Representing linguistic data

Statistical Methods in NLP 2
ISCL-BA-08

Çağrı Çöltekin
ccoltekin@sfs.uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

Summer Semester 2025

version: none @2025-05-09

Representing linguistic data

- Almost all machine learning methods require a fixed number of numeric predictors
- In most NLP task, our predictors are sequences of (categorical) units: letters/phonemes, words, sentences, documents
- Correct representations of these units have direct impact on the success of the ML methods
- We start with some basic methods of representations, and move on to *learning* these representations

Basics: how to represent categorical predictors?

binary case

- Representing binary predictors is relatively straightforward

$$x = \begin{cases} 0 & \text{for male} \\ 1 & \text{for female} \end{cases}$$

$$x = \begin{cases} -1 & \text{positive} \\ +1 & \text{negative} \end{cases}$$

- 0 and 1 is the most common choice in practice
- The choice of the numbers is arbitrary, any two real numbers would do
- The order is also arbitrary, but sometimes good to pick a particular order for interpretability

Categorical predictors

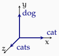
with more than two categories

- For categorical variables most common choice is the *one-hot* (or *one-of-k*) representation
- One-hot vectors are orthogonal, they do not indicate any (random) similarity or ordering
- Even for ordinal variables, it may often be better to use one-hot vectors
- They cannot represent any similarities between class values
- The vectors may become large if there are many values

POS tag	code
NOUN	00001
VERB	00010
ADJ	00100
ADV	01000
PRON	10000

Problem with one-hot representations

$$\begin{aligned} \text{cat} &= (0, \ldots, 1, 0, 0, \ldots, 0) \\ \text{cats} &= (0, \ldots, 0, 1, 0, \ldots, 0) \\ \text{dog} &= (0, \ldots, 0, 0, 1, \ldots, 0) \\ &\ldots \end{aligned}$$



- No notion of similarity
- It is particularly bad if the instances are not a closed class (e.g., documents)
- Large (but sparse) vectors

Properties of good representations

- Representations should allow measuring similarity: similar objects should be represented with similar vectors
- Representations should allow making distinctions required for the task
- Representations should be efficient to process
- Representations should be easy to obtain (automatically)

Bag of words (BoW) representation

The idea: use words that occur in text as features without paying attention to their order.

The document

It's a good thing most animated sci-fi movies come from Japan, because "titan a.e." is proof that Hollywood doesn't have a clue how to do it. I don't know what this film is supposed to be about.

BoW representation

how japan good n't I thing film what , proof titan a because . 's know does most hollywood is animated it do sci-fi a.e. " " supposed be come clue to this that from have movies about .

Bag of words representation

with binary features

The document

It's a good thing most animated sci-fi movies come from Japan, because "titan a.e." is proof that Hollywood doesn't have a clue how to do it. I don't know what this film is supposed to be about.

- If the word is in the document, the value of 1, otherwise 0
- The feature vector contains values for all words in our document collection

feature	value
to	1
do	1
a	1
thing	1
have	1
good	1
be	1
clue	1
great	0
pathetic	0
masterpiece	0
…	

Bag of words representation

with (document) frequencies

The document

It's a good thing most animated sci-fi movies come from Japan, because "titan a.e." is proof that Hollywood doesn't have a clue how to do it. I don't know what this film is supposed to be about.

- Use frequencies rather than binary vectors
- May help in some cases, but
  - effect of document length
  - frequent is not always good

feature	value
to	2
do	2
a	2
thing	1
have	1
good	1
be	1
clue	1
great	0
pathetic	0
masterpiece	0
…	

Bag of words representation

with relative frequencies

The document

It's a good thing most animated sci-fi movies come from Japan, because "titan a.e." is proof that Hollywood doesn't have a clue how to do it. I don't know what this film is supposed to be about.

- Relative frequencies are less sensitive to document length
- Still, high-frequency words dominate

feature	value
to	0.06
do	0.06
a	0.06
thing	0.03
have	0.03
good	0.03
be	0.03
clue	0.03
great	0.00
pathetic	0.00
masterpiece	0.00
…	

tf-idf weighting

- Intuition:
  - Words that appear multiple times in a document is important/representative for the document
  - Words that appear in many documents are not specific/useful
- tf-idf uses two components
  - tf term frequency - frequency of the word in the document
  - idf inverse document frequency - inverse of the ratio of documents that contain the term
- Both components are typically normalized

$$\text{tf-idf}_{t,d} = \frac{C_{t,d}}{|d|} \times \log \frac{N}{n_t}$$

term count in doc ($C_{t,d}$); doc length ($|d|$); number of docs (N); number of docs with t (n_t)

tf-idf example

Document 1 (d_1)

the	5
good	2
bad	1

Document 2 (d_2)

the	2
a	2
book	1

Document 3 (d_3)

the	1
a	2
good	3

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \times \text{idf}(t)$$

$$\text{tf-idf}(\text{good}, d_1) = ?\frac{2}{8} \times \log \frac{3}{2} = 0.15 \qquad \text{tf-idf}(\text{the}, d_1) = ?\frac{5}{8} \times \log \frac{3}{3} = 0.00$$

$$\text{tf-idf}(\text{bad}, d_1) = ?\frac{1}{8} \times \log \frac{3}{1} = 0.20 \qquad \text{tf-idf}(\text{good}, d_3) = ?\frac{3}{6} \times \log \frac{3}{2} = 0.29$$

Some notes on tf-idf

- tf-idf is an effective method for term weighting
- It was originally used for information retrieval, where it brought substantial improvements over other methods
- It is also very effective on text classification when using linear models
- There are some alternatives (e.g., BM25), and many variations: frequencies for TF, or use the log TF
- It has been difficult to improve over it (since 1970's)

Pointwise mutual information

for term weighting

- Another common weighting method is pointwise mutual information

$$\mathrm{PMI}(\mathrm{t}, \mathrm{d}) = \log \frac{\mathrm{P}(\mathrm{t}, \mathrm{d})}{\mathrm{P}(\mathrm{t})\mathrm{P}(\mathrm{d})}$$

- Besides normalizing for 'term frequency/probability', PMI also takes the 'document probability' into account
- Note that 'document' does not have to be a document, any definition of 'context' may result in useful representations (depending on the task)

A document is more than a BoW

The example document for sentiment analysis

It's a good thing most animated sci-fi movies come from Japan, because "titan a.e." is proof that Hollywood doesn't have a clue how to do it. I don't know what this film is supposed to be about.

- So far, we considered documents as simple BoW words
- BoW representations is surprisingly successful in many fields (IR, spam detection, ...)
- However, word order matters
  - According to a sentiment dictionary, our example contains one positive and one negative word

A document is more than a BoW

The example document for sentiment analysis

It's a good thing most animated sci-fi movies come from Japan, because "titan a.e." is proof that Hollywood doesn't have a clue how to do it. I don't know what this film is supposed to be about.

- So far, we considered documents as simple BoW words
- BoW representations is surprisingly successful in many fields (IR, spam detection, ...)
- However, word order matters
  - According to a sentiment dictionary, our example contains one positive and one negative word
- Paying attention to longer sequences allows us to get better results

Bag of n-grams

- Using n-grams rather than words allows us to capture more information in the data
- We can still use the same weighting methods (tf-idf)
- It is a common practice to use a range of (overlapping) n-grams
- This results in large set of features (millions for most practical applications)
- Data sparsity is a problem for higher order n-grams

The unreasonable effectiveness of character n-grams

An example document

It's a good thing ...

feature	value
`it␣`	2
`t'`	1
`'s`	2
`s␣a`	3
`␣a␣`	4
`a␣.`	5
`␣g`	2
`it'`	2
`t's␣a`	2
`'s␣a␣`	1
`s␣a␣g`	1
`␣a␣gо`	2
`a␣gоо`	2
...	

- For a number of text classification tasks (authorship attribution, language detection), character n-gram features found to be effective
- The idea is to use a range of character n-grams

$$\begin{array}{c} \\ \text{cat} \\ \text{dog} \\ \text{book} \\ \dots \end{array} \begin{array}{c} \begin{array}{cccc} \mathrm{d}_1 & \mathrm{d}_2 & \mathrm{d}_3 & \dots \end{array} \\ \begin{bmatrix} 0 & 3 & 1 & \dots \\ 0 & 0 & 3 & \dots \\ 4 & 1 & 4 & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix} \end{array} \begin{array}{c} \mathrm{d}_n \\ \begin{bmatrix} 4 \\ 3 \\ 5 \\ \dots \end{bmatrix} \end{array}$$

- Rows of the matrix represent words: words that appear in the same set of documents will be similar to each other
- The columns represent documents: documents with overlapping sets of words will be similar to each other
- Terms do not have to be words, any sequence we can count can be a term
- Contexts do not have to be documents, and any meaningful context can be used instead of documents
- Data is highly correlated (lots of redundancy)
- For practical applications we need huge (but sparse) matrices

A toy example

A four-sentence corpus with *bag of words* (BOW) model.

The corpus:

S1: She likes cats and dogs
S2: He likes dogs and cats
S3: She likes books
S4: He reads books

Term-document (sentence) matrix

	S1	S2	S3	S4
she	1	0	1	0
he	0	1	0	1
likes	1	1	1	0
reads	0	0	0	1
cats	1	1	0	0
dogs	1	1	0	0
books	0	0	1	1
and	1	1	0	0

A toy example

A four-sentence corpus with *bag of words* (BOW) model.

The corpus:

S1: She likes cats and dogs
S2: He likes dogs and cats
S3: She likes books
S4: He reads books

Term-term (left-context) matrix

	w	she	he	likes	reads	cats	dogs	books	and
she	2	0	0	0	0	0	0	0	0
he	2	0	0	0	0	0	0	0	0
likes	0	2	1	0	0	0	0	0	0
reads	0	0	1	0	0	0	0	0	0
cats	0	0	0	1	0	0	0	0	1
dogs	0	0	0	1	0	0	0	0	1
books	0	0	0	1	1	0	0	0	0
and	0	0	0	0	0	1	1	0	0

What about the linguistic features?

- Linguistically-informed representations is one potential area where linguistics can help building NLP system
- For text classification, the use of 'lexicons' has been common
- Other linguistic features such as
  - lemmas
  - sequences of POS tags
  - parser output: dependency triplets, or partial trees

  are also used in some tasks

A simple example from phonetics/phonology

	Vowel	High	Back	Round	Voice	Labial	Nasal	...
i	1	1	0	0	1	0	0	...
ɑ	1	0	1	0	1	0	0	...
n	0	0	0	0	1	0	1	...
m	0	0	0	0	1	1	1	...
p	0	0	0	0	0	1	0	...
b	0	0	0	0	1	1	0	...

- Compared to one-hot representations, these type of representations help identifying similar units

Are linguistic features useful at all?

- It is often difficult to get improvements over simple features
- It also makes systems more complex and language dependent
- Linguistic features can particularly be useful if the amount of data is limited
- They are particularly interesting for interpretable and explainable ML/NLP
- Considering the types of linguistics features that help is useful for structuring your input

Final remarks

- Representation of inputs to a ML model is important
- More meaningful/useful representations are likely to improve the systems
- Modern ML methods learn these representations from the data
- Informed/clever ways to represent the data may still be important in some cases (e.g., low-resource scenarios)

Next:

Fri/Mon Learning representations

Some sources of information

- Jurafsky and Martin (Chapter 6, 2025)

Jurafsky, Daniel and James H. Martin (2025). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. Online manuscript released January 12, 2025. URL: https://web.stanford.edu/~jurafsky/slp3/.