Introduction to Machine Learning Applications

Spring 2023

Natural language modeling

Minor Gordon

gordom6@rpi.edu

Rensselaer

Text mining

Text mining typically starts with a set of highly heterogeneous input texts.

- How is the text formatted?
- Which types of text are present?
- What should be removed because it is irrelevant?
- What types of analyses are likely to be useful?

Key terms

- Document: a piece of text
- Corpus: a collection of documents
- Terms/tokens: a word in a document
- Entity: person, place, organization, etc.

Tokens

- Tokens: character strings separated by whitespace and sometimes punctuation
- Example:
  - The bridge flies
    - w1 = The
    - w2 = bridge
    - w3 = flies
  - She’s
    - w1 = she
    - w2 = s

Stop Words

- Words/tokens that are filtered out *before* the overall analysis is conducted

"hither","home","how","howbeit","however","hundred","i","id","ie","if","i'll","im","immediate","immediately”,"importance","important","in","inc","indeed","index","information","instead","into","invention","inward","is","isn't”,"it","itd","it'll","its","itself","i've","j","just","k","keep",”keeps","kept","keys","kg","km","know","known","knows","l","largely","last","lately","later","latter","latterly”,"least","less","lest","let","lets","like","liked","likely","line","little","'ll","look","looking","looks","ltd","m","made”,"mainly","make","makes","many","may","maybe","me","mean","means","meantime","meanwhile","merely","mg”,"might","million","miss","ml","more","moreover","most","mostly","mr","mrs","much","mug","must","my","myself","n”,"na","name","namely","nay","nd","near","nearly","necessarily","necessary","need","needs","neither","never”,"nevertheless","new","next","nine","ninety","no”…..

Stemming

- The process for reducing derived words to their word stem or base
- Usually rule-based, syntactic

  - connection
  - connections
  - connective ---> connect
  - connected
  - connecting

Lemmatization

- The process of grouping together different forms of a word to capture a single meaning
- Usually more sophisticated than stemming, involving Part of Speech tagging
  - Change the stemming rules based on the PoS

  – run
  – runs -> run
  – ran

Text vectorization: converting a document into a numeric feature vector

Term Frequency

- How often does a term t_i occur in the document d_j?

$$TF(i,j) = n_{ij}$$

- How often does a term occur in the dataset (normalized for length of document)

$$TF(i,j) = n_{ij} / \sum n_{kj}$$

Term Document Matrix

A document-term matrix or term-document matrix … describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms.

D1 = "I like databases"
D2 = "I hate databases",

	I	**like**	**hate**	**databases**	
D1	1	1		0	1
D2	1	0		1	1

https://en.wikipedia.org/wiki/Document-term_matrix

Bag of Words: model

- Documents are bags of words
- Ignore word order or other linguistic structure
- Each word is a feature, so the goal will be to identify relevant words

Bag of Words: example

(1) John likes to watch movies. Mary likes movies too.
(2) John also likes to watch football games.

Vocabulary:

```
"John",
"likes",
"to",
"watch",
"movies",
"also",
"football",
"games",
"Mary",
"too"
```

Feature Vector

(1) [1, 2, 1, 1, 2, 0, 0, 0, 1, 1]
(2) [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]

https://en.wikipedia.org/wiki/Bag-of-words_model#Example_implementation

Inverse Document Frequency (IDF)

- A measure of sparseness
- Is this word in lots of different documents? Or is it a unique aspect that defines this document
    - "What's cooking": is this a unique ingredient that defines the recipe or a common ingredient in many recipes?

$$IDF(t) = 1 + \log(totalDocuments / NumberContainingt)$$

TF-IDF

- Term Frequency – Inverse Document Frequency
  - This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus (data-set).

  → Scaling the document vector we saw in Bag of Words to get an ML model-friendly feature vector
- Often used in information retrieval: scoring and ranking a document's relevance, given a query

TF-IDF Computation

- **STEP-1: Normalized Term Frequency (tf)** -- tf(t, d) = N(t, d) / ||D|| wherein, ||D|| = Total number of term in the document
  - tf(t, d) = term frequency for a term t in document d.
  - N(t, d) = number of times a term t occurs in document d
- **STEP-2: Inverse Document Frequency (idf)** -- idf(t) = N / df(t) = N/N(t)
  - idf(t) = log(N/ df(t))
  - idf(pizza) = log(Total Number Of Documents / Number Of Documents with term pizza in it)
- **STEP-3: tf-idf Scoring**
  - tf-idf(t, d) = tf(t, d)* idf(t, d)

A high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and tf–idf) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf–idf closer to 0.

https://en.wikipedia.org/wiki/Tf%E2%80%93idf#Term_frequency%E2%80%93inverse_document_frequency

https://en.wikipedia.org/wiki/Karen_Sp%C3%A4rck_Jones#/media/File:Karen_Sp%C3%A4rck.jpg
University of Cambridge, CC BY 2.5

TF-IDF Example

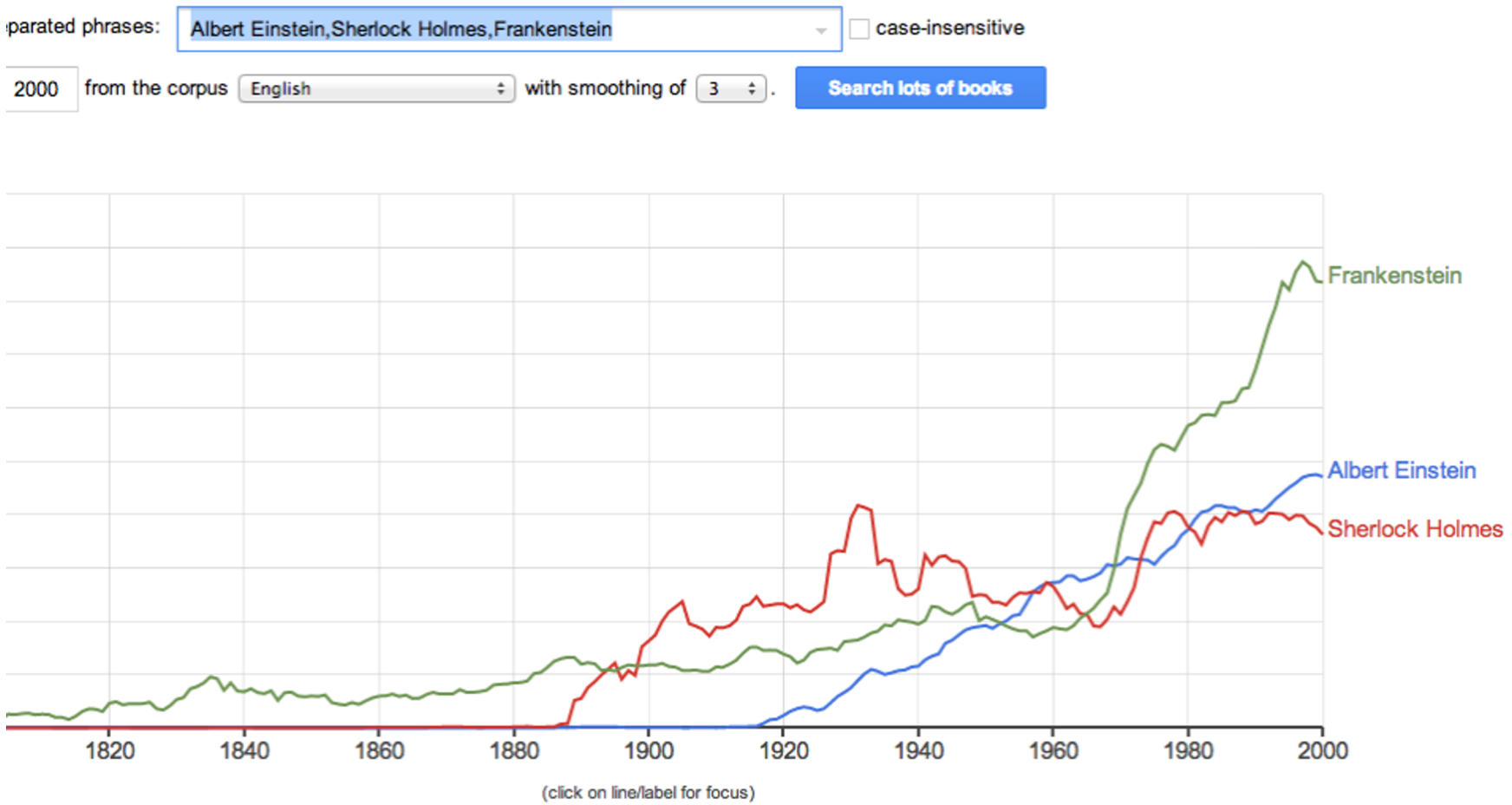
- Consider a document containing 100 words where:
  - the word “kitty” appears 3 times. The term frequency (i.e., tf) for kitty is then (3 / 100) = 0.03.
- Now, assume we have 10 million documents and the word kitty appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4.
- Thus, the Tf-Idf weight for the word “kitty” is the product of these quantities: 0.03 * 4 = 0.12.

n-grams

- An *n*-gram is a contiguous sequence of *n* items from a given text.
  - Typically words/terms, but could be syllables, letters, words or base
- The value of *n* could be >=1

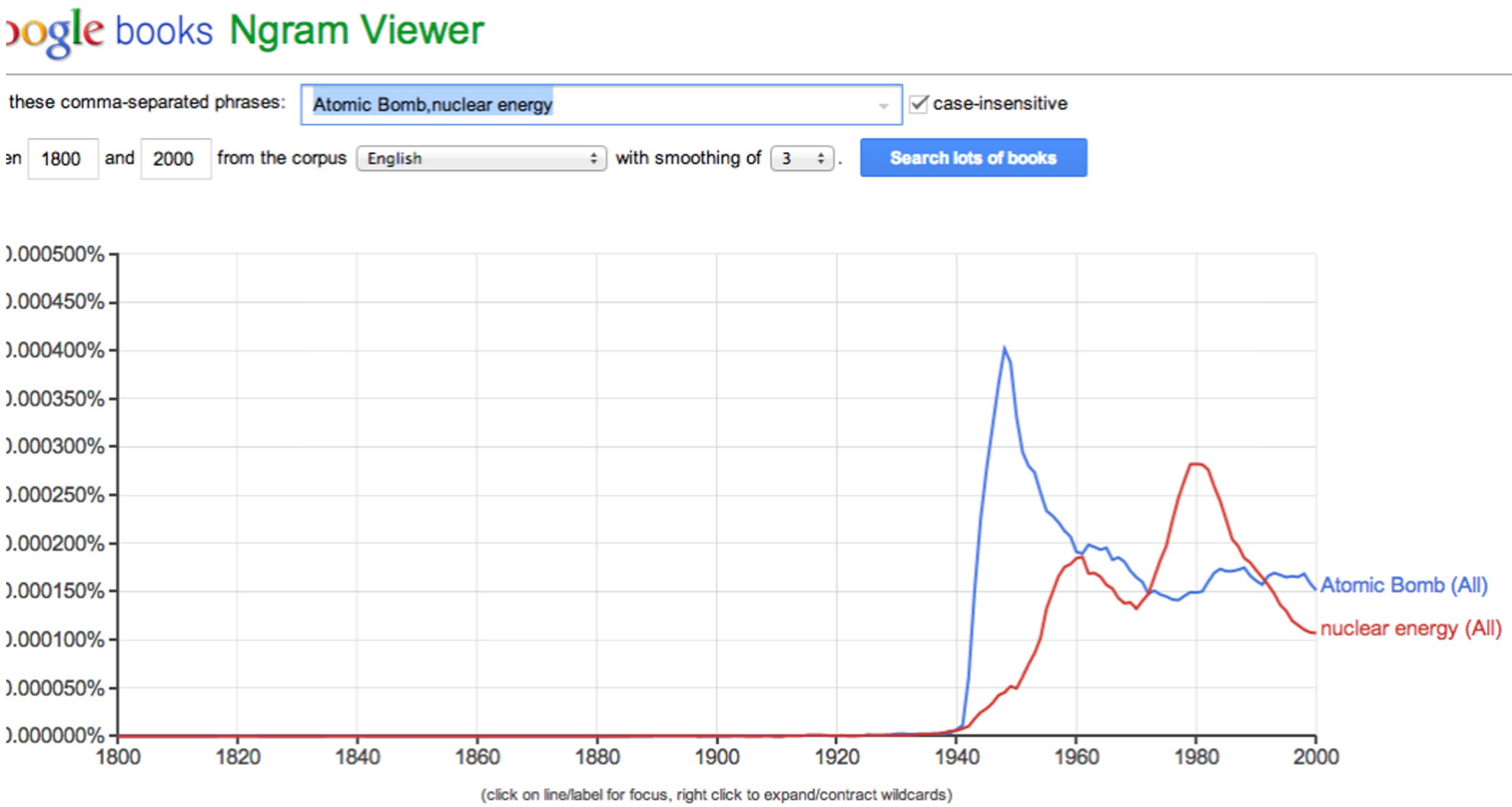
- Example: "Begin at the Beginning"
  - *Uni*-grams: {Begin, at, the, Beginning}
  - *Bi*-grams: {Begin at, at the, the Beginning}
  - *Tri*-grams: {Begin at the, at the Beginning}

Google N-Gram Viewer



https://books.google.com/ngrams

Google N-Gram Viewer



https://books.google.com/ngrams

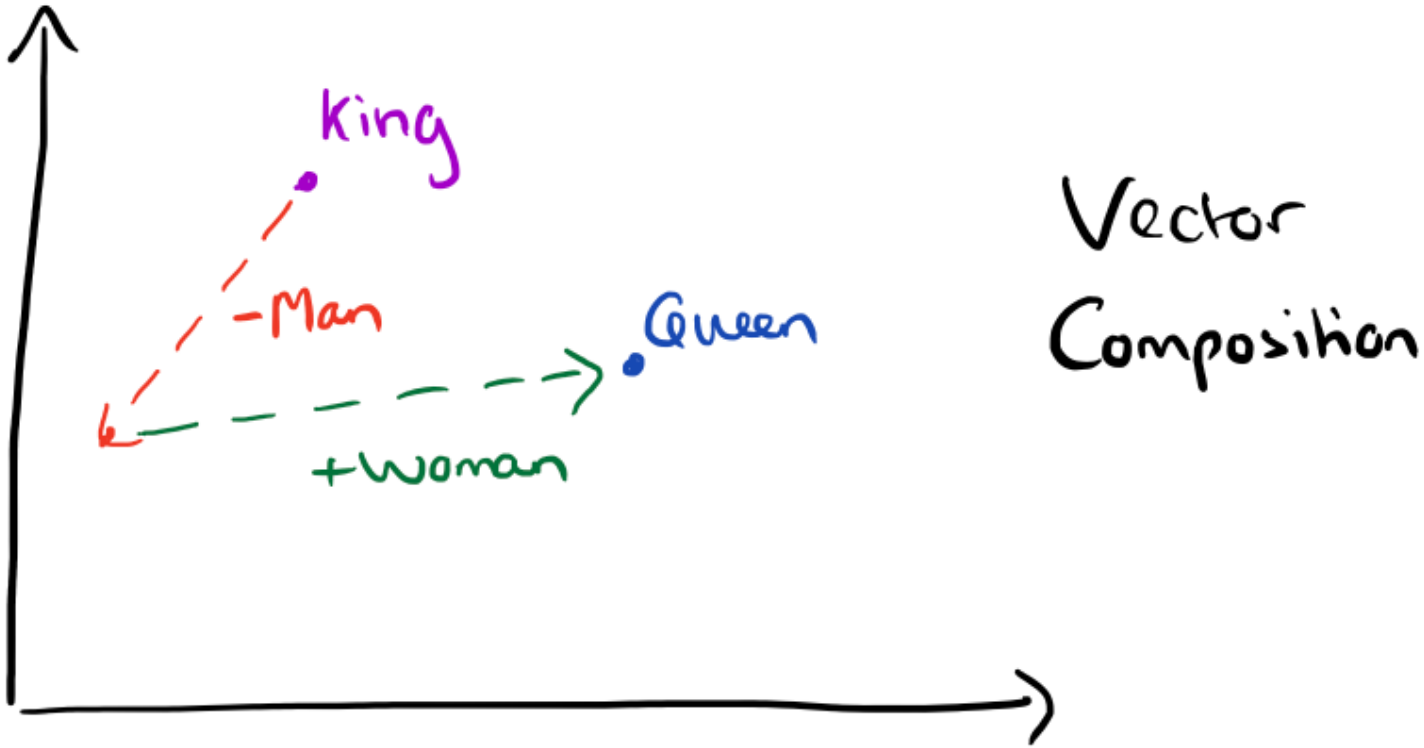
Word2vec

Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space.

Word2vec

- Word2vec uses word embeddings (relative position of words in sentence) to understand the relationships between words

This Results in an Amazing Property that Looks Like Math Magic



https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a method of pretraining language representations (Devlin et al., NAACL 2019)

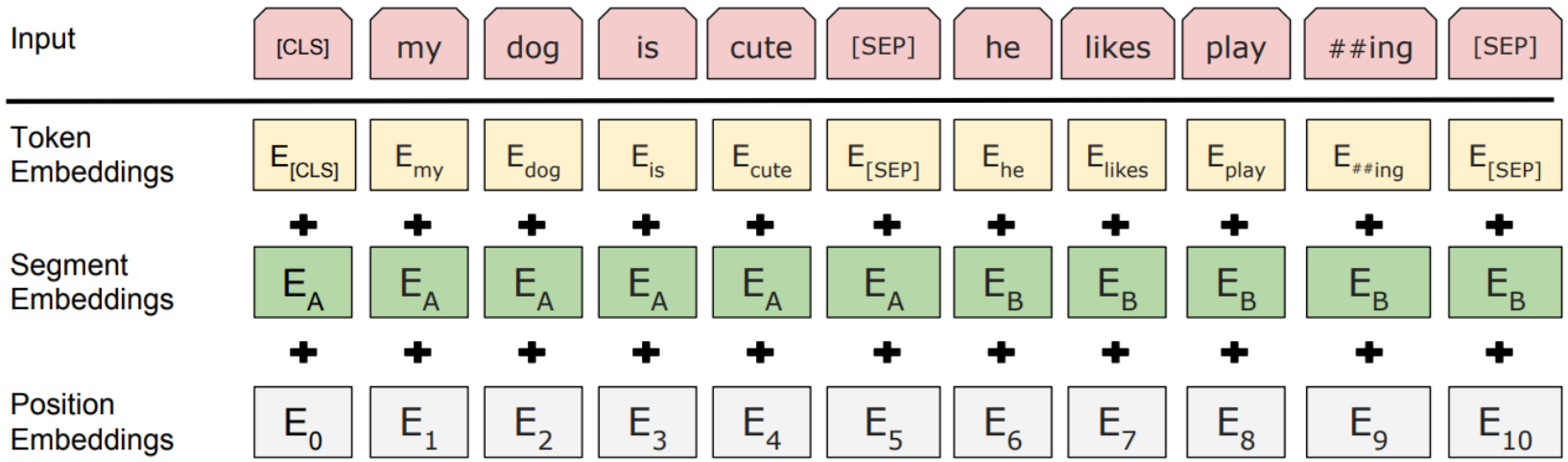


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Topic modeling

- There could be a wide variety of topics that may relate to a large number of words
- There are many types of advanced models for this, but it can also be simple

Example: From Random Acts of Pizza, we may want to incorporate a variety of items related to someone's job

Latent Dirichlet Allocation (LDA)

- Topic modeling: a *topic* is considered to be a set of terms that, taken together, suggest a shared theme.
- Example:
  - *dog*, *spaniel*, *beagle*, *golden retriever*, *puppy*, *bark*, and *woof* would suggest a **DOG_related**
  - *cat*, *siamese*, *main coon*, *tabby*, *manx*, *meow*, *purr*, and *kitten* would suggest a **CAT_related**

TOPIC MODELING is like PCA = DIMENSIONALITY REDUCTION