Embedding

LING83600
Kyle Gorman
Graduate Center, City University of New York

Machine learning systems used in speech and language processing employ linguistic units like words, phonemes, n-grams, etc., as multinomial features. Each multinomial feature with k levels can then *one-hot encoded* as a binary vector of length $k-1$. For example, a multinomial variable ranging over $\{\text{dc}, \text{lower}, \text{mixed}, \text{title}, \text{upper}\}$. can be exactly encoded using $\{0, 1\}^4$:

$$
\begin{aligned}
\text{dc} &\longrightarrow [0, 0, 0, 0] \\
\text{lower} &\longrightarrow [1, 0, 0, 0] \\
\text{mixed} &\longrightarrow [0, 1, 0, 0] \\
\text{title} &\longrightarrow [0, 0, 1, 0] \\
\text{upper} &\longrightarrow [0, 0, 0, 1]
\end{aligned}
$$

An *embedding* is a function which map words (etc.) onto vectors of real numbers. That is, an embedding is a function $V \times \mathbb{R}^k$ where V is the vocabulary and k is a hyperparameter. These real number vectors are an alternative to the sparse boolean vectors produced by one-hot encoding.

One can imagine an infinitude of embedding functions, including *random embeddings* that (deterministically) assign real vectors to each word (etc.). For an embedding to be useful—i.e., superior to a one-hot embedding—it needs to cluster words with similar linguistic behaviors/properties together.

The idea has been around for several decades but has become a very important component of neural network approaches to language processing. More generally, the subarea that studies how neural networks induce representations of linguistic data is sometimes called *representation learning*, and is the subject of two regularly-scheduled ACL workshops,

- the Workshop on Representation Learning for NLP (RepL4NLP) and
- Analyzing and interpreting neural networks for NLP (BlackboxNLP).

Introductory notions

The distributional hypothesis

The idea that we can understand words (or sentences, or documents) by analyzing co-occurrence statistics is sometimes known as the *distributional hypothesis*. Two obligatory quotes:

> *In other words, difference in meaning correlates with difference in distribution. (Harris 1954:43)*

> *You shall know a word by the company it keeps. (Firth 1957:11)*

But exactly what form should these representations take? And how should they be induced?

The bag of words model

One common—and unexpectedly effective—technique used to represent sentences or documents is the *bag of words* (BOW) representation. This can be as simple as counting all the words in a document.

```
>>> import collections
>>> bag = collections.Counter(tokens)
```

Text preparation

Before counting tokens in this fashion, one may wish to

- tokenize into sentences and tokens,
- remove stop-words,
- lemmatize or stem, and
- case-fold (though see Church 1995).

When working with a large collection of documents, one may also wish to

- deduplicate the document collection,
- exclude very long and very short documents, and
- exclude documents containing unexpected characters.

Term-document and word co-occurrence analysis

Term-document matrix example (after J&M, §6.3)

	As You Like It	*Twelfth Night*	*Julius Caesar*	*Henry V*
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

In-memory, we prefer a sparse representation:

```
{"As You Like It": [(1, 1), (2, 114), (3, 36), ...], ...}
```

From term-document matrices to vectors

We can think of

- the column vector [1, 114, 36, 20] as a 4d "representation" of the document *As You Like It*, and
- the row vector [36, 58, 1, 4] as a 4d "representation" of the token *food*.

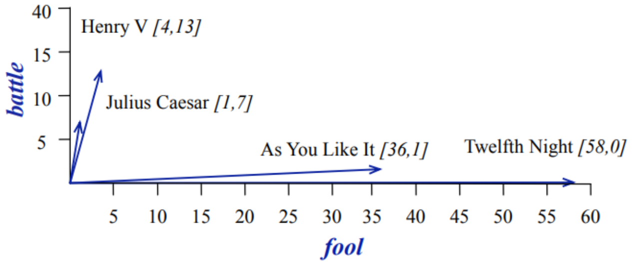

Figure: The words *battle* and *fool* in four works of Shakespeare (after J&M, §6.3).

Term weighting: motivations

Raw token frequencies are often less informative than we'd like, because

- ubiquitous words tend to carry little information (particularly function words), and
- words that occur in many documents tend to bear less information than words that occur in few documents.

E.g., as Church (2000) notes, not many documents mention *Noriega*, but those that do are in some sense "about" Noriega.

Term frequency

Term frequency, denoted $tf_{t,d}$, is the number of times a token t occurs in document d. It is often computed in log-space as

$$\log tf_{t,d} = \log(c_d(t) + 1)$$

or

$$\log tf_{t,d} = \begin{cases} 1 + \log c_d(t) \text{ if } c_d(t) > 0 \\ 0 \end{cases} .$$

Document frequency

Document frequency, denoted df_t, is the number of documents a token t occurs in. *Inverse document frequency*, idf_t, is this quantity scaled by the number of documents N. This is also often computed in log-space, as

$$\begin{aligned} \log idf_t &= \log(\frac{N}{df_t}) \\ &= \log N - \log df_t \end{aligned}$$

TF-IDF weighting

These statistics are often combined for a given term to give *term frequency-inverse document frequency* (TF-IDF):

$$tdidf_{t,d} = \log tf_{t,d} + \log idf_t$$

Using this instead of the raw frequencies tends to give more informative representations of a term's affiliation for a document.

TF-IDF term-document matrix example

	As You Like It	*Twelfth Night*	*Julius Caesar*	*Henry V*
battle	.07	.00	.22	.28
good	.00	.00	.00	.00
fool	.02	.02	.00	.01
wit	.05	.04	.02	.02

Word co-occurrence matrix example (after J&M, §6.3)

	aardvark	*computer*	*data*	*result*
cherry	0	2	8	9
strawberry	0	0	0	1
digital	0	1,679	1,683	85
information	0	3,325	3,982	378

Two minor implementational challenges here are

- to avoid counting (w_i, w_j) and (w_j, w_i) separately, and
- to avoid counting (w_i, w_i).

Pointwise mutual information: motivations

Raw co-occurrence frequencies are often less informative than we'd like, because ubiquitous, low-information words tend to co-occur with each other. We'd rather ask whether a word is particularly associated with another word.

Pointwise mutual information: definitions

Pointwise mutual information (Church and Hanks 1990) $PMI(w_i, w_j) : W \times W \longrightarrow \mathbb{R}$ is given by

$$
\begin{aligned}
PMI(w_i, w_j) &= \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \\
&= \log P(w_i, w_j) - \log P(w_i) - \log P(w_j).
\end{aligned}
$$

Positive pointwise mutual information

PMI has the range $(-\infty, \infty)$, but negative values are somewhat strange: they indicate that two words occur less often than we might expect by chance. In positive pointwise mutual information (PPMI), we simply replace negative PMI values with 0, thus

$$PPMI(w_i, w_j) = \max(PMI(w_i, w_j), 0)$$

PPMI examples (after J&M, §6.3)

	computer	*data*	*pinch*	*sugar*
apricot			2.25	2.25
pineapple			2.25	2.25
digital	1.66			
information		0.57		