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*.

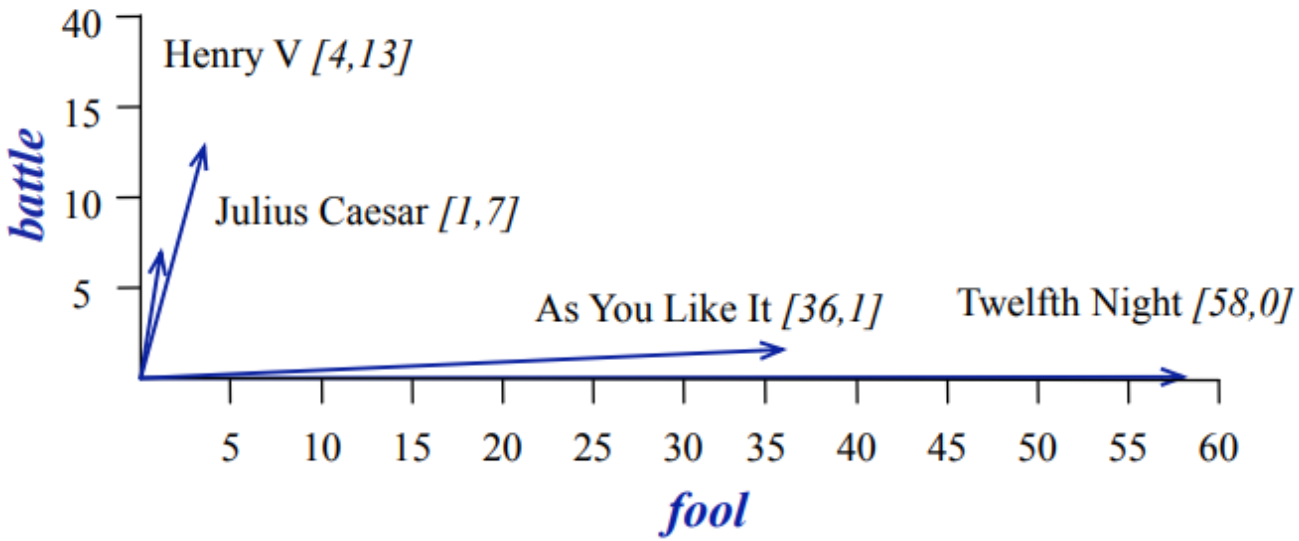

Figure: The words *battle* and *fool* in four works of Shakespeare (after J&M, §6.3).

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		

Phrase mining

The meanings of some frequent phrases (e.g., *Air Canada*) cannot be directly discerned from their constituents (i.e., they are not fully *compositional*). Mikolov et al. (2013) propose a simple method to induce "phrases". Given a sequence of two tokens $w_i w_j$, let us define the *phrase score* $ps(w_i, w_j) : W \times W \longrightarrow \mathbb{R}$ such that

$$ps(w_i, w_j) = \frac{c(w_i w_j) - \delta}{c(w_i)c(w_j)}$$

where $\delta \in \mathbb{R}^+$ is a user-specified hyperparameter. Bigrams whose phrase score exceeds a certain threshold are treated as a single token (e.g., `Air_Canada`) for subsequent processing. Implementation of this and related methods can be found in the `phrases` module of Gensim (Řehůřek and Sojka 2010).

Contexts

We can define contexts however we wish. Consider the sentence *The moment one **learns** English, complications set in.*

Brown clusters	{*one*}
Word2Vec, $h = 2$	{*moment*, *one*, *English*, *complications*}
Structured word2vec, $h = 2$	{(*moment*, −2), (*one*, −1), …}
Dependency contexts	{(*one*, nsubj), (*English*, dobj), …}

Table: Example contexts, after Eisenstein (2019:312).

Induced word embeddings

Cosine similarity: motivation

Embeddings represent words as (or embed words in) k-dimensional vectors of real numbers ($\mathbb{R}^k$). Between any two vectors of the same dimensionality, there is an (acute) angle. We can use the magnitude of this angle to measure two vectors' "similarity".

Cosine similarity: definition

Cosine similarity is defined as

$$\cos(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|_2 \times \|\mathbf{v}_2\|_2}$$

where $\mathbf{v}_1$ and $\mathbf{v}_2$ are two vectors, $\cdot$ is the dot product (e.g., $\sum_i v_{1,i} \times v_{2,i}$), and $\|\mathbf{v}\|_2$ is the Euclidean norm of $\mathbf{v}$ (e.g., $\sqrt{\sum_i v_i^2}$). Like the cosine of an acute angle, this ranges from $[0, 1]$ and takes its maximal value when the angle is $0°$.

Embedding as matrix factorization

Given a matrix $\mathbf{C} = \{c_{i,j}\}$ storing the co-occurrence statistics for a target i and a context j, we can obtain embeddings by approximating $c_{i,j}$ with a function of a target embedding $\mathbf{u}_i$ and a context embedding $\mathbf{v}_j$. That is, we wish to find embeddings such that minimize the reconstruction error

$$\min_{\mathbf{u},\mathbf{v}} \|\mathbf{C} - \widetilde{\mathbf{C}}(\mathbf{u}, \mathbf{v})\|_2$$

where $\widetilde{\mathbf{C}}(\mathbf{u}, \mathbf{v})$ is the reconstruction of the count matrix $\mathbf{C}$ given by embeddings $\mathbf{u}$ and $\mathbf{v}$.

Latent semantic analysis

One of the best-known examples of this approach is known as *latent semantic analysis* (LSA) or *latent semantic indexing* (Deerwester et al. 1990). Let V be the number of terms, K the size of the desired embedding, and $|C|$ the number of contexts. Then $\widetilde{\mathbf{C}}$ is given by $\mathbf{U}\,\mathbf{S}\,\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{V \times K}$, $\mathbf{S} \in \mathbb{R}^{K \times K}$, and $\mathbf{V} \in \mathbb{R}^{|C| \times K}$. Thus each element of $\mathbf{C}$ $c_{i,j}$ is given by $\sum_{k=1}^{K} u_{i,k} s_{k,k} v_{j,k}$. Finding $\mathbf{U}\,\mathbf{S}\,\mathbf{V}^\top$ approximating $\mathbf{C}$ is a well-studied problem known as *singular value decomposition* and is implemented in many linear algebra libraries.

LSA demo

Word2Vec

The highly-effective CBOW and skipgram embedding methods (together known as `word2vec`) were proposed by Mikolov et al. (2013). They find the skipgram model is generally more effective in intrinsic evaluation.

Continuous bag-of-words: motivations

The *continuous bag-of-words* (CBOW) model represents the local context for a target word w_m using the average of the embeddings of words in the window $m - h, m - h + 1, \ldots, m - 1, m + 1, \ldots, m + h - 1, m + h$ where h is the size of the window. This is *continuous* because we condition on a continuous vector constructed from the embeddings, and it is a *bag-of-words* model because we discard positional information within the window and ignore words outside the window.

Continuous bag-of-words: definition

The context embedding for target position m is given by

$$\begin{aligned}\bar{\mathbf{v}}_m &= \frac{1}{2h}\sum_{n=1}^{h}\mathbf{v}_{m-n} + \mathbf{v}_{m+n} \\ &= \frac{1}{h}\sum_{n=1}^{h}\mathbf{v}_{m-n} + \frac{1}{h}\sum_{n=1}^{h}\mathbf{v}_{m+n}\end{aligned}$$

where $\mathbf{v}$ are the term embeddings.

Continuous bag-of-words: optimization

We then optimize **u** and **v** to approximate the log-likelihood of the corpus frequencies

$$\log p(\mathbf{w}) \approx \sum_{m=1}^{M} \log p(w_m \mid m-h, \ldots, m+h)$$
$$= \sum_{m=1}^{M} \log \frac{\exp(\mathbf{u}_{w_m} \cdot \bar{\mathbf{v}}_m)}{\sum_{j=1}^{V} \exp(\mathbf{u}_j \cdot \bar{\mathbf{v}}_m).}$$

Skipgrams

In the CBOW formulation, we predict target words given their context. In the *skipgrams* model, we predict contexts given words. Let h_m be a randomly sampled neighborhood size sampled from the uniform distribution $\{1, 2, \ldots, h\}$. Then we optimize $\mathbf{u}$ and $\mathbf{v}$ to approximate the log-likelihoods of the corpus frequencies

$$\begin{aligned}\log p(\mathbf{w}) &\approx \sum_{m=1}^{M} \sum_{n=1}^{h_m} \log p(w_{m-n} \mid w_m) + \log p(w_{m+n} \mid w_m) \\ &= \sum_{m=1}^{M} \sum_{n=1}^{h_m} \log \frac{\exp(\mathbf{u}_{w_{m-n}} \cdot \mathbf{v}_{w_m})}{\sum_{j=1}^{V} \exp(\mathbf{u}_j \cdot \mathbf{v}_{w_m})} + \log \frac{\exp(\mathbf{u}_{w_{m+n}} \cdot \mathbf{v}_{w_m})}{\sum_{j=1}^{V} \exp(\mathbf{u}_j \cdot \mathbf{v}_{w_m})}\end{aligned}$$

Alternatives

- Brown et al. (1992) clusters are discussed in the Eisenstein text at length.
- GloVe (Pennington et al. 2014) attempts to directly approximate co-occurrence probabilities, compressing them into embeddings using least-squares.
- fastText (Bojanowski et al. 2017) adds several new tricks to the CBOW approach, including learned positional weights and character n-gram vectors (extracted from the target word) concatenated to the bag of words.

Software comparison

- `word2vec`: Apache 2.0 license, installs C command-line tool word2vec, abandonware
- `gensim.models.Word2Vec`: GNU LGPL 2.1 license, Python interface, great documentation, a bit inefficient
- GloVe: Apache 2.0 license, installs C command-line tool `glove` (and friends), "abandonware"
- fastText: MIT license, installs C++ command-line tool `fasttext`, also has a handy "supervised mode" for simple document classification problems

Pretrained embeddings

There are many pre-trained embeddings available (usually in a simple tabular format), e.g.:

- word2vec: 100 billion English words from Google News
- GloVe: 27 billion English words from Twitter
- GloVe: 840 billion English words from the Common Crawl
- fastText: Data from 157 languages from the Common Crawl and Wikipedia

But, training embeddings for your language, domain, and/or data is easy, reasonably fast, and requires no specialized hardware. And, embeddings trained on small amounts of relevant data may be very effective.

Contextual embeddings: motivations

Newer *contextual embedding* models like ELMo (Peters et al. 2018) and BERT (Devlin et al. 2019) do not give us a single vector for each word type, but rather a separate vector for each word token which also takes that words' context into account. However, the creation of such models require an enormous amount of computing resources (and power, and CO_2 emissions) to train from scratch (Strubell et al. 2019). For these, one is strongly recommended to make use of pre-trained models, e.g.:

- `bert-base-cased`: cased English text
- `bert-large-uncased`: case-folded English text (large model)
- `bert-base-multilingual-cased`: cased text in 104 languages
- CamemBERT: cased multi-genre French text
- `rubert-base-cased`: cased Russian Wikipedia and news text
- …

References I

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

P. Brown, P. deSouza, R. Mercer, V. della Pietra, and J. Lai. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4): 467–479, 1992.

K. W. Church. One term or two? In *Proceedings of the 18th Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 310–318, 1995.

K. W. Church. Empirical estimates of adaptation: the chance of two Noriegas is closer to p/2 than p2. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*, pages 180–186, 2000.

References II

K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

J. Eisenstein. *Introduction to Natural Language Processing*. MIT Press, 2019.

J. R. Firth. A synopsis of linguistic theory 1930–1955. In F. R. Palmer, editor, *Selected Papers of J. R. Firth*, pages 1–32. Longman, London, 1957.

References III

Z. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. 2013.

J. Pennington, R. Socher, and C. Manning. GloVe: global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

References IV

E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

R. Řehůřek and P. Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, 2010.