Computational Social Science

Word embeddings I

Dr. Thomas Davidson

Rutgers University

March 7, 2024

Plan

1. Course updates
2. Word embeddings
3. Contextualized embeddings

Course updates

- Project proposals due 5pm today
  - Submit form on Canvas
- Mid-semester evaluation
  - Please complete by Friday

Recap

- Language models are probabilistic models for language understanding and generation
  - Auto-complete, search suggestions, etc.
- N-gram language models
  - Probabilistic models predicting words based on previous N words used

Language models

Neural language models

- Recent advances in both the availability of large corpora of text *and* the development of neural network models have resulted in new ways of computing language models.
- By using machine-learning to train a language model, we can construct better, more meaningful vector representations

Word embeddings

Intuition

- We use the context in which a word occurs to train a language model
  - The model learns by viewing millions of short snippets of text (e.g 5-grams)
- This model outputs a vector representation of each word in k-dimensional space, where $k << |V|$.
  - Like LSA, these vectors are *dense*
    - Each element contains a real number and can be positive or negative

Word embeddings

Word2vec: Skip-gram and continuous bag-of-words (CBOW)

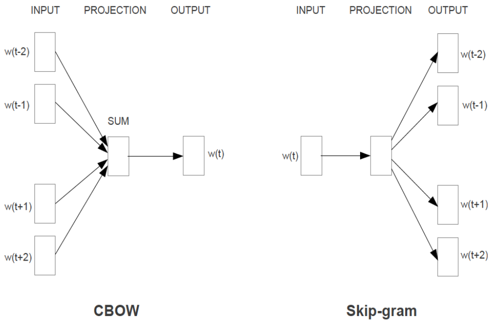


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

Word embeddings

Word2vec: CBOW intuition

- We start with a string where the focal word is known, but hidden from the model, but we know the context within a window, in this case two words on either side of the focal word
  - e.g. "The cat ? on the", where ? = "sat"
- The model is trained using a process called *negative sampling*, where it must distinguish between the true sentence and "fake" sentences where ? is replaced with another token.
  - Each "guess" allows the model to begin to learn the correct answer
- By repeating this for millions of text snippets the model is able to "learn" which words go with which contexts

Word embeddings

Word2vec: Skip-gram intuition

- We start with a string where the focal is known, but the context within the window is hidden
  - e.g. "$?_1$ $?_2$ sat $?_3$ $?_4$"
- The model tests different words in the vocabulary to predict the missing context words
  - Each "guess" allows the model to begin to learn the correct answer
- By repeating this for millions of text snippets the model is able to "learn" which contexts go with which words

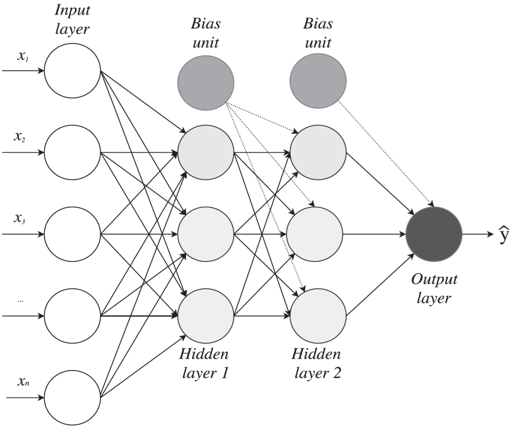
Word embeddings

Word2vec: Model

- Word2vec uses a shallow neural-network to predict a word given a context (CBOW) or a context given a word (skip-gram)
  - But we do not care about the prediction itself, only the *weights* the model learns
- It is a self-supervised method since the model is able to update using the correct answers
  - e.g. In CBOW the model knows when the prediction is wrong and updates the weights accordingly

Word embeddings

Word2vec: Feed-forward neural network



This example shows a two-layer feed-forward neural network.

Word embeddings

Word2vec: Estimation procedure

- Batches of text are passed through the network
  - After each batch, weights are updated using *back-propagation*
    - The model updates its weights in the direction of the correct answer (the objective is to improve predictive accuracy)
    - Optimization via *stochastic gradient descent*

Word embeddings

Vector representations of words

- Each word is represented as a vector of weights learned by the neural network
  - Word embeddings are *byproduct* of training a neural language model
  - Each element of this vector represents how strongly the word activates a neuron in the hidden layer of the network
  - This represents the association between the word and a given dimension in semantic space

Word embeddings

Distributional semantics

- The word vectors in the embedding space capture information about the context in which words are used
  - Words with similar meanings are situated close together in the embedding space
- *Distributional semantics* is the theory that the meaning of a word is derived from its context in language use
  - "You shall know a word by the company it keeps", linguist J.R. Firth (1957)
- This is consistent with philosopher Ludwig Wittgenstein's *use theory of meaning*
  - "the meaning of a word is its use in the language", *Philosophical Investigations* (1953)

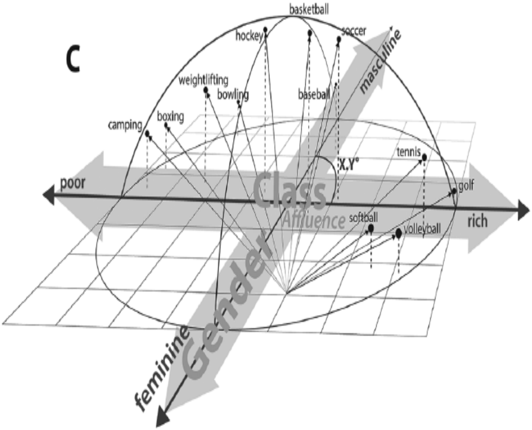
Word embeddings

Analogies

- The most famous result from the initial word embedding paper is the ability of these vectors to capture analogies:
  - $king - man + woman \approx queen$
  - $Madrid - Spain + France \approx Paris$

Word embeddings

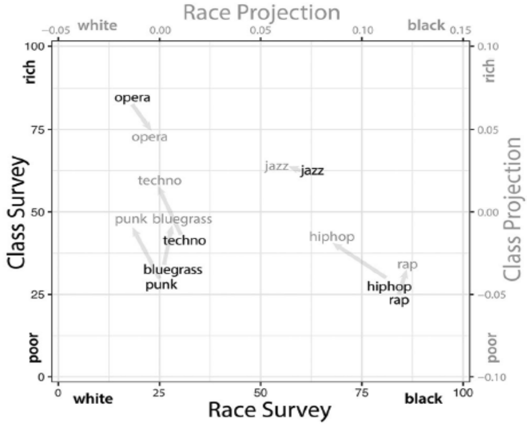
Sociological applications: Understanding social class



Kozlowski, Austin C., Matt Taddy, and James A. Evans. 2019. "The Geometry of Culture: Analyzing the Meanings of Class through Word Embeddings." American Sociological Review, September, 000312241987713.

Word embeddings

Sociological applications: Understanding social class



Word embeddings

Sociological applications: Latent dimensions

Table 4
Term pairs for immigration-citizenship cultural dimension.

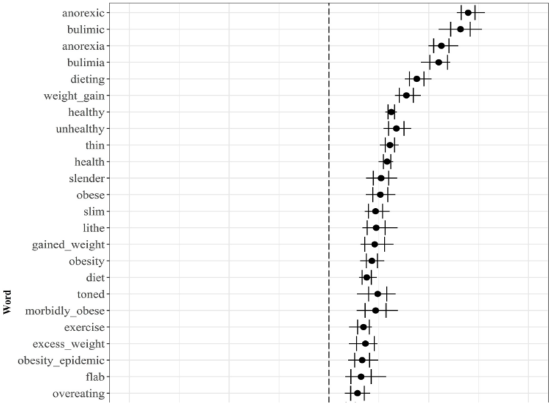
Immigrants	Citizens
Immigration	Citizenship
Immigrant	Citizen
Foreign	Domestic
Foreigner	Native
Outsider	Insider
Stranger	Local
Alien	Resident
Foreigner	Resident
Alien	Native
Immigrant	Local
Foreign	Familiar

Stoltz, Dustin S., and Marshall A. Taylor. 2021. "Cultural Cartography with Word Embeddings." *Poetics* 88 (October): 101567.

Word embeddings

Sociological applications: Understanding cultural schemas

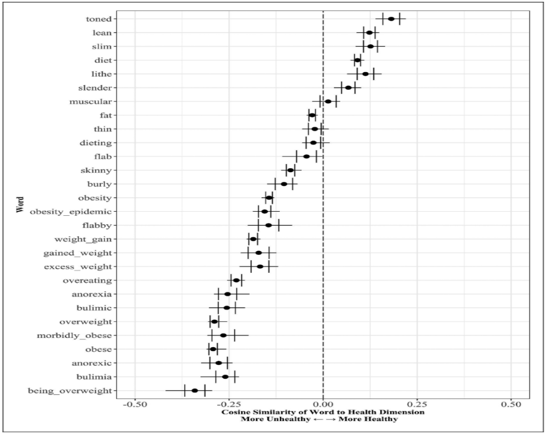
Figure 4: Gendering of Obesity-Related Words



Arseniev-Koehler, Alina, and Jacob G. Foster. 2022. "Machine Learning as a Model for Cultural Learning: Teaching an Algorithm What It Means to Be Fat." *Sociological Methods & Research* 51 (4): 1484–1539.

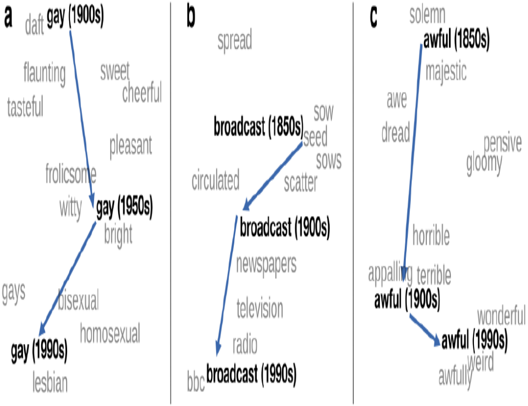
Word embeddings

Sociological applications: Understanding cultural schemas



Word embeddings

Sociological applications: Semantic change



Hamilton, William L., Jure Leskovec, and Dan Jurafsky. 2016. "Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change." In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 1489–1501.

Word embeddings

Sociological applications: Semantic change

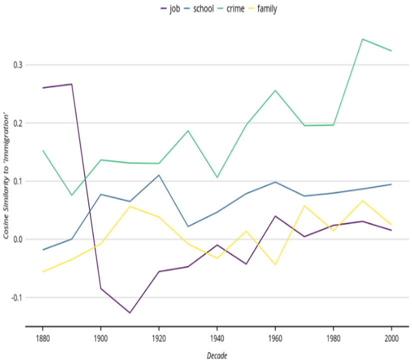
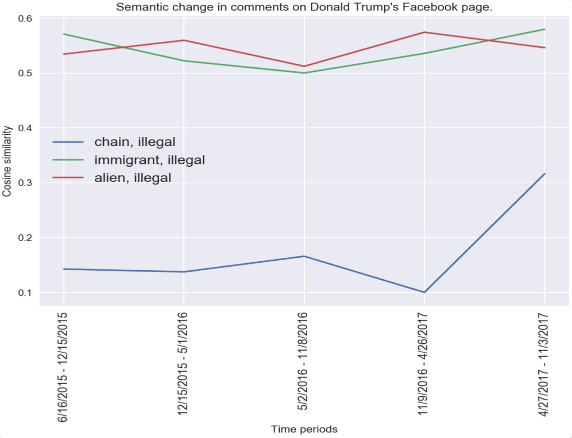


Fig. 1. Cosine Similarity of 'Immigration' and Key Terms by Decade, 1880 to 2000.

Stoltz, Dustin S., and Marshall A. Taylor. 2021. "Cultural Cartography with Word Embeddings." *Poetics* 88 (October): 101567.

Word embeddings

Sociological applications: Semantic change



Davidson 2017, unpublished.

Word embeddings

Sociological applications: Semantic change

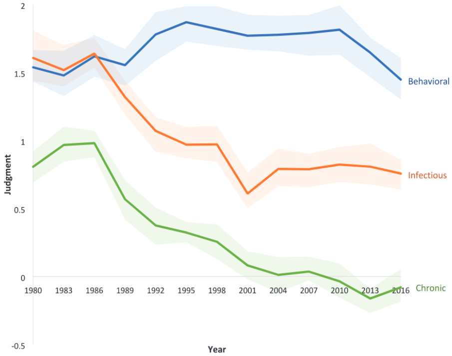


Figure 3. Judgment Scores for Behavioral, Infectious, and Chronic Diseases over Time
Note: More positive scores indicate stronger connotations of immorality and bad personality traits. More negative scores indicate stronger connotations of morality and good personality traits.

Best, Rachel Kahn, and Alina Arseniev-Koehler. 2023. "The Stigma of Diseases: Unequal Burden, Uneven Decline." *American Sociological Review* 88 (5): 938–69.

Word embeddings

Pre-trained word embeddings

- In addition to `word2vec` there are several other popular variants including `GloVe` and `Fasttext` (see Stolz and Taylor 2021)
  - Pre-trained embeddings are available to download so you don't need to train your own
- When to train your own embeddings?
  - The underlying language model / data generating process differ from that represented by existing corpora
    - e.g. A word embedding trained on newspapers may not be very useful for studying Twitter since online language use differs substantially from written news media
  - Requires large numbers of documents (> 10k)

Word embeddings

Loading a corpus: Politicians' tweets

```
library(stringr)
library(tidyverse)

df <- read_csv("data/politics_twitter.csv")
unique(df$screen_name)
```

```
##  [1] "JoeBiden"         "KamalaHarris"     "SpeakerPelosi"    "BernieSa
##  [5] "AOC"              "SenSchumer"       "LeaderMcConnell" "LindseyG
##  [9] "tedcruz"          "Mike_Pence"       "MarshaBlackburn" "lisamurk
```

Word embeddings

Word embeddings in R

We're going to use the library `word2vec`. The library is a R wrapper around a `C++` library. The the original library can be found here and the R version wrapper here.

```
#install.packages("word2vec")
library(word2vec)
set.seed(08901) # random seed
```

Word embeddings

Training a word2vec model

```
model <- word2vec::word2vec(x = tolower(df$text),
                            type="cbow",# continuous bag of words
                            dim=100, # 100-dim output vectors
                            window = 3L, # window size 3
                            iter = 10L, # train over 10 iterations
                            negative= 10L,# 10 negative samples for each positive
                            min_count = 10L) # Drop words less than 10 times
```

Word embeddings

Getting embeddings for words

We can use the `predict` function to find the nearest words to a given term.

```
predict(model, c("nation"), type = "nearest", top_n = 10)
```

```
## $nation
##     term1        term2 similarity rank
## 1  nation      country  0.9009249    1
## 2  nation      america  0.7633710    2
## 3  nation      economy  0.7069327    3
## 4  nation    democracy  0.7026614    4
## 5  nation       planet  0.7009467    5
## 6  nation    country's  0.6989810    6
## 7  nation     movement  0.6943195    7
## 8  nation      society  0.6893061    8
## 9  nation commonwealth  0.6746004    9
## 10 nation     nation's  0.6693724   10
```

Word embeddings

Testing analogical reasoning

```r
emb <- as.matrix(model) # Extracting embedding matrix
vector <- emb["king", ] - emb["man", ] + emb["woman", ]
predict(model, vector, type = "nearest", top_n = 10)
```

```
##              term similarity rank
## 1            king  0.9712596    1
## 2          taylor  0.8270658    2
## 3             rbg  0.8231881    3
## 4          martin  0.8157232    4
## 5          luther  0.8146862    5
## 6           clark  0.8134736    6
## 7          arbery  0.8056882    7
## 8  spaceforcedod  0.7973178    8
## 9              jr  0.7914040    9
## 10        breonna  0.7761958   10
```

Word embeddings

Testing analogical reasoning

```r
vector <- emb["austin", ] - emb["texas", ] + emb["illinois", ]
predict(model, vector, type = "nearest", top_n = 10)
```

```
##          term similarity rank
## 1    illinois  0.9675379    1
## 2      milley  0.9387642    2
## 3       lloyd  0.9378793    3
## 4   assistant  0.9282761    4
## 5     haaland  0.9229352    5
## 6        alex  0.9165374    6
## 7   inspector  0.8911901    7
## 8      deputy  0.8846538    8
## 9  postmaster  0.8803019    9
## 10        cop  0.8653453   10
```

Word embeddings

Loading a pre-trained embedding

Let's try another example. I downloaded a pre-trained word embedding model trained on a much larger corpus of English texts. The file is 833MB in size. Following the documentation we can load this model into R.

```
model.pt <- read.word2vec(file = "../data/sg_ns_500_10.w2v", normalize
```

Word embeddings

Similarities

Find the top 5 most similar terms to "nation" in the embedding space.

```
predict(model.pt, c("nation"), type = "nearest", top_n = 5)
```

Word embeddings

Similarities

Find the top 5 most similar terms to “immigration” in the embedding space.

```
predict(model.pt, c("immigration"), type = "nearest", top_n = 5)
```

Word embeddings

Repeating the analogy test

Let's re-try the analogy test. We still don't go great but now queen is in the top 5 results.

```
emb <- as.matrix(model.pt)
vector <- emb["king", ] - emb["man", ] + emb["woman", ]
predict(model.pt, vector, type = "nearest", top_n = 10)
```

Word embeddings

Repeating the analogy test

Let's try another analogy. The correct answer is second. Not bad.

```
vector <- emb["madrid", ] - emb["spain", ] + emb["france", ]
predict(model.pt, vector, type = "nearest", top_n = 10)
```

Word embeddings

Repeating the analogy test

Let's try another slightly more complex analogy. Not bad overall.

```
vector <- (emb["new", ] + emb["jersey", ])/2 - emb["trenton", ] + emb["
predict(model.pt, vector, type = "nearest", top_n = 10)
```

Word embeddings

Representing documents

Last week we focused on how we could represent documents using the rows in the DTM. So far we have just considered how words are represented in the embedding space. We can represent a document by averaging over its composite words.

```
descartes <- (emb["i", ] +
                emb["think", ] +
                emb["therefore", ] +
                emb["i", ] +
                emb["am", ])/5
predict(model.pt, descartes, type = "nearest", top_n = 10)
```

Word embeddings

Representing documents

The package has a function called `doc2vec` to do this automatically. This function includes an additional scaling factor (see documentation) so the results are slightly different.

```
descartes <- doc2vec(model.pt, "i think therefore i am")
predict(model.pt, descartes, type = "nearest", top_n = 10)
```

Word embeddings

Visualizing high-dimensional embeddings in low-dimensional space

- There are various algorithms available for visualizing word-embeddings in low-dimensional space
  - `PCA`, `t-SNE`, `UMAP`
- There are also browser-based interactive embedding explorers
  - See this example on the Tensorflow website

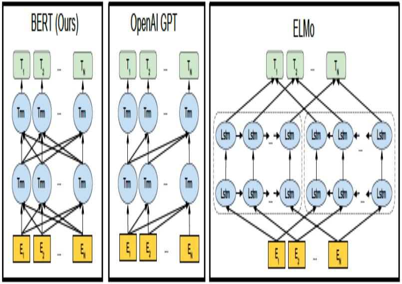
Contextualized embeddings

Limitations of existing approaches

- Word2vec and other embedding methods run into issues when dealing with *polysemy*
  - e.g. The vector for "crane" will be learned by averaging across different uses of the term
    - A bird
    - A type of construction equipment
    - Moving one's neck
  - "She had to crane her neck to see the crane perched on top of the crane".
- New methods have been developed to allow the vector for "crane" to vary according to different contexts
  - Intuition: Meaning varies depending on context

Contextualized embeddings

Architectures



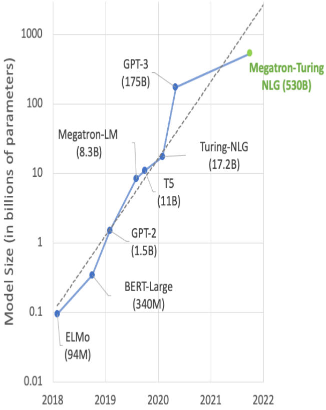
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” In Proceedings of NAACL-HLT 2019, 4171–86. ACL.

Contextualized embeddings

Methodological innovations

- More complex, deeper neural networks
  - *Attention* mechanisms, *LSTM* architecture, *transformers*
- Optimization over multiple tasks (not just a simple prediction problem like Word2vec)
- Character-level tokenization and embeddings
- Much more data and enormous compute power required
  - e.g. BERT trained on a 3.3 billion word corpus over 40 epochs, taking over 4 days to train on 64 TPU chips (each chip costs ~$10k).

Large language models



See Nvidia blog on Megatron-Turing NLG.

Summary

- Word embeddings use a neural language model to better represent texts as dense vectors
  - Distributional semantics
  - Analogical reasoning
  - Sociological analysis of meaning and representations
- Recent methodological advances better incorporate context
  - Better semantic representations but greater complexity