Chapter NLP:III

III. Text Models

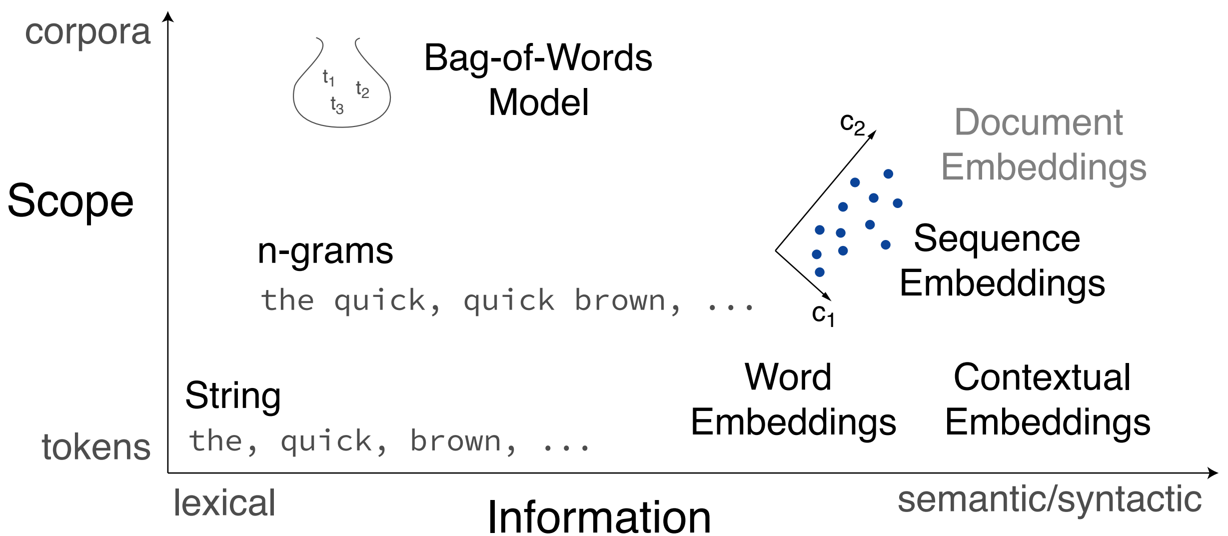
- ❑ Text Preprocessing
- ❑ Text Representation
- ❑ Text Similarity
- ❑ Text Classification
- ❑ Language Modeling
- ❑ Sequence Modeling

Text Representation

Models of Representation

Language computation requires different text models, depending on application.

- Common are character sequences, indices of a vocabulary, and vectors.
- The representation determines and is constrained by
  1. the preserved information. lexical, semantic, syntactic, ...
  2. the (computationally feasible) scope of modeled text.



Text Representation

Token Representations

Tokens can be naively modeled with standard data structures.

1. As strings, sequences of characters.

   `The quick brown fox jumps over the lazy dog`

2. As indices from an ordered vocabulary V. This loses all lexical similarity.

   $V = \{\texttt{a}^{1}, \texttt{aardvark}^{2}, \ldots, \texttt{fox}^{1386}, \ldots, \texttt{zebra}^{10000}\}$

   $\texttt{a} = 1 \quad \texttt{fox} = 1386 \quad \texttt{the} = 8992$

3. As one-hot vectors, all-zero vectors of length $|V|$, with a 1 at the token's index.

$$V = \begin{pmatrix} \texttt{a} \\ \texttt{aardvark} \\ \ldots \\ \texttt{fox} \\ \ldots \\ \texttt{zebra} \end{pmatrix} \qquad \texttt{a} = \begin{pmatrix} 1 \\ 0 \\ \ldots \\ 0 \\ \ldots \\ 0 \end{pmatrix} \qquad \texttt{fox} = \begin{pmatrix} 0 \\ 0 \\ \ldots \\ 1 \\ \ldots \\ 0 \end{pmatrix}$$

Text Representation

Document Representation

Documents can also be modeled as strings or vocabulary indices. This has a number of computational disadvantages:

- Documents are variable in length.

  Most methods like classification, clustering, or retrieval assume a fixed size input. Truncating or padding the sequences is less effective with very long sequences.

- Basic operations are computationally expensive on sequences.
  - Identity is at least $O(n)$.
  - Similarity is at least $O(m \times n)$. [NLP:II ff.]
  - How to find the most similar documents in a corpus?

- Lists of token indices are not suited as feature vectors. [NLP:II ff.]

  For a machine learning model, the (intuitive) interpretation would be: The document n has word w_i at position j, which would create a very sparse feature space.

Document Representation: Bag of Words Metaphor

Idea: The frequency of tokens is enough to model the content of a document. A document is just a bag of words (BoW).

- Word order is less important than storage space or computational cost.
- *The frequencies of words in a document tend to indicate the relevance of the document to a query* [Turney, Pantel 2010]

Example from Biden's inaugural speech (2020):

```
america (14)
nation (8)
story (7)
people (7)
democracy (7)
world (6)
unity (6)
stand (6)
...
```

Text Representation

Document Representation: Vector Space Model [Salton et. al. 1975]

Idea: Model documents $\mathbf{d_i} \in D$ as bags of words – vectors over a vocabulary $|V|$ and collections as a $|D| \times |V|$ Document-Term-Matrix (DTM).

- Term-frequency vectors (*tf*(t, d_i)): the absolute frequency of a term t in a document d_i. Also called count vectors; Often also normalized for document length.
- Term-weighted vectors (*tf*$(t, d_i) \cdot$ *idf*(t, D)): the "importance" of a term.

	tf(t, d_i)			*tf*$(t, d_i) \div \|d_i\|$			*tf* · *idf*		
V	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}
`a`	47	15	49	.019	.010	.019			
`america`	8	19	19	.003	.013	.007			
`country`	2	9	4	.001	.006	.002			
`great`	0	6	6	.000	.004	.002			
`nation`	1	6	13	.000	.004	.005			
`people`	7	10	11	.003	.007	.004			
`story`	0	0	9	.000	.000	.003			
`work`	6	0	6	.002	.000	.002			
`world`	6	6	8	.002	.004	.003			
...									
Length	2,395	1,433	2,540						

Text Representation

Document Representation: Vector Space Model [Salton et. al. 1975]

Idea: Model documents $\mathbf{d_i} \in D$ as bags of words – vectors over a vocabulary $|V|$ and collections as a $|D| \times |V|$ Document-Term-Matrix (DTM).

- Term-frequency vectors ($tf(t, d_i)$): the absolute frequency of a term t in a document d_i. Also called count vectors; Often also normalized for document length.
- Term-weighted vectors ($tf(t, d_i) \cdot idf(t, D)$): the "importance" of a term.

	$tf(t, d_i)$			$tf(t, d_i) \div \lvert d_i \rvert$			$tf \cdot idf$		
V	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}
`a`	47	15	49	.019	.010	.019			
`america`	8	19	19	.003	.013	.007			
`country`	2	9	4	.001	.006	.002			
`great`	0	6	6	.000	.004	.002			
`nation`	1	6	13	.000	.004	.005			
`people`	7	10	11	.003	.007	.004			
`story`	0	0	9	.000	.000	.003			
`work`	6	0	6	.002	.000	.002			
`world`	6	6	8	.002	.004	.003			
...									
Length	2,395	1,433	2,540						

Remarks:

- DTMs can become very large and very sparse (approx. 95% of elements are zero).
- DTMs can vary the elements (i.e. binary ($\mathbf{d_i}$ contains w_j) over counts), the words (n-grams over terms), or documents (sentences over documents).

- The set of index terms $T = \{t_1, \ldots, t_m\}$ is typically composed of the word stems of the vocabulary of a document collection, excluding stop words.
- The representation $\mathbf{d}$ of a document d is a $|T|$-dimensional vector, where the i-th vector component of $\mathbf{d}$ corresponds to a term weight w_i of term $t_i \in T$, indicating its importance for d. Various term weighting schemes have been proposed.

Text Representation

Term Weighting: *tf* · *idf*

To compute the weight w for a term t from document d under the vector space model, the most commonly employed term weighting scheme $\omega(t)$ is *tf* · *idf*:

- *tf* (t, d) denotes the normalized *term frequency* of term t in document d.
  The basic idea is that the importance of term t is proportional to its frequency in document d. However, t's importance does not increase linearly: the raw frequency must be normalized.

- *df* (t, D) denotes the *document frequency* of term t in document collection D.
  It counts the number of documents that contain t at least once.

- *idf* (t, D) denotes the *inverse document frequency*:

$$\mathit{idf}(t, D) = \log \frac{|D|}{\mathit{df}(t, D)}$$

  The importance of term t in general is inversely proportional to its document frequency.

A term weight w for term t in document $d \in D$ is computed as follows:

$$\omega(t) = \mathit{tf}(t, d) \cdot \mathit{idf}(t, D).$$

Text Representation

Term Weighting: *tf* · *idf*

To compute the weight w for a term t from document d under the vector space model, the most commonly employed term weighting scheme $\omega(t)$ is *tf* · *idf*:

- *tf* (t, d) denotes the normalized *term frequency* of term t in document d.
  The basic idea is that the importance of term t is proportional to its frequency in document d. However, t's importance does not increase linearly: the raw frequency must be normalized.

- *df* (t, D) denotes the *document frequency* of term t in document collection D.
  It counts the number of documents that contain t at least once.

- *idf* (t, D) denotes the *inverse document frequency*:

$$\mathit{idf}(t, D) = \log \frac{|D|}{\mathit{df}(t, D)}$$

  The importance of term t in general is inversely proportional to its document frequency.

A term weight w for term t in document $d \in D$ is computed as follows:

$$\omega(t) = \mathit{tf}(t, d) \cdot \mathit{idf}(t, D).$$

Text Representation

Term Weighting: *tf* · *idf*

To compute the weight w for a term t from document d under the vector space model, the most commonly employed term weighting scheme $\omega(t)$ is *tf* · *idf*:

- *tf*(t, d) denotes the normalized *term frequency* of term t in document d.
  The basic idea is that the importance of term t is proportional to its frequency in document d. However, t's importance does not increase linearly: the raw frequency must be normalized.

- *df*(t, D) denotes the *document frequency* of term t in document collection D.
  It counts the number of documents that contain t at least once.

- *idf*(t, D) denotes the *inverse document frequency*:

$$\mathit{idf}(t, D) = \log \frac{|D|}{\mathit{df}(t, D)}$$

  The importance of term t in general is inversely proportional to its document frequency.

A term weight w for term t in document $d \in D$ is computed as follows:

$$\omega(t) = \mathit{tf}(t, d) \cdot \mathit{idf}(t, D).$$

Text Representation

Term Weighting: *tf* · *idf*

To compute the weight w for a term t from document d under the vector space model, the most commonly employed term weighting scheme $\omega(t)$ is *tf* · *idf*:

- *tf* (t, d) denotes the normalized *term frequency* of term t in document d.
  The basic idea is that the importance of term t is proportional to its frequency in document d. However, t's importance does not increase linearly: the raw frequency must be normalized.

- *df* (t, D) denotes the *document frequency* of term t in document collection D.
  It counts the number of documents that contain t at least once.

- *idf* (t, D) denotes the *inverse document frequency*:

$$\mathit{idf}(t, D) = \log \frac{|D|}{\mathit{df}(t, D)}$$

  The importance of term t in general is inversely proportional to its document frequency.

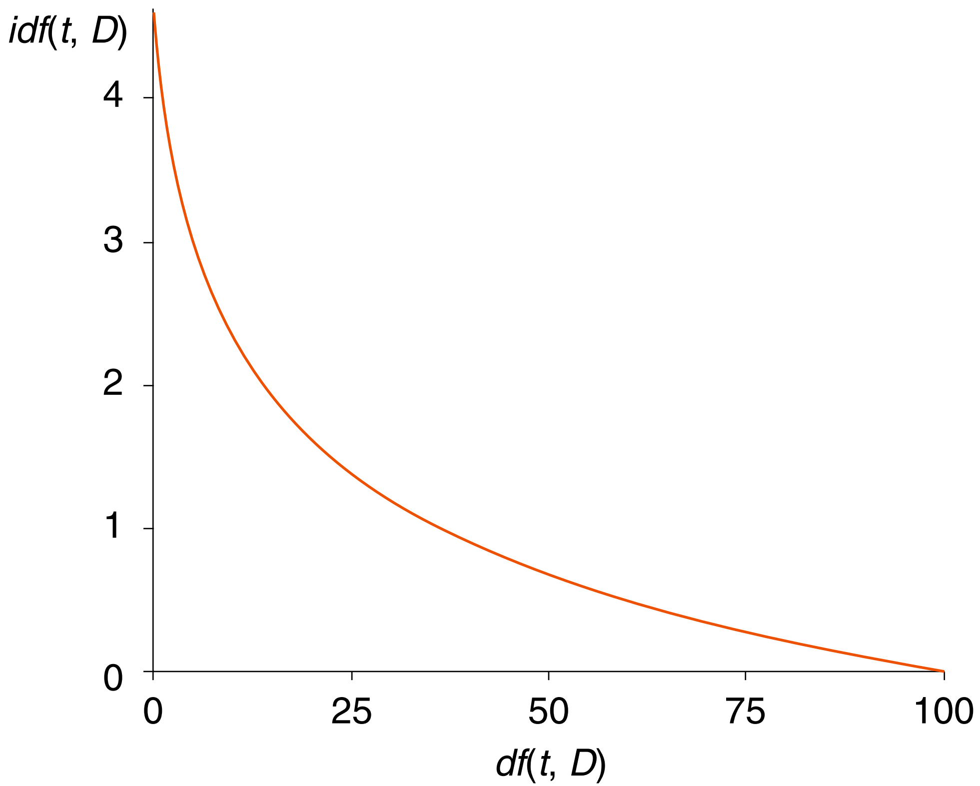
A term weight w for term t in document $d \in D$ is computed as follows:

$$\omega(t) = \mathit{tf}(t, d) \cdot \mathit{idf}(t, D).$$

Text Representation

Term Weighting: *tf* · *idf*

Plot of the function $\mathit{idf}(t, D) = \log \frac{|D|}{\mathit{df}(t, D)}$ for $|D| = 100$.



Text Representation

Term Weighting: *tf* · *idf* Example

$$idf(\texttt{a}, D) = \log \frac{|D|}{df(\texttt{a}, D)} = \log \frac{3}{3} = 0$$

$$tf \cdot idf(\texttt{a}, d_{\text{Obama}}) = 47 \cdot 0 = 0$$

$$tf \cdot idf(\texttt{a}, d_{\text{Trump}}) = 15 \cdot 0 = 0$$

$$tf \cdot idf(\texttt{a}, d_{\text{Biden}}) = 49 \cdot 0 = 0$$

Weighted DTM using tf

	$tf(t, d_i)$			$tf(t, d_i) \div \lvert d_i \rvert$			$tf \cdot idf$		
V	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}
`a`	47	15	49	.019	.010	.019	0	0	0
`great`	0	6	6	.000	.004	.002			
`story`	0	0	9	.000	.000	.003			

Text Representation

Term Weighting: *tf* · *idf* Example

$$idf(\texttt{great}, D) = \log \frac{|D|}{df(\texttt{great}, D)} = \log \frac{3}{2} = 0.41$$

$$tf \cdot idf(\texttt{great}, d_{\text{Obama}}) = 0 \cdot 0.41 = 0$$

$$tf \cdot idf(\texttt{great}, d_{\text{Trump}}) = 6 \cdot 0.41 = 2.46$$

$$tf \cdot idf(\texttt{great}, d_{\text{Biden}}) = 6 \cdot 0.41 = 2.46$$

Weighted DTM using tf

	$tf(t, d_i)$			$tf(t, d_i) \div \lvert d_i \rvert$			$tf \cdot idf$		
V	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}
`a`	47	15	49	.019	.010	.019	0	0	0
`great`	0	6	6	.000	.004	.002	0	2.46	2.46
`story`	0	0	9	.000	.000	.003			

Text Representation

Term Weighting: *tf* · *idf* Example

$$idf(\texttt{great}, D) = \log \frac{|D|}{df(\texttt{great}, D)} = \log \frac{3}{1} = 1.10$$

$$tf \cdot idf(\texttt{great}, d_{\text{Obama}}) = 0 \cdot 1.10 = 0$$

$$tf \cdot idf(\texttt{great}, d_{\text{Trump}}) = 0 \cdot 1.10 = 0$$

$$tf \cdot idf(\texttt{great}, d_{\text{Biden}}) = 9 \cdot 1.10 = 9.9$$

Weighted DTM using tf

	$tf(t, d_i)$			$tf(t, d_i) \div \lvert d_i \rvert$			$tf \cdot idf$		
V	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}
`a`	47	15	49	.019	.010	.019	0	0	0
`great`	0	6	6	.000	.004	.002	0	2.46	2.46
`story`	0	0	9	.000	.000	.003	0	0	9.9

Text Representation

Term Weighting: *tf* · *idf* Example

$$idf(\texttt{great}, D) = \log \frac{|D|}{df(\texttt{great}, D)} = \log \frac{3}{1} = 1.10$$

$$tf \cdot idf(\texttt{great}, d_{\text{Obama}}) = 0 \cdot 1.10 = 0$$

$$tf \cdot idf(\texttt{great}, d_{\text{Trump}}) = 0 \cdot 1.10 = 0$$

$$tf \cdot idf(\texttt{great}, d_{\text{Biden}}) = 0.003 \cdot 1.10 = 0.30$$

Weighted DTM using $tf \div |d_i|$

	$tf(t, d_i)$			$tf(t, d_i) \div \lvert d_i \rvert$			$tf \cdot idf$		
V	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}	d_{Obama}	d_{Trump}	d_{Biden}
`a`	47	15	49	.019	.010	.019	.0	.0	.0
`great`	0	6	6	.000	.004	.002	.0	.002	.001
`story`	0	0	9	.000	.000	.003	.0	.0	.030

Remarks:

- Term frequency weighting was invented by Hans Peter Luhn: “There is also the probability that the more frequently a notion and combination of notions occur, the more importance the author attaches to them as reflecting the essence of his overall idea.” [Luhn 1957]

- The importance of a term t for a document d is not linearly correlated with its frequency. Several normalization factors have been proposed [Wikipedia]:

  – $\mathit{tf}(t, d)/|d|$

  – $1 + \log(\mathit{tf}(t, d))$ for $\mathit{tf}(t, d) > 0$

  – $k + (1 - k)\dfrac{\mathit{tf}(t, d)}{\max_{t' \in d}(\mathit{tf}(t', d))}$, where k serves as smoothing term; typically $k = 0.4$

- Inverse document frequency weighting was invented by Karen Spärck Jones: “it seems we should treat matches on non-frequent terms as more valuable than ones on frequent terms, without disregarding the latter altogether. The natural solution is to correlate a term’s matching value with its collection frequency.” [Spärck Jones 1972]

- Spärck Jones gives little theoretical justification for her intuition. Given the success of *idf* in practice, over the decades, numerous attempts at a theoretical justification have been made. A comprehensive overview has been compiled by [Robertson 2004].

Text Representation

Vocabulary Pruning

- Vocabularies, even of small collections, can get very large. [NLP:II-20 ff.]
- This is often not desired, since DTM's become very sparse and lower the performance of learning methods. (cf. Curse of Dimensionality)

- Methods of limiting the vocabulary size:
  - Tokenization or stopping. [NLP:III-28 ff.]
  - Pruning: Prune the vocabulary V of a collection D by removing all types t_i with *tf*$(t_i, D) \notin [f_{min}, f_{max}]$, where f is the upper/lower pruning threshold. The threshold can be absolute or relative.

Text Representation

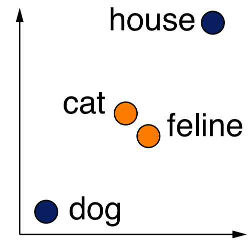
Distributional Representations of Words

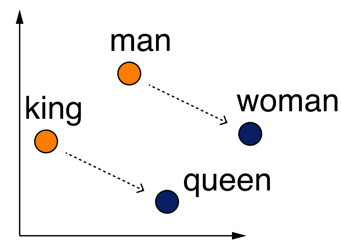
Distributional representations of words (Word Vectors) are embeddings of words in a latent space whose dimensions correspond to differences in word meaning.

- The vectors are dense and 'low-dimensional'.
- Semantically similar words have similar vectors.

  Similar vectors: `cat, feline`
- Similar vector difference implies similar semantic difference.

  Similar difference: `man` → `woman`   `king` → `queen`
- Vectors can be inferred without supervision or labels.





According to the Distributional hypothesis, modeling the (distribution of the) context of a word yields such representations.

You shall know a word by the company it keeps [Harris 1951, Firth 1957]

Text Representation

Distributional Representations of Words

Common approaches to compute word vectors:

- **Co-occurrence matrices.** [NLP:VI-6 ff.]
  $|V| \times |V|$ matrices which count how often a word j occurs in the vincinity of word i. Often combined with dimensionality reduction.

- **Skip-gram.**
  What is commonly called Word2Vec; Given a word, learn to predict it's context with a neural network. The fitted weights are the word vectors.

- **Continuous Bag-of-Words (CBOW).** [NLP:VI-10 ff.]
  Similar to skip-gram, but learn to predict the center word from the context.

- **GloVe.**
  Adds co-occurance matrices to the (skip-gram) model to encode global word statistics.

- **FastText.**
  Uses subwords (character sequences) instead of tokens; robust against noisy text.

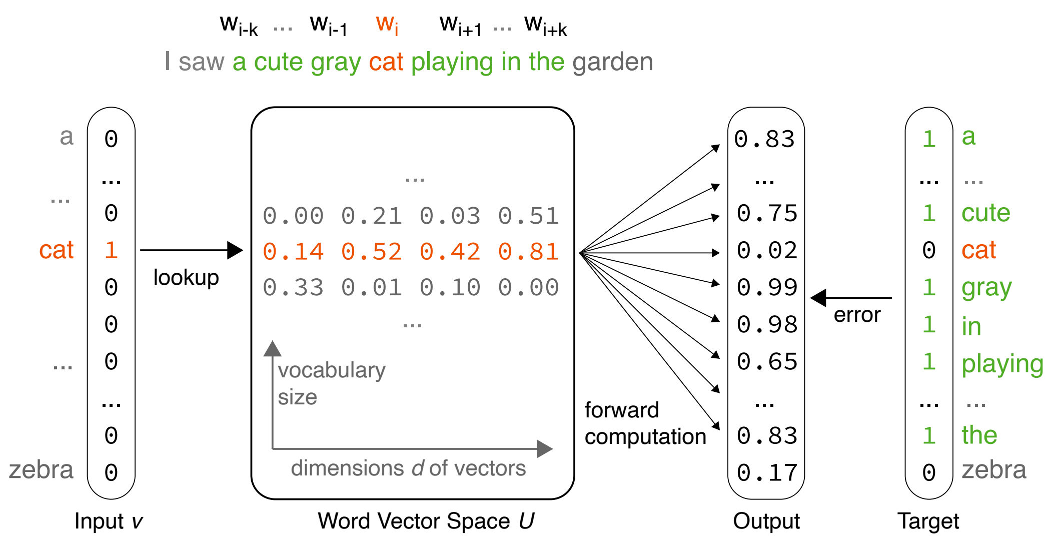
- **Contextual Embeddings from Transformers.**
  Encoding layer of a GPT or output layer of a BERT. Through attention mechanism, these embeddings also encode context-dependent variation of word meaning.

Text Representation

Word2Vec [Mikolov et al. 2013]

Idea: Learn the vectors by predicting the k-window context words from the center word (skip-gram). Model predicts similar contexts from similar vectors

- 2-layer feed forward neural network, trained with gradient descent.
- Input is a one-hot vector v of the center word w_i. $|V|, v_i = 1, v_j = 0, j \neq i$
- Hidden layer U is the word vector space. Row u_i is the word vector of w_i.
- Output vector is used to compute the error to the observed context.



Text Representation

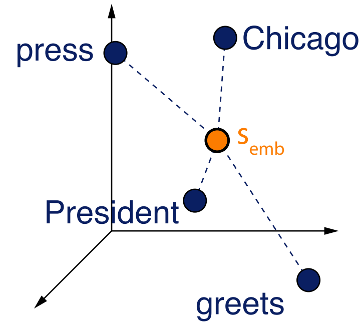
Sentence Embeddings [Iyyer et al. 2015]

❑ Vector Average

Compute a sentence embedding by averaging the word vectors of all tokens in the sentence.

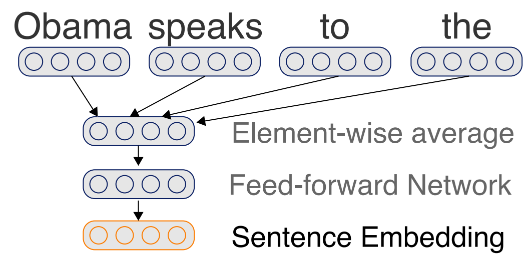
$s = \texttt{The President greets the press in Chicago}$

$$\mathbf{s}_{emb} = \frac{1}{|s|} \cdot \sum_{\mathbf{w}_i \in s} \mathbf{w}_i$$



❑ Deep Average Networks

Train a feed-forward neural network to predict the sentence embedding given the geometric average of the word vectors as input. Often trained on classification tasks (i.e. sentiment detection).



Text Representation

Sentence Embeddings [Cer et al. 2018, Reimers et al. 2019]

- Universal Sentence Encoder and Sentence-BERT

  Transformer-encoder have the same input and output size. The input is prepended with a special [CLS] token. The output vector of this token is used for the (sentence) classification part of the pre-training and often resembles a sentence embedding.