in multiple results sets. These are relationships which are not depicted in the sankey view. So to summarize, the t-SNE view provides an approximate 2-D representation of term proximity, the sankey view links related terms and conveys similarity via node and edge width, while the graph view provides a more complex contextual perspective where first and second order results are visible.

Adjusting word embeddings with interactive refitting

Human interaction with word embeddings allows for the injection of human judgment into the model. Humans are better at detecting bias and can use their expert knowledge to improve or correct other relationships in the embedding space. We are good at organizing terms according to more nuanced relationships, for example by emphasizing particular word groupings, flagging certain words as orthogonal to other words, reducing emphasis on syntactic relationships, and adjusting for the effects of bias. Interaction can occur after embeddings are generated via an unsupervised method. User provided modifications to the embedding space can be directly applied to word vectors or recorded and analyzed for consistency and resolution if re-fitting adjustments from different users target the same word or set of words.

There's an important caveat to the benefits of having a human identify and minimize bias: we all have implicit biases. Implicit biases are unconscious stereotypical associations we learn over the course of our life. No matter how hard we try, most of us bring implicit biases to situations and circumstances we encounter, even efforts to address bias. This includes interactively mitigating bias in word embeddings. If you lack awareness of your implicit biases, you might inadvertently substitute one kind of stereotype for another, thereby exacerbating bias rather than reducing it. There are online tests such as those provided by "Project Implicit" [6] and "Learning for Justice" [7], which allow individuals to assess their own implicit biases. Both tests are based on the "Implicit Association Test" (Bertrand et al, 2005) which is discussed in more detail below. It is good practice for individuals engaged in evaluating and mitigating bias in word embeddings to know their own biases beforehand. It is also recommended to incorporate input from multiple individuals when possible.

With that in mind, we can now look at how *WEN* enables users to interactively address bias in word embeddings. In their paper "Retrofitting Word Vectors to Semantic Lexicons" (Faruqui et al. 2014) , the authors considered how to improve word embeddings by injecting additional semantic information into previously trained word embedding models. Their goal was to improve word embeddings by using relationships specified in a curated lexical dictionary such as WordNet [8] to adjust words that were identified as synonyms, so that their embedding vectors were more similar. They developed a light-weight post-processing approach that searches for embedding model terms in a lexical dictionary such as WordNet, identifies and locates the vectors of any synonyms, and moves the synonym vectors closer to the target term. Mathematically, their post-processing reduces the Euclidean distance of each synonym vector to the target term vector.

They were able to demonstrate that this approach improved representational semantics of the embedding vectors. Inspired by this work, we explored replacing data from a lexical dictionary with real-time human judgment. Instead of affecting the embedding model by moving synonyms, we implemented a mechanism whereby the user could interactively select words to be moved. Our project enables users to trigger one of two types of refitting of user-selected word vectors in the word embedding model using WEN (Figure 6). One moves a chosen term or list terms closer to a target word where the target word vector remains unchanged, while the other allows the user to indicate that the vectors for a set of terms should be moved closer to one another. We refer to this technique as refitting (Powell et al. 2020).

Refitting enables a *WEN* user to make adjustments to the embedding vectors of selected words and immediately see the results of these adjustments in subsequent queries and visualizations. Further adjustments can be made in an iterative fashion until the desired changes are achieved. Users can use whatever criteria they wish in order to make refitting adjustments.

Users are presented with two primary methods for modifying the embedding space: to move a single word closer to a list of related terms (Figure 7), or to move a set of words closer to one another (Figure 8). These affordances allow users to improve (decrease the distance) between embedding vectors. Each user interaction affects the vectors for the selected words. The recalculated word vectors are stored in a separate modified word embeddings table, along with a unique identifier, so that they can be used, analyzed, reclustered, and reconciled as needed. In addition to offering users the ability to refine relationships among words, users can take actions that compensate for bias in word embeddings, for example by moving gender specific pronouns closer to gender neutral pronouns.

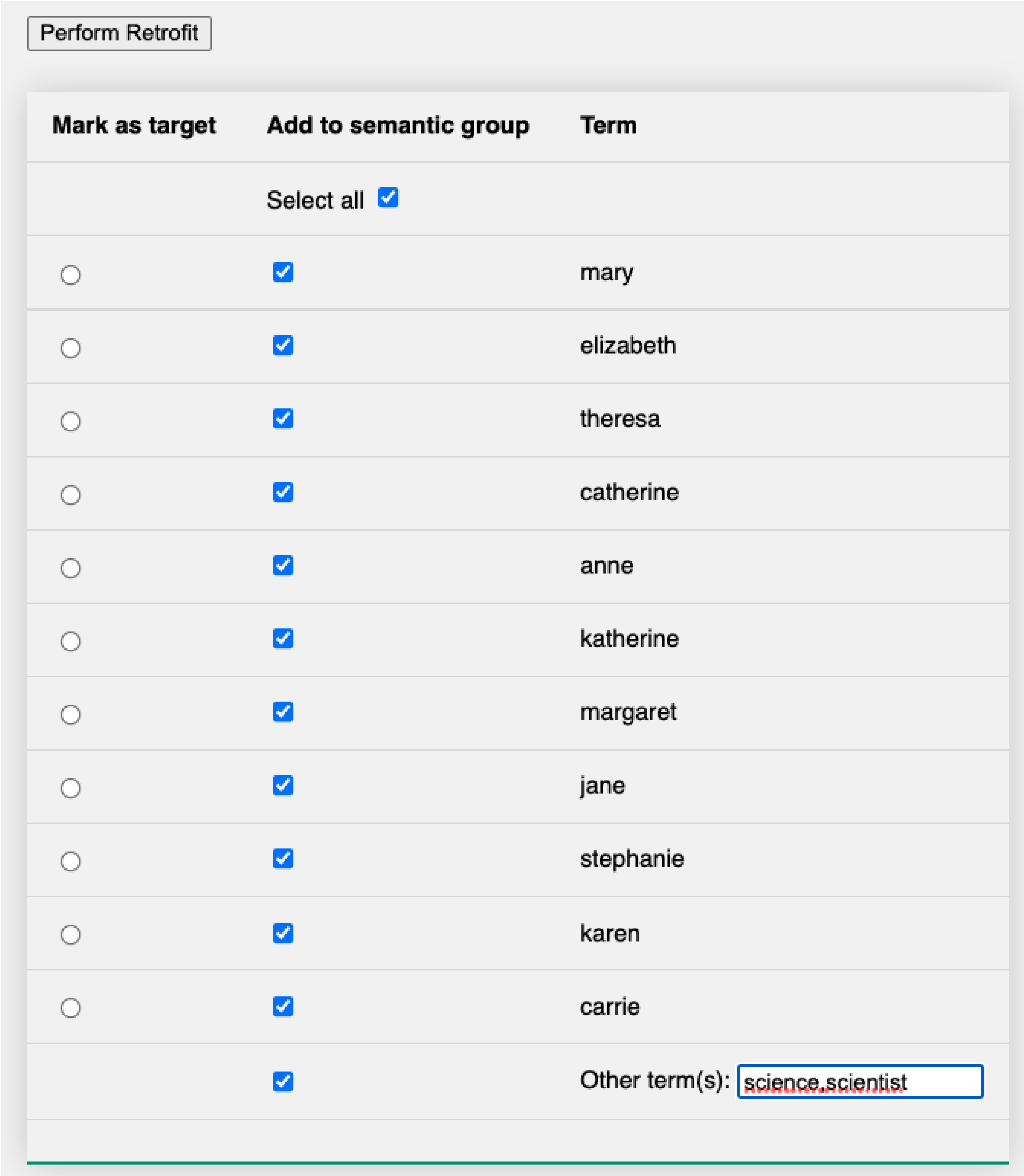


Figure 6. Example of the refitting interface which a user has configured to adjust a set of selected terms.

As noted above, the original retrofit objective is at the heart of the refitting strategy. We adapted the retrofit algorithm to support interactive word embedding adjustments and exposed it as a Web service used by *WEN*. Parameters include a set of words and optionally a target word. If a target is not provided, then each provided term will be adjusted using its original embedding vectors and the refitting objective so as to move it closer to all other terms provided. Otherwise, only the target embedding will be adjusted. The Web service returns the pre- and post-refitted vectors, as well as cosine distance scores among the terms that were refitted.

The target word embedding vectors are loaded into a dictionary from the word2vec model:

```
for idx, key in enumerate(this_model.wv.vocab):
    wordVecs[key] = this_model.wv[key]
```

The refitting function iterates over the entries in wordVecs from this_model. The new embedding vector will be temporarily stored in castVector

```
castVector = [float(i) for i in wordVecs[key]]
```

Once the word embedding vector is adjusted per user input, the refitted version is stored in the target word embedding model:

```
this_model.wv.syn0[this_model.wv.vocab[key].index] = castVector
```

The old embedding vector for the word represented by the variable “key” is replaced with a new refitted embedding vector. This changes the in-memory version of the target word2vec model. To permanently serialize the updated version of the model to the filesystem, we call the save method of the target embedding model:

```
this_model.save(model_filename)
```

When a refitting task is completed, a radar visualization is presented to show how the selected terms were moved in relation to one another, based on old and new cosine distances.

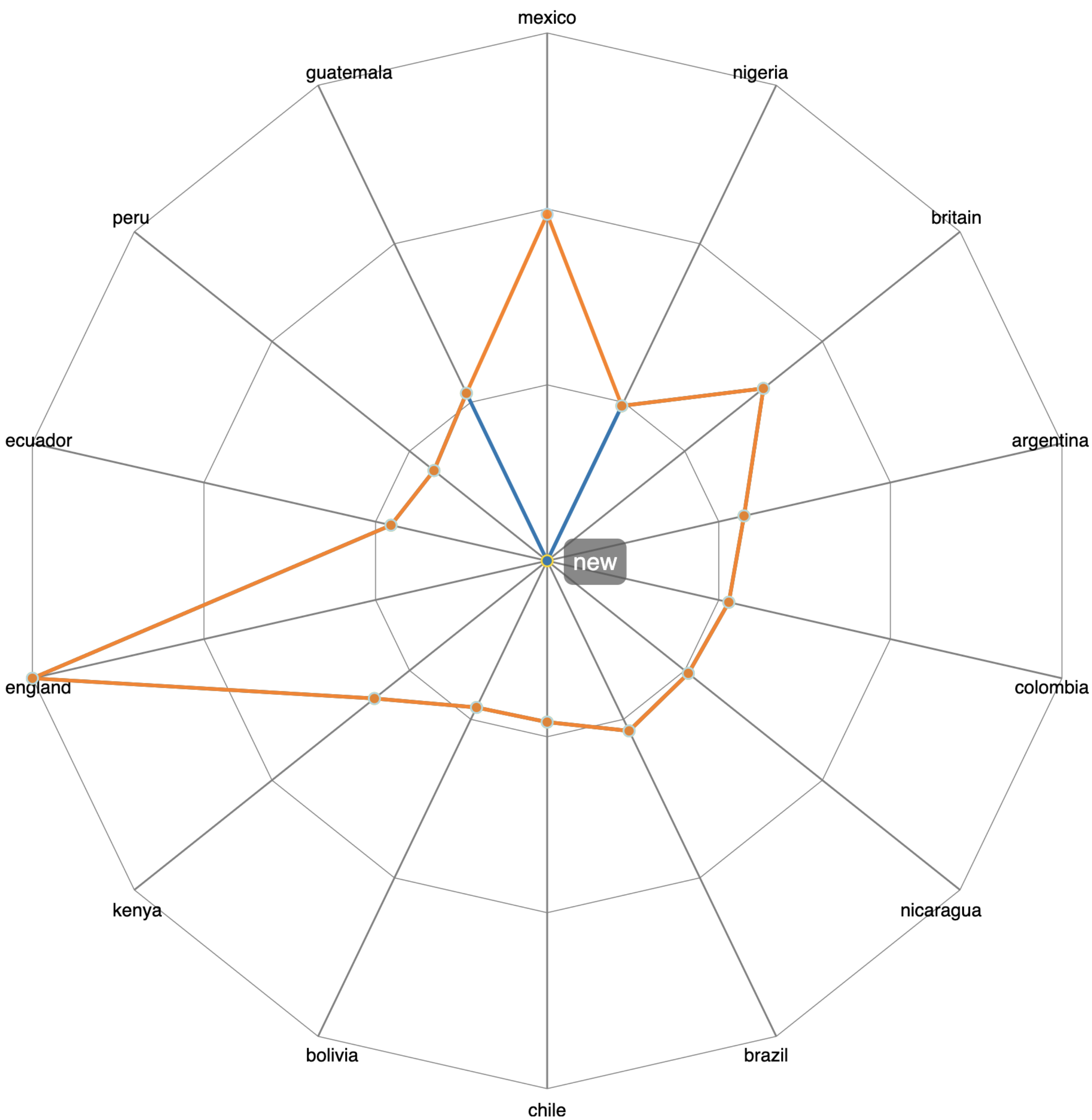


Figure 7. A before-and-after radar plot illustrating the effects of adjusting the word embedding vector for one term (“mexico”) so that it is closer to a list of other words (names of other countries).

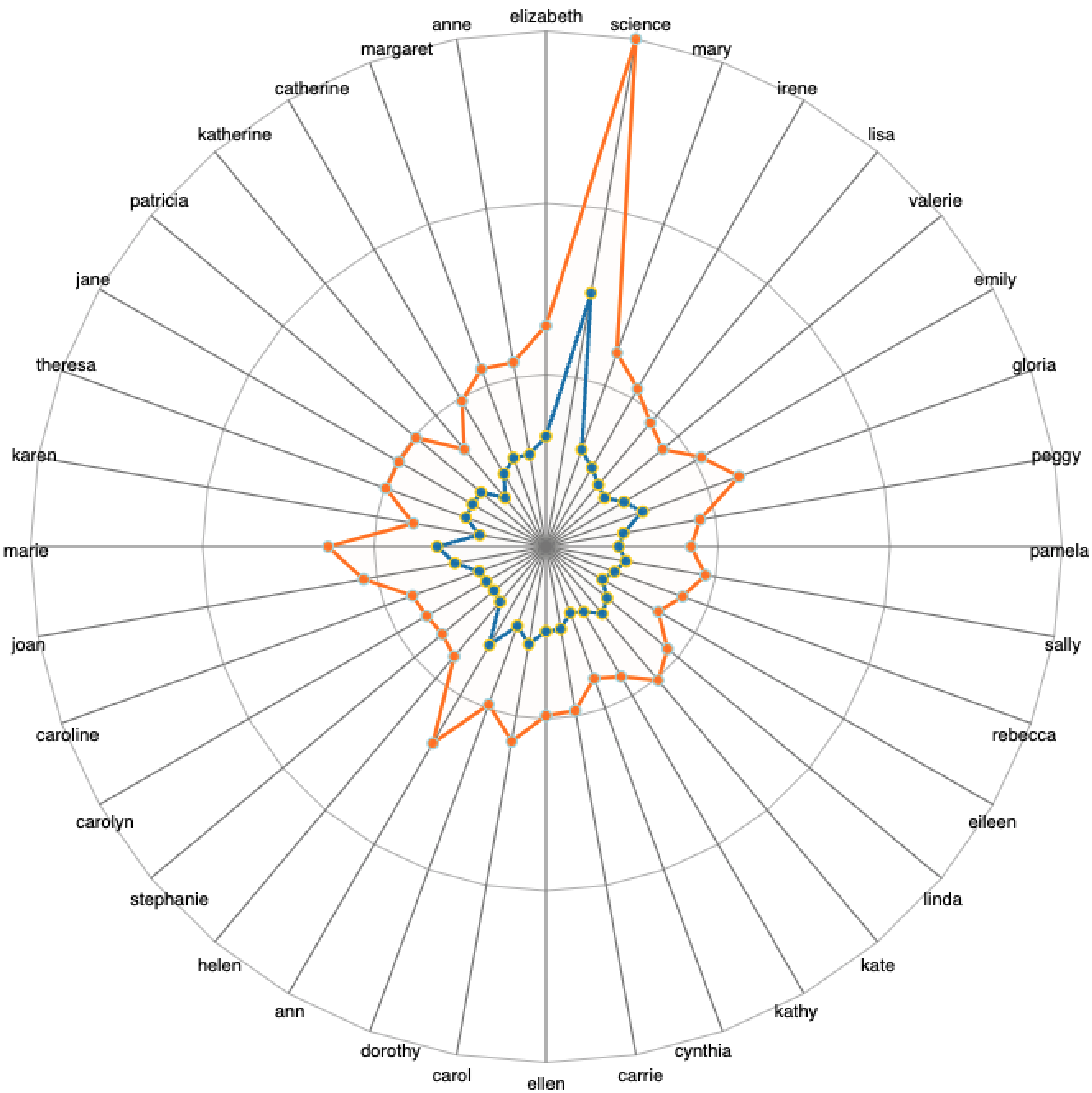


Figure 8. A before-and-after radar plot illustrating the effects of refitting an entire set of terms.

Each time an embedding vector is adjusted, WEN stores the previous and updated vector in a MySQL database, together with the identifier for the target embedding model, an action label, and a timestamp. Although this is currently just a logging feature, the plan for this data in the future is to add an interface that would allow a user to "roll back" a selected vector to a previous state.

Evaluating *WEN* refitting

We would be remiss if we were to discuss gender bias issues without acknowledging the various persistent historical biases including gender bias, that have been perpetuated in language for centuries, sometimes causing great harm. References to binary gender identities and to gender stereotypes are based on historical data and are referenced in order to provide clear examples for the purpose of potential bias mitigation techniques applied to data consumed by machine learning algorithms. It is not our intent to perpetuate or endorse any form of bias, or to make any claim that any historical social or cultural norms are superior to current norms. Furthermore, we note that the corpus and the examples used in our evaluation are all based on English language terms and publications, including news articles from the New York Times [9] spanning 1987-2006. Because of this, many of our examples may not be directly relevant to non-Western, non-English speaking audiences working with non-English corpora who are considering non-Western cultures, historical narratives, stereotypes, or biases.

Considerable effort has been invested in recent years in characterizing bias, especially gender bias, in word embeddings, for example (Brunet et al. 2019) and (Garg et al. 2018), and the impact on machine learning tasks such as machine translation (Savoli et al. 2021). From a technical perspective, two theories in particular have dominated the investigation of how gender bias is represented in word embeddings: the role of word frequency and the potential existence of a gender-specific component in word embeddings. Word frequency has also been found to impede post-processing debiasing. We believe the value of a tool such as *WEN* is that it is agnostic with respect to the way in which bias was represented in a training corpus, or how it was propagated to a word embedding model. It likely would not scale as a production tool for mitigating bias in large embedding models but is intended to be an exploratory platform in which one may perform queries and adjustments in an iterative fashion, enabling users to perform experiments based on theory and intuition.

Our data set consists of word embeddings constructed from four five year snapshots of the annotated New York Times Annotated Corpus, which contains over 1.8 million articles from the New York Times spanning January 1987 until June 2007. We generated local word2vec word embeddings for each five year collection (1987-1991, 1992-1996, 1997-2001, and 2002-2006) after extracting the articles from the original XML source and performing some rudimentary text preprocessing which normalized all text to lowercase and removed most punctuation, except for punctuation which marked sentence boundaries. We used the gensim Word2Vec library with the cbow (continuous bag of words) model, eliminating any words that occurred fewer than four times in the corpus, trained for 20 epochs resulting in word vectors containing 100 dimensions per term. The sliding window for evaluating co-occurrences was left at its default value of 5, as were other parameters not explicitly listed here:

```
model = Word2Vec(sentences=sentences, size=100,  workers=4, min_count=4,
                 sample=0.05,  sg = 0, iter=20, hs = 0)
```

Some details about the source corpora and resulting word embedding models, as well as some initial bias scores for the models, are provided in Table 1.

NYT article corpus year span		Word count	Embedding size	*WEAT*: career and family	*WEAT*: math and arts	*WEAT*: science and arts
1987-1991	490380	240723430	436398	1.48	0.84	0.77
1992-1996	391855	178927558	351059	1.64	0.95	1.03
1997-2001	448514	219950197	416824	1.68	1.1	0.94
2002-2006	447227	219969632	426894	1.68	1.04	0.74

Table 1: Corpus and embedding models overview

Using *WEN* and refitting, we conducted several experiments to determine if we could in fact reduce gender bias as measured by the Word-Embedding Association Test (*WEAT*) (Caliskan et al. 2017). *WEAT* was inspired by the Implicit Association Test (*IAT*), which measured the response time of individuals who were asked to perform a word pairing test: pairing two words they found similar and two words they found different. *WEAT* uses the cosine distance between pairs of words "as analogous to reaction time in *IAT*" (Caliskan et al. 2017). Word pair lists for *WEAT* can be defined by the user, but standard sets of words are provided with the toolkit for measuring several types of bias. In the first two tests, a single adjustment was made to a set of terms and the resulting model was evaluated. In the third experiment, we combined the two strategies before measuring the effects. We applied identical strategies to all four models. The resulting *WEAT* scores were compared to the hard debiasing strategy described in (Bolukbasi et al. 2016).

WEAT scores				
Corpus	1. pronouns	2. personal names	A combination of 1 and 2	Hard debiasing using *debiaswe*
1987-1991	0.68	**0.46**	>0.49	0.48
1992-1996	0.94	0.89	0.76	**0.57**
1997-2001	0.86	0.79	0.71	**0.6**
2002-2006	0.65	**0.34**	0.51	0.5

Table 2: WEN refitting strategies versus application of hard debiasing. Best results are in bold.

The first two experiments were inspired by research into the causes of gender bias and the occurrence of gender stereotypes in word embeddings: the effect of personal pronouns on gender bias (Atir et al. 2018), and the role played by the common usage of personal names (Johns et al. 2019), specifically first names with respect to gender and science.

Experiment 1: the "personal pronouns" strategy is based on the notion that personal pronouns may play a significant role in gender bias that is learned by word embeddings. We perform a search for science, and then perform a refitting with "she, her, hers." We performed the same task for each of the four embedding models and then computed the *WEAT* "science and art" bias score for each.

Experiment 2: the "personal names" strategy. This entailed first performing a search for female personal names in each embedding model. We searched for "mary" which is the most common female name from the last 100 years per the US Social Security Administration [10]. We then selected all entries in this result set and refitted them with the terms "science" and "scientist" and again evaluated the results using *WEAT*.

In experiment 3 we combined strategies 1 and 2 in order on each embedding model, and then measured the results. Table 2 summarizes the results of the experiments.

For comparison, we used the Word Embeddings Fairness Framework (*WEFE*) [11], for debiasing and gender bias evaluation. The WEFE python library is to word embeddings and bias, what gensim is to NLP. It implements a variety of common debiasing strategies and metrics proposed in literature, thus collecting many state of the art approaches to these problems under a single umbrella. We selected *WEFE's* hard debiasing strategy for comparison to the results of our refitting experiments.

Using it is straightforward: we first load each of the NYT embedding models as *gensim* KeyedVectors and then apply the hard debiasing strategy as illustrated by this code excerpt:

```
debiaswe_wordsets = fetch_debiaswe()

definitional_pairs = debiaswe_wordsets["definitional_pairs"]
equalize_pairs = debiaswe_wordsets["equalize_pairs"]
gender_specific = debiaswe_wordsets["gender_specific"]

hd = HardDebias(verbose=False, criterion_name="gender").fit(
    word2vec_model,
    definitional_pairs=definitional_pairs,
    equalize_pairs=equalize_pairs,
)

```

```
gender_debiased_model = hd.transform(
    word2vec_model, ignore=gender_specific, copy=True
)
```

Three data sets are used by *debiaswe* hard debiasing to reduce gender bias. The definitional pairs data set includes pairs of words such as “woman” and “man” and “girl” and “boy.” The equalized pairs set is more role and career oriented, and includes pairs such as “king” and “queen” and “brother” and “sister.” The gender specific list appears to be learned from the original corpus and includes many additional social roles, biological gender specific topics such as “obstetrics” and “prostate cancer,” and many personal names. The first set is used to identify what the authors refer to as a “bias subspace.” This is used to determine how to neutralize bias. Words found in the gender specific list are omitted from bias neutralization. The remaining terms are adjusted by removing the bias direction identified from the bias subspace. Finally, equalized pairs are adjusted again so that they are equidistant from the vector representing the bias direction.

Results and Conclusion

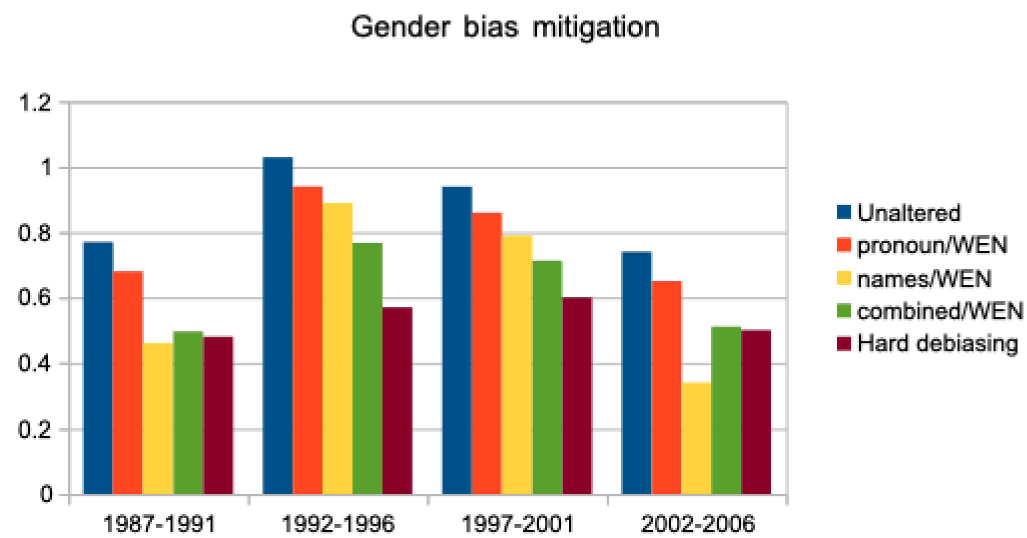


Figure 9. Plot which comparing the results of hard debiasing and several strategies applied using refitting in WEN on four temporal instances of the NYT corpus.

Prior to any debiasing efforts, Table 1 illustrates the gender bias scores for several areas of bias commonly found in text corpora: career and family, math and arts, and science and arts. The bias score from the *WEAT* metric can range from 2.0 to -2.0, where positive values indicate male gender bias, and negative values represent bias toward women. A score of zero suggests that gender bias does not exist per the test suite, or has been neutralized. We specifically focused on test cases related to “science” and “arts”, as the two topics seem distinct and queries for the two terms showed there was no overlap between them among their 200 nearest terms. Strategy 1 yielded only modest improvements, in the presence of gender bias as measured by the *WEAT* metric, moving the proverbially gender needle slightly away from male gender bias by between 8 and 12%. The personal names strategy was more successful, but varied dramatically from model to model, with a 13.5% shift for the 1992-1996 model, but a much more substantial 54% reduction in male gender bias for 2002-2006. Combining the two refitting strategies yielded consistently good results scoring which were consistently closer to the hard debiasing results than any other approach. Meanwhile, hard debiasing achieved better results for two models, and the improvements it made to the models were in general more consistent. Figure 9 illustrates the before and after impact of each strategy.

Future work

There are several areas ripe for future exploration with *WEN*. A follow up paper inspired by the concept of retrofitting entitled “Counter-fitting word vectors to linguistic constraints” (Mrkšić et al. 2016) proposed a post-processing technique to introduce other information from lexical dictionaries. Notably this technique would identify antonym relationships and adjust word vectors accordingly. This could be readily adapted to *WEN*. It would be interesting to see how users would apply counterfeiting in a word embedding model. Would they focus strictly on antonym relationships or would they take advantage of the feature to reduce the proximity of terms for other reasons?

A full implementation of the capabilities enabled by refit logging would be another area of exploration. Questions such as how to select refitting actions for rollback, reversing a sequence of refitting tasks, and investigating the impact of selective rollbacks would be possible. Related to this would be expanding the somewhat neglected temporal capabilities of *WEN*. Would refitting across temporally aligned models (diachronic refitting) be a desirable feature?

More generally, *WEN* would benefit from additional methods of visualizing word embedding vectors. For example, the plotly python library supports a large number of visualizations and its performance is quite good. Related to this, it would be interesting to further explore what kinds of visualizations would best facilitate diachronic word analysis and latent knowledge discovery across temporally aligned word embeddings. Although exploration of temporal word embeddings inspired *WEN*, it lacks sufficient features to really explore words along a temporal dimension.

Finally, integrating bias metrics would be a highly desirable feature. But as with most debiasing-related approaches, these are designed to be run as post-processing tasks. Thus performance would likely be an issue, as would determining which metric(s) to implement. Recent research raises questions about the effectiveness of WEAT in mitigating bias, for example. Some debiasing strategies fail to actually remove bias, even though they result in improved bias scores. This raises several questions, such as does *WEN* do better or worse at debiasing? Are there other bias metrics that more accurately measure bias? Are bias metrics suitable for integration with an interactive application? These are questions that could be explored in a future iteration of *WEN*.

WEN is primarily intended to be used with smaller, locally trained embedding models such as those based on the contents in an institutional repository, documents from a specific genre, or temporal embeddings where distinct models represent individual time slices of the original corpus. It allows users to test out different potential strategies to make adjustments to word embedding models, whether to neutralize bias, or to otherwise compensate for limitations in the source data upon which the model was trained. Locally trained models may have unique forms of bias that may not be effectively addressed by post-processing tools such as *WEFE*'s hard debiasing. We believe that *WEN* fills a niche in NLP / machine learning pipelines not adequately addressed by any existing tools.

References

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

Pennington, J., Socher, R. and Manning, C.D., 2014, October. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Whitaker, B., Newman-Griffis, D., Haldar, A., Ferhatosmanoglu, H. and Fosler-Lussier, E., 2019. Characterizing the impact of geometric properties of word embeddings on task performance. arXiv preprint arXiv:1904.04866.

Tsakalidis, A., Basile, P., Bazzi, M., Cucuringu, M. and McGillivray, B., 2021. DUKweb, diachronic word representations from the UK Web Archive corpus. Scientific Data, 8(1), pp.1-12.

Bailey, A.H., Williams, A. and Cimpian, A., 2022. Based on billions of words on the internet, people= men. Science Advances, 8(13), p.eabm2463.

Bolukbasi, T., Chang, K.W., Zou, J.Y., Saligrama, V. and Kalai, A.T., 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. Advances in neural information processing systems, 29.

Brunet, M.E., Alkalay-Houlihan, C., Anderson, A. and Zemel, R., 2019, May. Understanding the origins of bias in word embeddings. In International conference on machine learning (pp. 803-811). PMLR.

Savoldi, B., Gaido, M., Bentivogli, L., Negri, M. and Turchi, M., 2021. Gender bias in machine translation. Transactions of the Association for Computational Linguistics, 9, pp.845-874.

Garg, N., Schiebinger, L., Jurafsky, D. and Zou, J., 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. Proceedings of the National Academy of Sciences, 115(16), pp.E3635-E3644.

Padilla, T., Allen, L., Frost, H., Potvin, S., Russey Roke, E. and Varner, S., 2019. Always already computational: collections as data. Texas Digital Library doi:10.5281/ZENODO.3152934

Park, D., Kim, S., Lee, J., Choo, J., Diakopoulos, N. and Elmqvist, N., 2017. Conceptvector: text visual analytics via interactive lexicon building using word embedding. IEEE transactions on visualization and computer graphics, 24(1), pp.361-370.

Smilkov, D., Thorat, N., Nicholson, C., Reif, E., Viégas, F.B. and Wattenberg, M., 2016. Embedding projector: Interactive visualization and interpretation of embeddings. arXiv preprint arXiv:1611.05469.

Katricheva, N., Yaskevich, A., Lisitsina, A., Zhordaniya, T., Kutuzov, A. and Kuzmenko, E., 2019, July. Vec2graph: A python library for visualizing word embeddings as graphs. In International Conference on Analysis of Images, Social Networks and Texts (pp. 190-198). Springer, Cham.

Ghai, B., Hoque, M.N. and Mueller, K., 2021, May. WordBias: An Interactive Visual Tool for Discovering Intersectional Biases Encoded in Word Embeddings. In Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems (pp. 1-7).

Sentz, K., Powell, J., Skurikhin, A., Porter, R. 2019. Searching for ConText: Microtasking to Solve Computationally Unsolvable Problems. LDRD Reserve Final Report LA-UR-19-30118.

Tshitoyan, V., Dagdelen, J., Weston, L., Dunn, A., Rong, Z., Kononova, O., Persson, K.A., Ceder, G. and Jain, A., 2019. Unsupervised word embeddings capture latent knowledge from materials science literature. Nature, 571(7763), pp.95-98.

Levy, O. and Goldberg, Y., 2014, June. Linguistic regularities in sparse and explicit word representations. In Proceedings of the eighteenth conference on computational natural language learning (pp. 171-180).

Van der Maaten, L. and Hinton, G., 2008. Visualizing data using t-SNE. Journal of machine learning research, 9(11).

Bertrand, M., Chugh, D. and Mullainathan, S., 2005. Implicit discrimination. American Economic Review, 95(2), pp.94-98.

Faruqui, M., Dodge, J., Jauhar, S.K., Dyer, C., Hovy, E. and Smith, N.A., 2014. Retrofitting word vectors to semantic lexicons. arXiv preprint arXiv:1411.4166.

Powell, J. and Sentz, K., 2020. Interactive Re-Fitting as a Technique for Improving Word Embeddings. arXiv preprint arXiv:2010.00121.

Caliskan, A., Bryson, J.J. and Narayanan, A., 2017. Semantics derived automatically from language corpora contain human-like biases. Science, 356(6334), pp.183-186.

Greenwald, A.G., McGhee, D.E. and Schwartz, J.L., 1998. Measuring individual differences in implicit cognition: the implicit association test. Journal of personality and social psychology, 74(6), p.1464.

Atir, S. and Ferguson, M.J., 2018. How gender determines the way we speak about professionals. Proceedings of the National Academy of Sciences, 115(28), pp.7278-7283.

Johns, B.T. and Dye, M., 2019. Gender bias at scale: Evidence from the usage of personal names. Behavior research methods, 51(4), pp.1601-1618.

Mrkšić, N., Séaghdha, D.O., Thomson, B., Gaši?, M., Rojas-Barahona, L., Su, P.H., Vandyke, D., Wen, T.H. and Young, S., 2016. Counter-fitting word vectors to linguistic constraints. arXiv preprint arXiv:1603.00892.

Jo, H. and Choi, S.J., 2018. Extrofitting: Enriching word representation and its vector space with semantic lexicons. arXiv preprint arXiv:1804.07946.

Gonen, H. and Goldberg, Y., 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. arXiv preprint arXiv:1903.03862.

End Notes

[1] Common Crawl. See https://commoncrawl.org/

[2] "Teaching Method: Explanation" A discussion of explanation as a teaching method. See https://en.wikipedia.org/wiki/Teaching_method#Explanation

[3] gensim "Topic Modeling for Humans" software library. See https://radimrehurek.com/gensim/

[4] plot.ly visualization library. See https://plotly.com/

[5] d3 Network graph gallery. See https://d3-graph-gallery.com/network.html

[6] "Project Implicit." See https://implicit.harvard.edu/implicit/takeatest.html

[7] "Test Yourself for Hidden Bias." See https://www.learningforjustice.org/professional-development/test-yourself-for-hidden-bias

[8] WordNet. See https://wordnet.princeton.edu/

[9] The New York Times annotated corpus. See https://doi.org/10.35111/77ba-9×74

[10] "Top Names Over the Last 100 Years" from the US Social Security Administration. See https://www.ssa.gov/oact/babynames/decades/century.html

[11] Documentation for "The Word Embeddings Fairness Evaluation Framework." See https://wefe.readthedocs.io/en/latest/index.html

About the Authors

James Powell (jepowell@lanl.gov) is a Research Engineer and a member of the Digital Library Research and Prototyping Team at the Research Library at Los Alamos National Laboratory, where he has worked for 17 years. Previously he worked at the University Libraries at Virginia Tech. His latest book is A Librarian's Guide to Graphs, Data and the Semantic Web. Chandos Information Professional Series. Waltham, MA: Chandos, 2015.

Kari Sentz (ksentz@lanl.gov) is currently a scientist in the Information Sciences Group (CCS-3) at Los Alamos National Laboratory. She has a Ph. D. is in Systems Science from Binghamton University and a Master's in Linguistics from the University of Virginia. Sentz has worked at both Los Alamos and Sandia National Laboratories since 2000 researching generalized probability, text mining, risk analysis, probabilistic graphical modeling, and data and information fusion.

Elizabeth Moyer (emoyer@lanl.gov) is a librarian and member of the Reference and Research Services team at the Los Alamos National Laboratory Research Library. In this role, she helps support researchers by providing research support and is the Physics Liaison. She has an interest in the social bias of artificial intelligence through directed research conducted while she was a student at the University of British Columbia School of Information under the direction of Dr. Richard Arias-Hernández. She has continued interest and supports research in this domain for researchers at LANL through the development of a resource guide. Elizabeth holds a Masters of Library and Information Studies (MLIS) from the University of British Columbia School of Information in Vancouver, British Columbia, Canada.

Martin Klein (mklein@lanl.gov), Los Alamos National Laboratory, is a scientist and lead of the Prototyping Team in LANL's Research Library. In this role, he focuses on research and development efforts in the realm of web archiving, scholarly communication, digital system interoperability, and data management. He is involved in standards and frameworks such as Memento, ResourceSync, Signposting, and Robust Links. Martin holds a Diploma in Computer Science from the University of Applied Sciences Berlin, Germany, and a Ph.D. in Computer Science from Old Dominion University.

Subscribe to comments: For this article | For all articles