Text Data in Business and Economics

Basel University – Autumn 2024

6. Word Embeddings Without Neural Nets

What have we been doing? ***Learning representations*** of the data

- Dictionary methods: document is represented as a count over the lexicon
- N-grams: document is a count over a vocabulary of phrases
- Topic models: document is a vector of shares over topics
- Text classifiers: produces $\hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i; \hat{\theta})$, a vector of predicted probabilities across classes for each document i.
  - This vector of class probabilities is a **compressed** **representation** of the outcome-predictive text features $\boldsymbol{x}_i$
  - the vector of features, $\boldsymbol{x}_i$, is itself a compressed representation of the unprocessed document $\mathcal{D}_i$.
- For topic models or classifiers: the learned parameters $\hat{\theta}$ can also be understood as a **learned compressed representation of the whole dataset**:
  - it contains information about the training corpus, the text features, and the outcomes.

Information in $\hat{\theta}$: Preview of Word Embeddings

θ = matrix of parameters learned from logit, relating words to outcomes.

- If $\boldsymbol{x}$ is a bag-of-words representation for a document consisting of a list of tokens $\{w_1, ..., w_t, ..., w_n\}$, we can write

$$\boldsymbol{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$$

  - where x_t is an n_x-dimensional one-hot vector – all entries are zero except equals one for the word at t.

Information in $\hat{\theta}$: Preview of Word Embeddings

$\theta =$ matrix of parameters learned from logit, relating words to outcomes.

- If $\boldsymbol{x}$ is a bag-of-words representation for a document consisting of a list of tokens $\{w_1, ..., w_t, ..., w_n\}$, we can write

$$\boldsymbol{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$$

  - where x_t is an n_x-dimensional one-hot vector – all entries are zero except equals one for the word at t.
- Let θ_t be the row of θ corresponding to the word w_t: a **word embedding** for w_t containing the outcome-relevant information for that word.

Information in $\hat{\theta}$: Preview of Word Embeddings

$\theta =$ matrix of parameters learned from logit, relating words to outcomes.

- If $\boldsymbol{x}$ is a bag-of-words representation for a document consisting of a list of tokens $\{w_1, ..., w_t, ..., w_n\}$, we can write

$$\boldsymbol{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$$

  - where x_t is an n_x-dimensional one-hot vector – all entries are zero except equals one for the word at t.
- Let θ_t be the row of θ corresponding to the word w_t: a **word embedding** for w_t containing the outcome-relevant information for that word.
- We can construct a **document vector**

$$\vec{\boldsymbol{d}} = \frac{1}{n}\sum_{t=1}^{n_i} \theta_t$$

  the sum of the n_y-dimensional word representations (the row vectors from above).

  - this is called the "continuous bag of words (CBOW)" representation (Goldberg 2017).
  - Note that $\vec{\boldsymbol{d}} = \theta \cdot \boldsymbol{x}$, and thus θ is a **word embedding matrix**.

Outline

Words and Local Contexts

How do I use Word Embeddings?

Bias in Language

Word Embeddings

- "Word embeddings" often refer to Word2Vec or GloVe – these are particular (popular) models for producing word embeddings.
  - **Previously:** focus on global document counts or predict an outcome
  - **Now**: represent the meaning of words by the neighboring words – their **local contexts**.

Word Embeddings

- "Word embeddings" often refer to Word2Vec or GloVe – these are particular (popular) models for producing word embeddings.
  - **Previously:** focus on global document counts or predict an outcome
  - **Now**: represent the meaning of words by the neighboring words – their **local contexts**.
    - $\rightarrow$ rather than predicting some metadata, they predict the co-occurence of neighboring words.

Word Embeddings

- "Word embeddings" often refer to Word2Vec or GloVe – these are particular (popular) models for producing word embeddings.
  - **Previously:** focus on global document counts or predict an outcome
  - **Now**: represent the meaning of words by the neighboring words – their **local contexts**.
    - → rather than predicting some metadata, they predict the co-occurence of neighboring words.
- From high-dimensional sparse representations to low-dimensional dense representations

Words and Contexts

A long line of NLP research aims to capture the distributional properties of words using a **word-context matrix** $\boldsymbol{M}$:

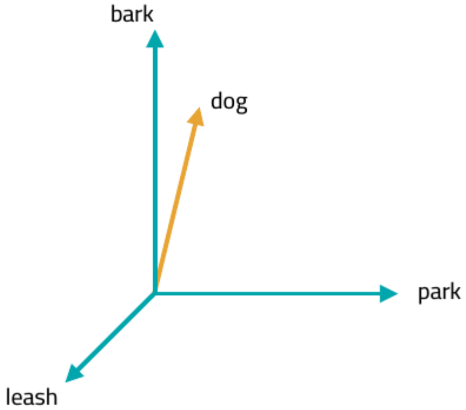
- each row w represents a **word** (e.g. "income"), each column c represents a linguistic **context** in which words can occur (e.g. "... pay corporate income ___ to the relevant ...").
  - A matrix entry $\boldsymbol{M}_{[w,c]}$ quantifies the strength of association between a word and a context in a large corpus.
  - Popular embeddings (word2vec and glove) generally use 5- or 10-word windows as the context.

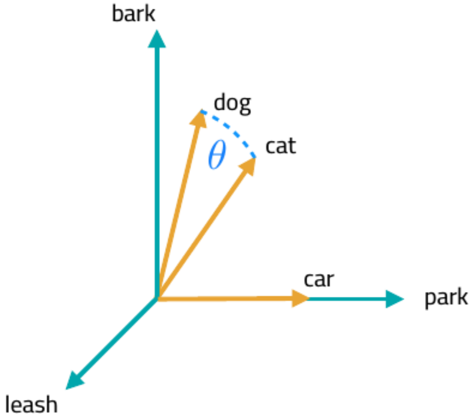
Words and Contexts

A long line of NLP research aims to capture the distributional properties of words using a **word-context matrix** $\boldsymbol{M}$:

- each row w represents a **word** (e.g. "income"), each column c represents a linguistic **context** in which words can occur (e.g. "... pay corporate income ___ to the relevant ...").
  - A matrix entry $\boldsymbol{M}_{[w,c]}$ quantifies the strength of association between a word and a context in a large corpus.
  - Popular embeddings (word2vec and glove) generally use 5- or 10-word windows as the context.
- each word (row) $\boldsymbol{M}_{[w,:]}$ gives a distribution over contexts.
  - different definitions of contexts and different measures of association $\rightarrow$ different types of **word vectors**.
  - these vectors have a **spatial interpretation** $\rightarrow$ geometric distances between word vectors reflect semantic distances between words.

Word Embeddings in the 2-Dimensional Space





Word Similarity

- Once words are represented as vectors $\{v_1 = \boldsymbol{M}_{[w_1,:]},\ v_2 = \boldsymbol{M}_{[w_2,:]}, \ldots\}$, we can use linear algebra to understand the relationships between words:
  - Words that are geometrically close to each other are similar: e.g. "dog" and "cat":
- The standard metric for comparing vectors is cosine similarity:

$$\cos\theta = \frac{v_1 \cdot v_2}{||v_1||||v_2||}$$

  - alternatives include e.g. Jaccard similarity (Goldberg 2017)

Word Similarity

- Once words are represented as vectors $\{v_1 = \boldsymbol{M}_{[w_1,:]},\ v_2 = \boldsymbol{M}_{[w_2,:]}, \ldots\}$, we can use linear algebra to understand the relationships between words:
  - Words that are geometrically close to each other are similar: e.g. "dog" and "cat":
- The standard metric for comparing vectors is cosine similarity:

$$\cos\theta = \frac{v_1 \cdot v_2}{||v_1||||v_2||}$$

  - alternatives include e.g. Jaccard similarity (Goldberg 2017)
- Thanks to linearity, can compute similarities between groups of words by averaging the groups.

Word Vectors can produce Document Vectors

$$\vec{D} = \sum_{w \in D} a_w \vec{w}$$

- The "continuous bag of words" representation for document D is the sum, or the average (potentially weighted by a_w), of the vectors $\vec{w}$ for each word w in the document.
  - word vectors $\vec{w}$ constructed using Word2Vec or GloVe (pre-trained or trained on the corpus).
  - "Document" could be sentence, paragraph, section, etc.

Word Vectors can produce Document Vectors

$$\vec{D} = \sum_{w \in D} a_w \vec{w}$$

- The "continuous bag of words" representation for document D is the sum, or the average (potentially weighted by a_w), of the vectors $\vec{w}$ for each word w in the document.
  - word vectors $\vec{w}$ constructed using Word2Vec or GloVe (pre-trained or trained on the corpus).
  - "Document" could be sentence, paragraph, section, etc.
- Arora, Liang, and Ma (2017) provide a "tough to beat baseline", the SIF-weighted ("smoothed inverse frequency") average of the vectors:

  $$a_w = \frac{\alpha}{\alpha + p_w}$$

  where p_w is the probability (frequency) of the word and $\alpha = .001$ is a smoothing parameter.

What do Word Embeddings Capture? (Budansky and Hirst, 2006)

- Semantic **similarity**: words sharing salient attributes / features
  - synonymy (car / automobile)
  - hypernymy (car / vehicle)
  - co-hyponymy (car / van / truck)

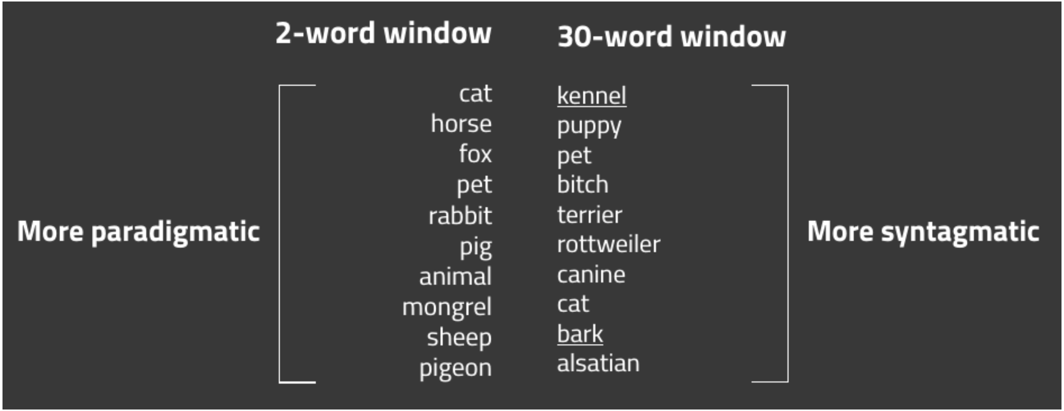
What do Word Embeddings Capture? (Budansky and Hirst, 2006)

- Semantic **similarity**: words sharing salient attributes / features
  - synonymy (car / automobile)
  - hypernymy (car / vehicle)
  - co-hyponymy (car / van / truck)
- Semantic **relatedness**: words semantically associated without necessarily being similar
  - function (car / drive)
  - meronymy (car / tire)
  - location (car / road)
  - attribute (car / fast)

What do Word Embeddings Capture? (Budansky and Hirst, 2006)

- Semantic **similarity**: words sharing salient attributes / features
  - synonymy (car / automobile)
  - hypernymy (car / vehicle)
  - co-hyponymy (car / van / truck)
- Semantic **relatedness**: words semantically associated without necessarily being similar
  - function (car / drive)
  - meronymy (car / tire)
  - location (car / road)
  - attribute (car / fast)
- Word embeddings will recover one or both of these relations, depending on how contexts and associated are constructed.

Most similar words to dog, depending on context window size



- Small windows pick up substitutable words; large windows pick up topics.

Parts of Speech and Phrases

- In the default model multiple senses of a word are merged.
  - e.g. "I like a bird" (verb) and "I am like a bird" (preposition).

Parts of Speech and Phrases

- In the default model multiple senses of a word are merged.
  - e.g. "I like a bird" (verb) and "I am like a bird" (preposition).
- Can improve the quality of embeddings in these cases by attaching the POS to the word (e.g. "like:verb", "like:prep") before training.

Parts of Speech and Phrases

- In the default model multiple senses of a word are merged.
  - e.g. "I like a bird" (verb) and "I am like a bird" (preposition).
- Can improve the quality of embeddings in these cases by attaching the POS to the word (e.g. "like:verb", "like:prep") before training.
- The default model only works by word, but "new york $\neq$ "new" + "york"
  - can tokenize phrases together (see Week 2 lecture) before training.

The black sheep problem

- The trivial or obvious features of a word are not mentioned in standard corpora.

The black sheep problem

- The trivial or obvious features of a word are not mentioned in standard corpora.
- For example, although most sheep are white, you rarely see the phrase "white sheep".
  - so word2vec tells you sim(black,sheep) $>$ sim(white,sheep).

The black sheep problem

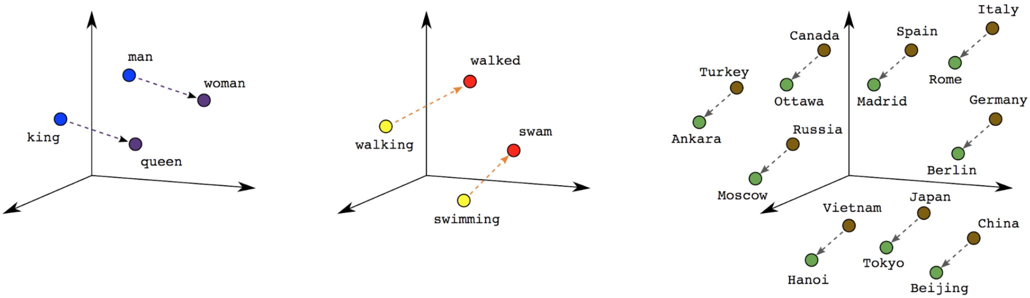
- The trivial or obvious features of a word are not mentioned in standard corpora.
- For example, although most sheep are white, you rarely see the phrase "white sheep".
  - so word2vec tells you sim(black,sheep) $>$ sim(white,sheep).
- This is really important when we will use embeddings to anayze beliefs/attitudes.
  - And I don't see a solution to it.

The black sheep problem

- The trivial or obvious features of a word are not mentioned in standard corpora.
- For example, although most sheep are white, you rarely see the phrase "white sheep".
  - so word2vec tells you sim(black,sheep) $>$ sim(white,sheep).
- This is really important when we will use embeddings to anayze beliefs/attitudes.
  - And I don't see a solution to it.
- Relatedly, antonyms are often rated similarly, have to be careful with that.

Vector Directions $\leftrightarrow$ Meaning

- Intriguingly, word2vec algebra can depict conceptual, analogical relationships between words:



Male-Female Verb Tense Country-Capital

Word Embeddings for Analogies

$$vec(king) - vec(man) + vec(woman) \approx vec(queen)$$

Word Embeddings for Analogies

$$vec(king) - vec(man) + vec(woman) \approx vec(queen)$$

- More generally: The analogy $a_1 : b_1 :: a_2 : b_2$ can be solved (that is, find b_2 given a_1, b_1, a_2) by

$$\arg\max_{b_2 \in V} \cos(b_2, a_2 - a_1 + b_1)$$

  where V excludes (a_1, b_1, a_2).

Word Embeddings for Analogies

$$vec(king) - vec(man) + vec(woman) \approx vec(queen)$$

- More generally: The analogy $a_1 : b_1 :: a_2 : b_2$ can be solved (that is, find b_2 given a_1, b_1, a_2) by
$$\arg\max_{b_2 \in V} \cos(b_2, a_2 - a_1 + b_1)$$
where V excludes (a_1, b_1, a_2).
- Often works better with normalized vectors (so that one long vector doesn't wash out the others)

Word Embeddings for Analogies

$$vec(king) - vec(man) + vec(woman) \approx vec(queen)$$

- More generally: The analogy $a_1 : b_1 :: a_2 : b_2$ can be solved (that is, find b_2 given a_1, b_1, a_2) by
$$\arg\max_{b_2 \in V} \cos(b_2, a_2 - a_1 + b_1)$$
where V excludes (a_1, b_1, a_2).
- Often works better with normalized vectors (so that one long vector doesn't wash out the others)
- Levy and Goldberg (2014) recommend the following "CosMul" metric which tends to perform better:
$$\arg\max_{b_2 \in V} \frac{\cos(b_2, a_2)\cos(b_2, b_1)}{\cos(b_2, a_1) + \epsilon}$$
  - requires normalized, non-negative vectors (can transform using $(x+1)/2$)
  - ϵ is a small smoothing parameter.

Outline

Tokenizing for Word Embeddings

- Need to think about what's the right way to pre-process data
  - drop capitalization
  - punctuation is optional
  - don't drop stopwords/function-words
  - add special tokens for start of sentence and end of sentence
  - for out-of-vocab words, substitute a special token or replace with part-of-speech tag

Which word-embedding?

- In many settings (e.g. a small corpus), better to use pre-trained embeddings.
- e,g, spaCy's GloVe embeddings:
  - one million vocabulary entries
  - 300-dimensional vectors
  - trained on the Common Crawl corpus
- Can initialize models with pre-trained embeddings, can fine-tune as needed.

“Enriching word vectors with subword information” (Bojanowski et al 2017)

- each word is represented as a bag of (hashed) character n-grams. (e.g., spicy = (spi, pic, icy)).
- learn embeddings for the character segments, and construct word embedding by summing over the segment embeddings

“Enriching word vectors with subword information” (Bojanowski et al 2017)

- each word is represented as a bag of (hashed) character n-grams. (e.g., spicy = (spi, pic, icy)).
- learn embeddings for the character segments, and construct word embedding by summing over the segment embeddings
- competitive with word2vec in standard tasks; better in some languages.
- produces good embeddings for unseen words.

Limitations

Standard word embeddings (e.g. word2vec/glove) have a number of limitations:

- **polysemy**: you get one vector for multiple senses of a word
  (e.g. "**glass** of water" vs "window **glass**")

Limitations

Standard word embeddings (e.g. word2vec/glove) have a number of limitations:

- **polysemy**: you get one vector for multiple senses of a word (e.g. "**glass** of water" vs "window **glass**")
- **rare words**: a word that shows up just once or twice won't be well-defined
- **n-grams**: does not produce embeddings for multi-word phrases

Scientists attending ACL work on **cutting edge** research in NLP

Petrichor: the earthy scent produce when rain falls on dry soil

Roger Federer won the first **set**^{NN} of the match

Limitations

Standard word embeddings (e.g. word2vec/glove) have a number of limitations:

- **polysemy**: you get one vector for multiple senses of a word (e.g. "**glass** of water" vs "window **glass**")
- **rare words**: a word that shows up just once or twice won't be well-defined
- **n-grams**: does not produce embeddings for multi-word phrases

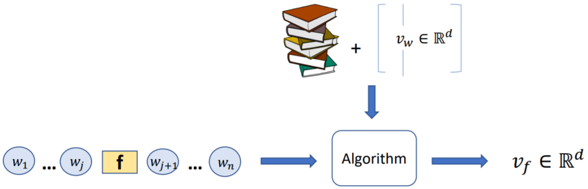
Scientists attending ACL work on **cutting edge** research in NLP

Petrichor: the earthy scent produce when rain falls on dry soil

Roger Federer won the first **set**^{NN} of the match

- Goal of Khodak et al (2018): produce embeddings "a la carte" given a context:

Given: Text corpus and high quality word embeddings trained on it



Input: A feature in context(s)

Output: Good quality embedding

A la carte embeddings

- Given a target word f and its context c, define

$$v_f^{avg} = \frac{1}{|c|} \sum_{w \in c} v_w$$

  the average vector for the words in the context.
- Arora et al (2018) prove that for vectors produced by a generative language model, there exists a matrix A such that

$$v_f \approx A v_f^{avg}$$

A la carte embeddings

- Given a target word f and its context c, define

$$v_f^{avg} = \frac{1}{|c|} \sum_{w \in c} v_w$$

  the average vector for the words in the context.
- Arora et al (2018) prove that for vectors produced by a generative language model, there exists a matrix A such that

$$v_f \approx A v_f^{avg}$$

- The "induction matrix" A can be learned with a least-squares (linear regression) objective

$$A^* = \arg\min_A \sum_w |v_w - A v_w^{avg}|_2^2$$

  where w indexes over all the tokens in the corpus.
- empirically:

$$\text{cosine}(v_f, A^* v_f^{avg}) \geq 0.9$$

Outline

Implicit attitudes

"Attitudes that affect our understanding, actions, and decisions in an unconscious manner" (Kirnan institute, OSU)

Implicit attitudes

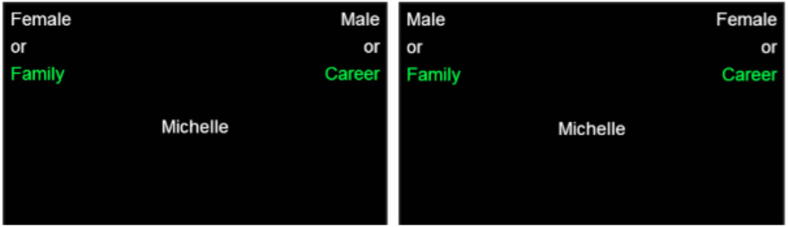
"Attitudes that affect our understanding, actions, and decisions in an unconscious manner" (Kirnan institute, OSU)

- Generally measured using Implicit Association Tests (IATs)

Implicit attitudes

"Attitudes that affect our understanding, actions, and decisions in an unconscious manner" (Kirnan institute, OSU)

- Generally measured using Implicit Association Tests (IATs)
- Subjects asked to assign words to categories (Greenwald et al. 1998)

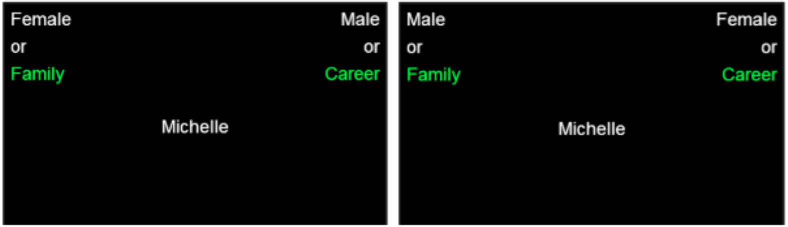


- Comparing reaction times across trials with different word pairs:
  - subjects tend to be slower and more error-prone in assignments against stereotype (e.g. "Michelle" goes to "Female or Career").

Implicit attitudes

"Attitudes that affect our understanding, actions, and decisions in an unconscious manner" (Kirnan institute, OSU)

- Generally measured using Implicit Association Tests (IATs)
- Subjects asked to assign words to categories (Greenwald et al. 1998)



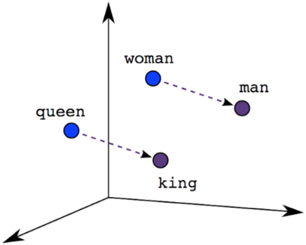
- Comparing reaction times across trials with different word pairs:
  - subjects tend to be slower and more error-prone in assignments against stereotype (e.g. "Michelle" goes to "Female or Career").
  - IAT score = difference in reaction time between stereotype-consistent and stereotype-inconsistent rounds.

Caliskan, Bryson, and Narayanan (*Science* 2017)

- "We replicated a spectrum of known biases, as measured by the Implicit Association Test, using a widely used, purely statistical machine-learning model trained on a standard corpus of text from the World Wide Web. . . "

Caliskan, Bryson, and Narayanan (*Science* 2017)

- "We replicated a spectrum of known biases, as measured by the Implicit Association Test, using a widely used, purely statistical machine-learning model trained on a standard corpus of text from the World Wide Web. . . "

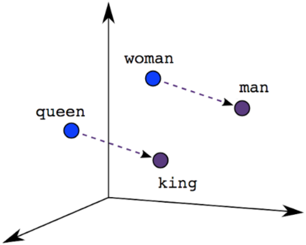


Analogies

- king : queen :: man : woman
- walked : walking :: swam : swimming

Caliskan, Bryson, and Narayanan (*Science* 2017)

- "We replicated a spectrum of known biases, as measured by the Implicit Association Test, using a widely used, purely statistical machine-learning model trained on a standard corpus of text from the World Wide Web. . . "



Analogies

- king : queen :: man : woman
- walked : walking :: swam : swimming
- **man : programmer :: woman : homemaker**
- **he : physician :: she : nurse**

Example Stimuli

- Targets:
  - **Flowers**: aster, clover, hyacinth, marigold, poppy, azalea, crocus, iris, orchid, rose, bluebell, daffodil, lilac, pansy, tulip, buttercup, daisy, lily, peony, violet, carnation, gladiola, magnolia, petunia, zinnia.
  - **Insects**: ant, caterpillar, flea, locust, spider, bedbug, centipede, fly, maggot, tarantula, bee, cockroach, gnat, mosquito, termite, beetle, cricket, hornet, moth, wasp, blackfly, dragonfly, horsefly, roach, weevil.

Example Stimuli

- Targets:
  - **Flowers**: aster, clover, hyacinth, marigold, poppy, azalea, crocus, iris, orchid, rose, bluebell, daffodil, lilac, pansy, tulip, buttercup, daisy, lily, peony, violet, carnation, gladiola, magnolia, petunia, zinnia.
  - **Insects**: ant, caterpillar, flea, locust, spider, bedbug, centipede, fly, maggot, tarantula, bee, cockroach, gnat, mosquito, termite, beetle, cricket, hornet, moth, wasp, blackfly, dragonfly, horsefly, roach, weevil.
- Attributes:
  - **Pleasant**: caress, freedom, health, love, peace, cheer, friend, heaven, loyal, pleasure, diamond, gentle, honest, lucky, rainbow, diploma, gift, honor, miracle, sunrise, family, happy, laughter, paradise, vacation.
  - **Unpleasant**: abuse, crash, filth, murder, sickness, accident, death, grief, poison, stink, assault, disaster, hatred, pollute, tragedy, divorce, jail, poverty, ugly, cancer, kill, rotten, vomit, agony, prison.

Results

- Pleasant vs. Unpleasant?
  - Flowers vs. Insects
  - Musical instruments vs. weapons.

Results

- Pleasant vs. Unpleasant?
  - Flowers vs. Insects
  - Musical instruments vs. weapons.
  - European-American names vs. African-American names

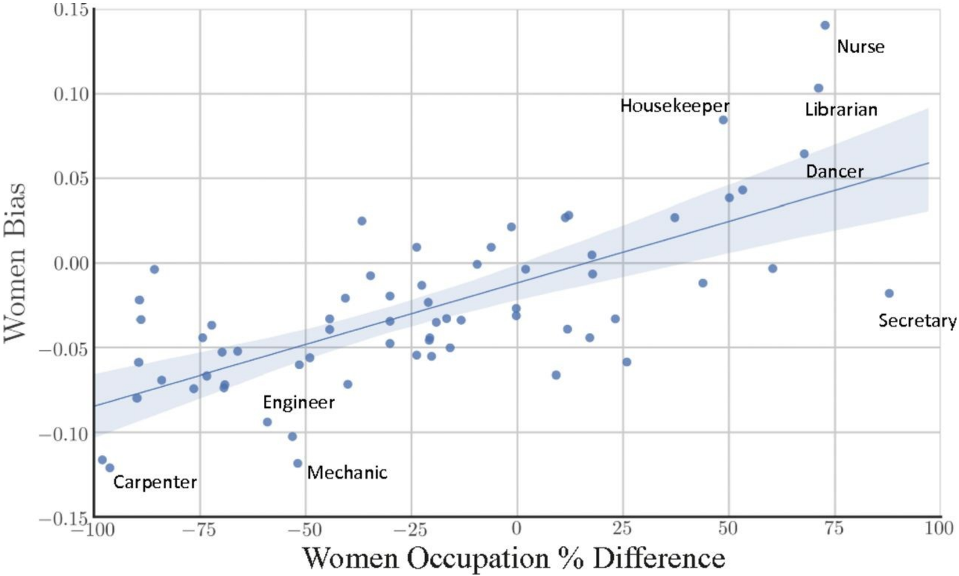
Results

- Pleasant vs. Unpleasant?
  - Flowers vs. Insects
  - Musical instruments vs. weapons.
  - European-American names vs. African-American names
- Male names vs. Female names:

Results

- Pleasant vs. Unpleasant?
  - Flowers vs. Insects
  - Musical instruments vs. weapons.
  - European-American names vs. African-American names
- Male names vs. Female names:
  - Career words (e.g. professional, corporation, ...) vs. family words (e.g. home, children, ...)
  - Math/science words vs arts words

Garg, Schiebinger, Jurafsky, and Zou (PNAS 2018)



Women's occupation relative percentage vs. embedding bias in Google News vectors.