Class 6: Text Basics

Topic 2: Text

Computational Analysis of Text, Audio, and Images, Fall 2023

Aarhus University

Mathias Rask (mathiasrask@ps.au.dk)

Aarhus University

Workflow

1. Collect corpus
2. Processing of text (e.g. removal of stopwords or digits)
3. Numerical representation
   - Bag-of-Words (BoW):
     - ▷ one-hot encoding
     - ▷ binary/count representation
     - ▷ tf-idf representation
   - Embeddings:
     - ▷ Static (e.g. Word2Vec)
     - ▷ Dynamic (e.g. Transformers)
     - ▷ ...
4. Analysis and inference
   - Ideology, sentiment, emotive rhetoric
   - Text similarity and classification
   - Identifying topics

   ⇝ Can be done using ML/DL tools!

Assumptions

All quantitative analyses of texts require assumptions. What must we assume?

1. Text is an observable implication of some underlying characteristic
   - ⇝ Text conveys meaning
2. The observable implication in a text can be numerically represented
   - ⇝ The "good" variation is not lost in vectorization
3. If humans can, so can computers
   - ⇝ If we can not detect what we are looking for, computers probably can't as well

Key Concepts

- Corpus: A collection text/documents
- Text/Document: One observation
- Word: A single word
- Tokens: Splitting text into smaller units (e.g. words)
- Vocabulary: Unique tokens in the corpus
- Stemming: Words with suffixes removed (“Caring” $\rightarrow$ “Car”)
- Lemmatization: Word base (“Caring” $\rightarrow$ “Care”)
- Stop words: Words that are excluded from analysis
- Digits: Numbers and digits

Notation:
A corpus $\mathcal{C}$ with documents/texts $\mathcal{D}_i$ for $i \in \{1, \dots, N\}$ where the vocabulary $\mathcal{V}$ is the set of all unique tokens in $\mathcal{C}$: $|\mathcal{V}| \leq |\mathcal{C}|$.

Today’s Menu

Tokenization and Preprocessing

Vectorization

Table of Contents

Tokenization

The first step is to break our text into words:

$$\underbrace{\text{“Toronto ligger 159km fra Buffalo.”}}_{\text{raw text}} \underbrace{\rightarrow}_{\text{tokenizer}} \underbrace{[\text{Toronto, ligger, 159, km, fra, Buffalo, .}]}_{\text{tokens}}$$

Tokens:

- Words
- Numbers
- Punctuation
- Special characters (N.Y.C)

Libraries:

spaCy Hugging Face NLTK AllenNLP

Preprocessing

Often, we also want to preprocess our text. Why?

⇝ To increase signal-to-noise ratio

- ▷ Eliminate “bad” variation, maintain the “good”

Operations:

- Common steps: Casing, stopwords/digits, and word reduction
- Advanced steps: NER, POS, parsing
- Others: n-grams and removal of frequent/infrequent tokens

⇝ The preprocessing steps are highly dependent upon the task at hand!

⇝ And our results are (often) sensitive to the preprocessing steps!

Casing

Upper and lower casing: “A” vs “a”

Example: “Venstre er nu til venstre for midten”

Without casing	**With casing**
{Venstre, er, nu, til, venstre, for midten}	{venstre, er, nu, til, for midten}

Implications:

- $\rightsquigarrow$ Reduces $\mathcal{V}$
- $\rightsquigarrow$ Information loss

Stopwords

Frequent but low-information words

Example: "Venstre er nu til venstre for midten"

Without removal	**With removal**
{Venstre, er, nu, til, venstre, for midten}	{Venstre, venstre, midten}

Implications:

- ⇝ Reduces $\mathcal{V}$
- ⇝ Reduces context

Stemming and Lemmatization

Reducing words:

- Stemming: Removes suffixes (ends) of words
- Lemmatization: Base words

Example: "Udlændinge kommer herop og begår kriminalitet"

tokens $\rightsquigarrow$ {udlændinge, kommer, herop, og, begår, kriminalitet}

Stemming

{udlænding, kom, herop og, begår, kriminalit}

Lemmatization

{udlænding, komme, herop, og, begå, kriminalitet}

Implications:

- $\rightsquigarrow$ Reduces $\mathcal{V}$
- $\rightsquigarrow$ Meaningless words ("kriminalit")
- $\rightsquigarrow$ Word choice is insensitive

Table of Contents

Why Vectorize?

A corpus $\mathcal{C}$ with $|\mathcal{C}| = 3$

$\mathcal{D}_1$: [Venstre, er, nu, til, venstre, for midten]

$\mathcal{D}_2$: [Udlændinge, kommer, herop, og, begår, kriminalitet]

$\mathcal{D}_3$: [Vi, støtter, lovforslaget]

Imagine we want to identify speeches about crime. How could we do that?

1. Rule-based
2. Machine/deep learning
    - ▷ Algorithms require fixed-length vectors!

Vectorization enables us to construct fixed-length representations by mapping words to numbers:

- Absolute word presence
- Relative word presence
- Word meaning

⇝ Ideally, our vectorization encodes the semantics and meaning of text

Bag-of-Words (BoW)

A general approach is to vectorize text by the presence of words

- Main idea: The meaning of a text is encoded in $\mathcal{V}$
  - $\mathcal{D}_i$ and $\mathcal{D}_j$ belong to the same class if they share common words
- Called a "bag" since we throw away the order of the words
  - We only care about whether a word is present or not

Binary Bow

Corpus $\mathcal{C}$ with $|\mathcal{C}| = 4$:

$\mathcal{D}_1$: "Red Bull drops hint on F1 engine."

$\mathcal{D}_2$: "Honda exits F1, leaving F1 partner Red Bull."

$\mathcal{D}_3$: "Hamilton eyes record eighth F1 title."

$\mathcal{D}_4$: "Aston Martin announces sponsor."

Vocabulary $\mathcal{V}$ with $|\mathcal{V}| = 20$:

- ['Aston' 'Bull' 'F1' 'Hamilton' 'Honda' 'Martin' 'Red' 'announces' 'drops', 'eighth' 'engine' 'exits' 'eyes' 'hint' 'leaving' 'on' 'partner' 'record', 'sponsor', 'title']
- { 'Aston': 0, 'Bull': 1, 'F1': 2, 'Hamilton': 3, 'Honda': 4, 'Martin': 5, 'Red': 6, 'announces': 7, 'drops': 8, 'eighth': 9, 'engine': 10, 'exits': 11, 'eyes': 12, 'hint': 13, 'leaving': 14, 'on': 15, 'partner': 16, 'record': 17, 'sponsor': 18, 'title': 19}

Binary BoW

From $\mathcal{V}$ to document-feature-matrix:

- { 'Aston': 0, 'Bull': 1, 'F1': 2, 'Hamilton': 3, 'Honda': 4, 'Martin': 5, 'Red': 6, 'announces': 7, 'drops': 8, 'eighth': 9, 'engine': 10, 'exits': 11, 'eyes': 12, 'hint': 13, 'leaving': 14, 'on': 15, 'partner': 16, 'record': 17, 'sponsor': 18, 'title': 19}

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$\mathcal{D}_1$	0	1	1	0	0	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0
$\mathcal{D}_2$	0	1	1	0	1	0	1	0	0	0	0	1	0	0	1	0	1	0	0	0
$\mathcal{D}_3$	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1
$\mathcal{D}_4$	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0

How many columns would have with ignoring the order?

$$P(n,k) = \frac{!n}{(n-k)!} = \frac{!6}{(6-6)!} = \frac{720}{1} = 720$$

What is the assumption behind binary vectorization?

Frequency BoW

Binary:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$\mathcal{D}_1$	0	1	1	0	0	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0
$\mathcal{D}_2$	0	1	**1**	0	1	0	1	0	0	0	0	1	0	0	1	0	1	0	0	0
$\mathcal{D}_3$	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1
$\mathcal{D}_4$	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Frequency:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$\mathcal{D}_1$	0	1	1	0	0	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0
$\mathcal{D}_2$	0	1	**2**	0	1	0	1	0	0	0	0	1	0	0	1	0	1	0	0	0
$\mathcal{D}_3$	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1
$\mathcal{D}_4$	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0

What is the assumption behind count vectorization?

What Did We Achieve?

- Fixed-length numerical vectors
- From a sequence of symbols to points in a multidimensional vector space – what's the dimension?
  - The $|\mathcal{V}|$-dimensional space is intended to encode some meaning about our text
    - ▷ Meaning is encoded by the presence of words using BoW

$\rightsquigarrow$ Huge step! Implication?

- ▷ We can quantify that texts sharing the same vocabulary have closer vectors in the vector space
- ▷ We can measure similarity! (classification/matching)

Measuring Similarity: The Dot-Product

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$$

- $\mathbf{a}$ and $\mathbf{b}$ are equal length vectors
- Their corresponding elements are multiplied and added together
- Example:

$$\mathbf{a} = [1, 2, 3] \qquad \mathbf{b} = [4, 5, 6]$$

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= (1 \cdot 4) + (2 \cdot 5) + (3 \cdot 6) \\ &= 4 + 10 + 18 \\ &= 32 \end{aligned}$$

Why can't we just use the simple dot product?

$\rightsquigarrow$ Higher frequency leads to a larger dot product and hence larger similarity!

Solution:

Normalize by vector lengths:

$$\underbrace{\|\mathbf{a}\| = \sqrt{\sum_{i=1}^{n} a_i^2}}_{\text{Euclidean norm}}$$

- $\mathbf{a} = [1, 2, 3]$
- $1^2 + 2^2 + 3^2 = 1 + 4 + 9 = 14$

Result: Cosine similarity

$$\cos\theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$$

- Bounded between -1 and 1
- ± 1: Similar
- 0: Dissimilar
- When using BoW, we won't have values lower than 0. Why?

Exercise

Consider the two sentences:

$\mathcal{D}_1$: [Jeg, elsker, slik]

$\mathcal{D}_2$: [Chokolade, er min favorit]

Tasks:

1. Discuss whether they convey a similar meaning. Do want our similarity measure to be high or low?
2. Convert the sentences to a document-term-matrix using a binary BoW with documents as rows and words as columns
3. Compute the cosine similarity between the two documents (*Hint:* it is sufficient to compute the dot-product in this case. Why?)

Exercise

Document-term-matrix:

	jeg	elsker	slik	chokolade	er	min	favorit
	0	1	2	3	4	5	6
$\mathcal{D}_1$	1	1	1	0	0	0	0
$\mathcal{D}_2$	0	0	0	1	1	1	1

Vectors:

$$\begin{aligned} \mathbf{a} &= [1, 1, 1, 0, 0, 0, 0] \\ \mathbf{b} &= [1, 1, 1, 0, 0, 0, 0] \end{aligned}$$

Dot product:

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= \sum_{i=1}^{n} a_i b_i \\ &= (1 \cdot 0) + (1 \cdot 0) + (1 \cdot 0) + (0 \cdot 1) + (0 \cdot 1) + (0 \cdot 1) + (0 \cdot 1) \\ &= 0 \end{aligned}$$

Drawbacks

- Doesn't capture synonyms and semantics
- Can't handle OOV words (out-of-vocabulary)
- Inefficient representation: sparse matrix (a lot of 0s)
- Doesn't capture word ordering ("FCM slår FCK" vs."FCK slår FCM")
  - $\rightsquigarrow$ Can be somewhat mitigated with n-grams!

N-grams: Joining Consecutive Words

Grams:

- 2-gram: Combines two tokens
- 3-gram: Combines three tokens
- ⋮
- Examples:
  1. "FCM slår FCK"
     - ▷ Token 1: [FCM, slår]
     - ▷ Token 2: [slår, FCK]
  2. "FCK slår FCM"
     - Token 1: [FCK, slår]
     - Token 2: [slår, FCM]
- Helps capture context better than single tokens
- Possible to combine single tokens with n-grams (e.g. "social media")
- Drawbacks:
  - ▷ Can't still handle OOV words (out-of-vocabulary)
  - ▷ Dimensionality increases

Weighting Words

Binary and frequency BoW vectorization is a good baseline, but we can do better using a simple BoW approach $\rightsquigarrow$ *relative* frequency

Intuition:

1. If a word occurs frequently in a single $\mathcal{D}_i$ (or a few) but not in other documents that word is likely important for $\mathcal{D}_i$
2. If a word occurs frequently in *all* documents that word is *less* important

Approach: TF-IDF (Term Frequency – Inverse Document Frequency)

- Term frequency (TF): $\mathit{tf}(t, \mathcal{D}_i) = \log_{10}(f_{t\mathcal{D}_i} + 1)$
- Inverse Document Frequency (IDF): $\mathit{idf}(t, \mathcal{C}) = \log_{10}(\frac{N}{n_t})$
  - $\rightsquigarrow$ Upweights words that appear in fewer documents
- TF-IDF for a single word: $\log_{10}(f_{t\mathcal{D}_i} + 1) \times \log_{10}(\frac{N}{n_t})$
  - $\rightsquigarrow$ The more frequently a word appears and the fewer times it appears in a document, the higher the TF-IDF score
- Why do we use the $\log_{10}(\cdot)$?

See you next week!

Topic 2: Text

Computational Analysis of Text, Audio, and Images, Fall 2023

Aarhus University

References i