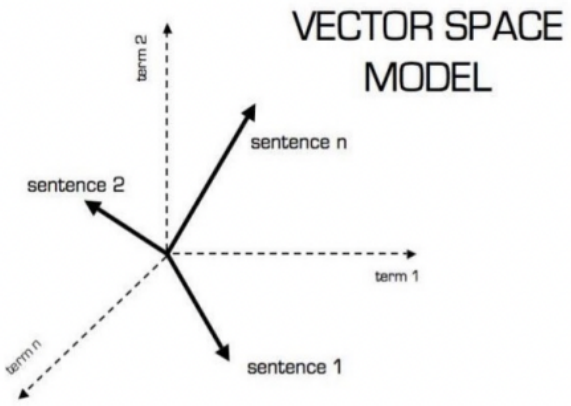
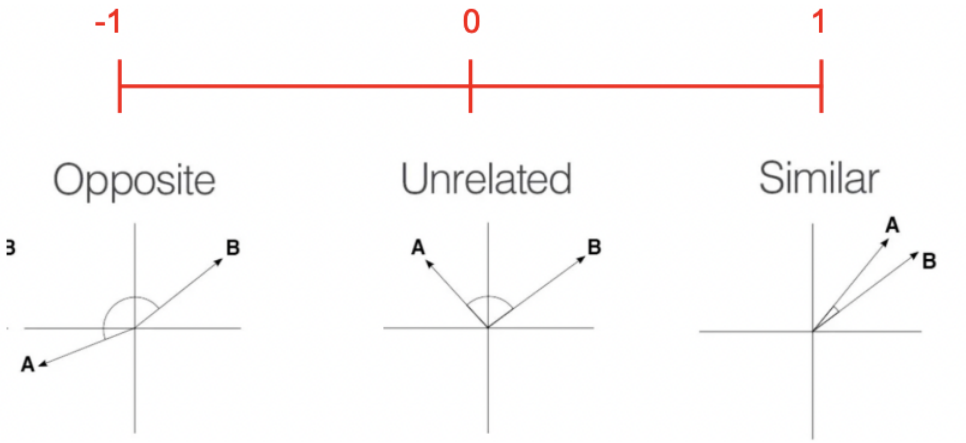
CAS CS 505
Lec 6

Vector Models Continued; Principal Component Analysis; Distributional semantics; Word embeddings and word2vec; Sentence and text embeddings

1. Text as vectors (TF BOW, TF-IDF, etc…)
    a. So we have a |V|-dimensional vector space (|V| is the size of vocabulary)
    b. Words/tokens are the axes of the space
    c. Sentences, documents etc. are points or vectors in this space
    d. Very high-dimensional: Number of dimensions = size of vocabulary, often 10,000 or more.
    e. These are very sparse vectors - most entries are zero.
    f. The frequency of the sentence becomes a vector in the vector space where the words are the axis
    g. 

2. Cosine similarity
    a. Recall: Cosine Similarity is a measure of how similar two objects in vector space are, as the cosine of the angle between them, irrespective of their magnitude. The scale is [-1 .. 1].
    b. We are interested in the percentage of words (not the length of sentences)
    c. The cosine similarity gives you the angle between the two vectors (how close the two vectors are, ignoring the length)
    d. $$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}},$$

e.




3. An Example: A Vector Space Model for Characters in PotC

a.

BOW for will turner:

| Word | Frequency |
|---|---|
| jack | 22 |
| will | 12 |
| find | 8 |
| get | 7 |
| elizabeth | 6 |
| know | 5 |
| us | 5 |
| euh | 5 |
| come | 5 |
| need | 5 |
| key | 5 |
| ship | 5 |
| hold | 5 |
| pearl | 4 |
| stop | 4 |
| sparrow | 4 |
| chest | 4 |
| back | 4 |
| away | 4 |
| leave | 3 |

BOW for jack sparrow:

| Word | Frequency |
|---|---|
| want | 15 |
| come | 11 |
| know | 9 |
| oh | 9 |
| will | 9 |
| bugger | 8 |
| dirt | 8 |
| love | 8 |
| hey | 7 |
| one | 7 |
| mate | 6 |
| captain | 6 |
| key | 6 |
| would | 6 |
| jones | 6 |
| save | 6 |
| jar | 6 |
| chest | 6 |
| much | 5 |
| way | 5 |

BOW for gibbs:

| Word | Frequency |
|---|---|
| jack | 11 |
| aye | 9 |
| us | 8 |
| will | 6 |
| ho | 5 |
| sea | 5 |
| got | 4 |
| think | 4 |
| bit | 4 |
| like | 4 |
| cast | 4 |
| rum | 3 |
| captain | 3 |
| seems | 3 |
| key | 3 |
| cap'n | 3 |
| back | 3 |
| made | 3 |
| chief | 3 |
| believe | 3 |

BOW for elizabeth swann:

| Word | Frequency |
|---|---|
| will | 22 |
| jack | 12 |
| find | 7 |
| know | 7 |
| oh | 7 |
| want | 6 |
| man | 6 |
| good | 5 |
| something | 5 |
| sparrow | 4 |
| would | 4 |
| chance | 4 |
| us | 3 |
| captain | 3 |
| compass | 3 |
| give | 3 |
| going | 3 |
| came | 3 |
| way | 3 |
| yes | 3 |

BOW for lord cutler beckett:

| Word | Frequency |
|---|---|
| jack | 7 |
| sparrow | 7 |
| one | 5 |
| will | 5 |
| compass | 5 |
| mister | 4 |
| turner | 4 |
| freedom | 3 |
| world | 3 |
| must | 3 |
| something | 3 |
| currency | 3 |
| governor | 2 |
| mercer | 2 |
| arrest | 2 |
| william | 2 |
| oh | 2 |
| believe | 2 |
| question | 2 |
| perhaps | 2 |

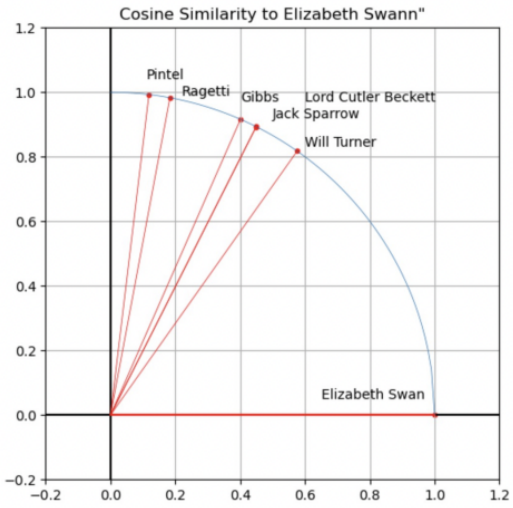
b. First, we calculate the cosine similarity between each character's BOW

c.

| Cosine Similarity | Degrees | Characters | |
|---|---|---|---|
| 0.5758 | 54.84 | elizabeth swann | will turner |
| 0.4909 | 60.6 | gibbs | will turner |
| 0.4497 | 63.28 | elizabeth swann | lord cutler beckett |
| 0.4493 | 63.3 | elizabeth swann | jack sparrow |
| 0.4203 | 65.15 | jack sparrow | will turner |
| 0.401 | 66.36 | elizabeth swann | gibbs |
| 0.3982 | 66.53 | lord cutler beckett | will turner |
| 0.3618 | 68.79 | davy jones | will turner |
| 0.344 | 69.88 | gibbs | jack sparrow |
| 0.3383 | 70.23 | davy jones | elizabeth swann |
| 0.3236 | 71.12 | jack sparrow | lord cutler beckett |
| 0.3093 | 71.98 | davy jones | lord cutler beckett |
| 0.3021 | 72.42 | gibbs | lord cutler beckett |
| 0.2823 | 73.6 | gibbs | pintel |
| 0.2807 | 73.7 | pintel | ragetti |
| 0.2752 | 74.03 | davy jones | gibbs |
| 0.2717 | 74.23 | davy jones | jack sparrow |
| 0.2709 | 74.28 | gibbs | ragetti |
| 0.2408 | 76.07 | jack sparrow | pintel |
| 0.2357 | 76.37 | jack sparrow | ragetti |
| 0.22 | 77.29 | ragetti | will turner |
| 0.1834 | 79.43 | elizabeth swann | ragetti |
| 0.1826 | 79.48 | pintel | will turner |
| 0.1545 | 81.11 | lord cutler beckett | ragetti |
| 0.1326 | 82.38 | davy jones | ragetti |
| 0.1187 | 83.18 | elizabeth swann | pintel |
| 0.0962 | 84.48 | lord cutler beckett | pintel |
| 0.0936 | 84.63 | davy jones | pintel |

d. Here is each character's cosine similarity to Elizabeth Swann:

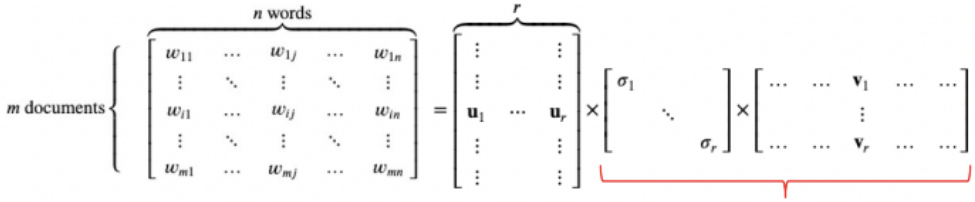
| Cosine Similarity | Degrees | Characters | |
|---|---|---|---|
| 0.5758 | 54.8443 | elizabeth swann | will turner |
| 0.4497 | 63.2756 | elizabeth swann | lord cutler beckett |
| 0.4493 | 63.3012 | elizabeth swann | jack sparrow |
| 0.401 | 66.3593 | elizabeth swann | gibbs |
| 0.1834 | 79.4321 | elizabeth swann | ragetti |
| 0.1187 | 83.1829 | elizabeth swann | pintel |



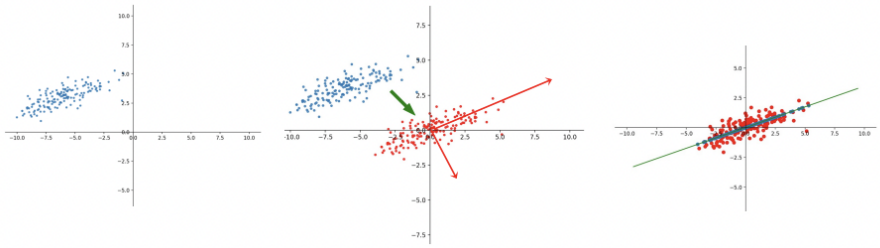

e. This does not mean that Pintel and Ragetti are close to each other.

f. This simply means that they are highly unrelated with Elizabeth Swann

4. Digression: Dimensionality Reduction using PCA
   a. The total vocabulary size for this task is 1249 words; thus the cosine similarity for character vectors takes place in 1249 dimensions!
   b. It is impossible to get any intuition for so many dimensions!
   c. Principal Components Analysis
      i. Factor the term-document matrix using singular value decomposition:



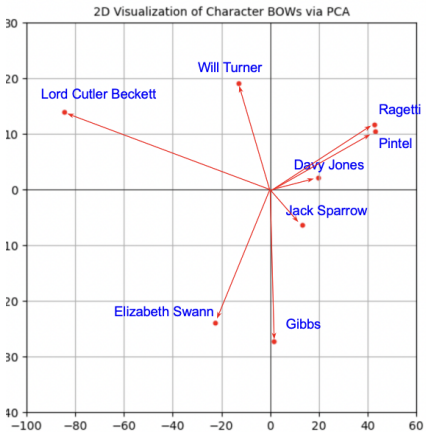

      ii.
      iii. The eigenvectors are the principal components of the data and the eigenvalues $o_1, \ldots, o_n$ give the "importance" of each componenent.
   d. Eigenvectors can be used with images
   e. It reduces the dataset by eigenvalues providing the importance of each word
   f. Principal Components Analysis
      i. The principal components tell you which parts of the data are more significance (have more variance)
      ii. The largest two principal components give us a 2D view of the data;
      iii. Illustration of reducing a 2D data set to 1 dimension:



iv.

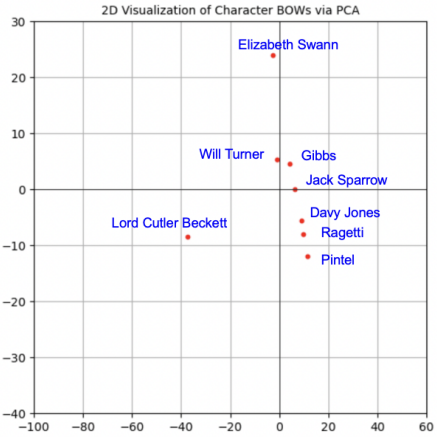
g. The steps:
    i. Center the components
    ii. Look at the distance of the plots in a few directions
    iii. Choose the line with the longest length
h. Taking the data → reducing the dimensions → preserving as much data as possible (you are still losing information)
i. The axis do not correspond to any of the original words

5. Vector Space models for Characters in PotC
    a. Just using Term Frequency Counts (stop words Not deleted)



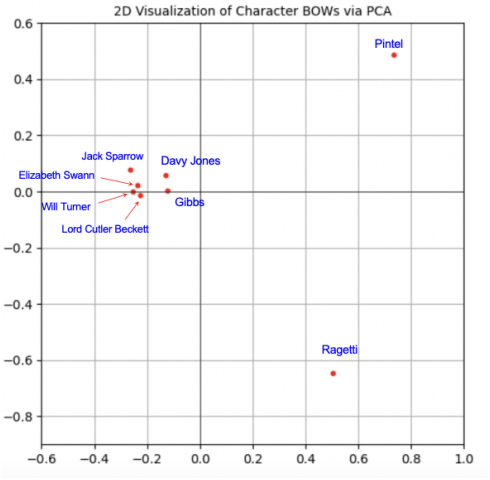

i.

    ii. This was reduced into 15 * 2 from 15 * 1249
    b. Just using Term Frequency Counts (stop words deleted)

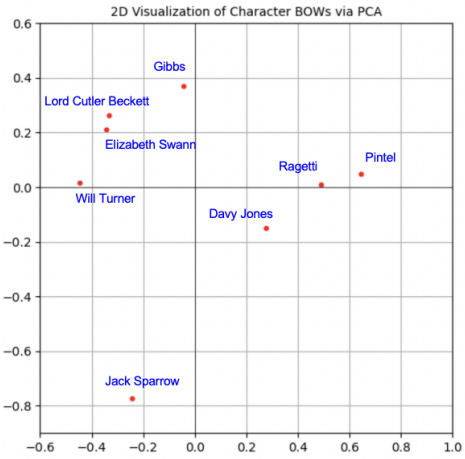



i.

c. Just using TF-IDF (stop words NOT deleted)

i. 


d. Just using TF-IDF (with stop words deleted)

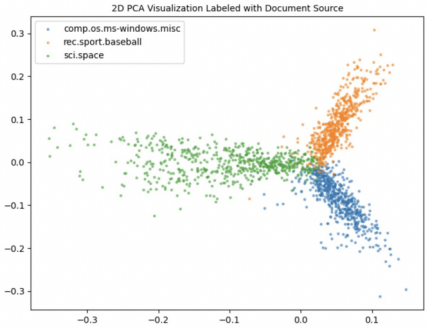
i. 


e. Why there is not more consistency between these different views of the data?

i. There is no much data → cannot find patterns

6. Vector Space Models for Social Media

```
from sklearn.datasets import fetch_20newsgroups
categories = ['comp.os.ms-windows.misc', 'sci.space', 'rec.sport.baseball']
news_data = fetch_20newsgroups(subset='train', categories=categories)
```

a. 


b. We are not doing cosine similarity, so the angles do not matter anymore

c. They have different trends and suggest that the data is classifiable

7. What's wrong with Term-Frequency Vector Models
    a. Inefficient: Huge sparse vectors (could be 100,000 words!), mostly 0's.
        i. They don't scale well!
    b. They don't generalize well:
        i. TF models don't generalize across NLP tasks and domains
        ii. Can't be used for transfer learning
    c. Statistics ≠ Semantics:
        i. They do not have semantics (there is no ordering)
        ii. Cannot tell the subtle differences between words
        iii. What words occur together?
        iv. How are they used?
        v. What do words mean?
        vi. There is no sense in these questions answered in the model
        vii. Can only find the relationship regarding probability
    d. Can't handle complex linguistic phenomena:
        i. Synonyms: car and automobile have no relationship!
        ii. Polysemy (multiple meanings): sound = audio signal, healthy, body of water (19 noun meanings, 12 adjective meanings, 12 verb meanings, etc.)
        iii. In python, words are stored as strings and do not mean anything (that is why we assign probabilities to each and conduct the code)
    e. Short, dense vectors are a better solution!
        i. Short: 50-1000 dimensions o Dense: Most elements are not 0
        ii. Efficient: Easier to use as features in machine learning
        iii. Generalize better than explicit counts
        iv. May capture synonymy better, since fewer dimensions
        v. In general, to capture semantics better!
8. Vector Models of Meaning: Short and Dense
    a. Naïve idea: Meaning as a point in space (Osgood et al. 1957) Define a word by abstract characteristics:
        i. 3 affective dimensions for a word
            1. valence: pleasantness
            2. arousal: intensity of emotion
            3. dominance: the degree of control exerted
        ii. Hence the connotation of a word is a vector in 3-space

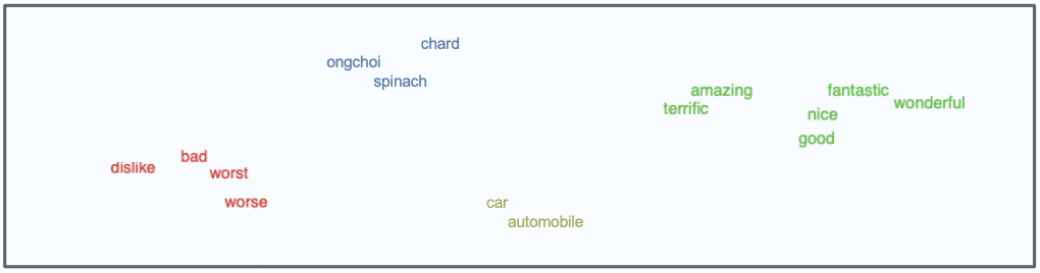
| | Word | Score | | Word | Score |
|---|---|---|---|---|---|
| **Valence** | love | 1.000 | | toxic | 0.008 |
| | happy | 1.000 | | nightmare | 0.005 |
| **Arousal** | elated | 0.960 | | mellow | 0.069 |
| | frenzy | 0.965 | | napping | 0.046 |
| **Dominance** | powerful | 0.991 | | weak | 0.045 |
| | leadership | 0.983 | | empty | 0.081 |

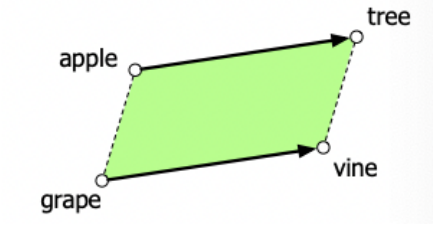
        iii.
        iv. But this doesn't generalize well to all English words: what about nouns?

b. Embeddings are short and dense word/text vectors (take a word and put it in a vector)
    i. “Neural Language Model”-inspired embeddings
        1. Word2vec (skipgram, CBOW), GloVe
    ii. Singular Value Decomposition (SVD)
        1. A special case of this is called LSA – Latent Semantic Analysis
    iii. Alternative to these "static embeddings":
        1. Contextual Embeddings (ELMo, BERT)
        2. Compute distinct embeddings for a word in its context
        3. Separate embeddings for each token of a word

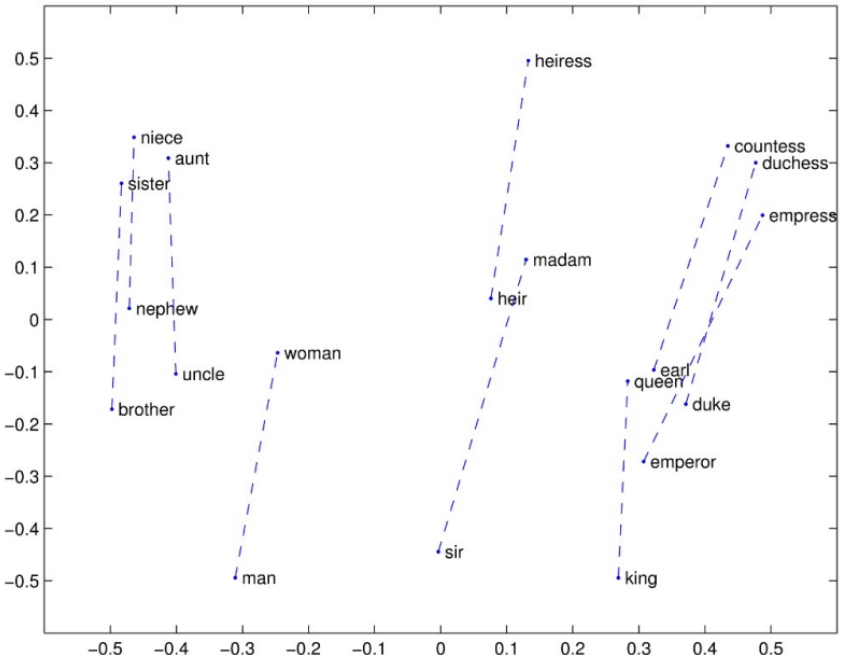
9. Vector Models of Meaning: Embeddings
    a. How to capture the meaning of words? By context!
    b. Ludwig Wittgenstein: "The meaning of a word is its use in the language”
    c. Distributional Semantics: “You shall know a word by the company it keeps” (Firth 1959)
        i. Words are defined by the words around them
        ii. Synonyms can be substituted into the same contexts
            1. “The fast car was speeding down the road.”
            2. “The fast automobile was speeding down the road.”
            3. The word “car” and “automobile” are synonym and therefore can be used interchangeably
    d. What does recent English borrowing ongchoi mean?
        i. Suppose you see these sentences:
            1. Ong choi is delicious sautéed with garlic.
            2. Ong choi is superb over rice
            3. Ong choi leaves with salty sauces
        ii. And you've also seen these:
            1. …spinach sautéed with garlic over rice
            2. Chard stems and leaves are delicious
            3. Collard greens and other salty leafy greens
        iii. Conclusion:
            1. Ongchoi is a leafy green like spinach, chard, or collard greens
                a. We could conclude this based on words like "leaves" and "delicious" and "sauteed"
    e. Simple way to think about embeddings: A word is placed in vector space by its context
    f. A skip-gram is a context (+/- N) before and after a word:
        i. (2,2) Skip-Gram
        ii. The fast car was speeding
    g. A word context is a set of words nearby (i.e., +/- N words):

i. Word Word context:
ii. car {fast, speeding , the, was }

h. Words have similar meanings iff they have similar contexts.
i. This is a machine learning task (return later)
   i. Start with a random D-dimensional vectors as a word's initial embedding
   ii. Train a classifier based on embedding similarity
   iii. Take a corpus and take pairs of words that co-occur as positive examples
   iv. Take pairs of words that don't co-occur as negative examples
   v. Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
   vi. Throw away the classifier code and keep the embeddings.
j. Embeddings keep similar words near each other and dissimilar words far apart.
k. 

l. They also allow a simple form of analogical reasoning: To solve: "Apple is to tree as grape is to _____?"
   i. 

   ii. Use vector arithmetic:
      1. X = tree – apple + grape
   iii. Search for the closest word (using cosine sim):
      1. argminX( cosine_sim(tree – apple + grape, X) ) => vine

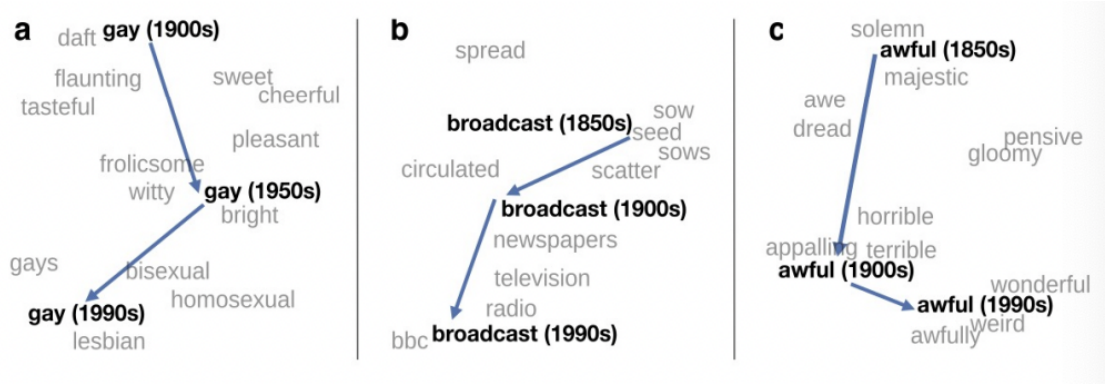



m.

n. Embeddings as a window onto historical semantics

o. Train embeddings on different decades of historical text to see meanings shift

   i. ~30 million books, 1850-1990, Google Books data




p.

q. Word embeddings can be extended to document embeddings:

r. Generally, a document embedding is simply the sum of its word embeddings.

s.