Machine Learning for Natural Language Processing

Representing text with vectors

Session 2

Benjamin Muller

INRIA Paris - ALMANACH
benjamin.muller@inria.fr

Outline of the course

1. The Why and What of Natural Language Processing
2. Representing text with vectors
3. Modeling textual data
4. Neural Natural Language Processing
5. Language Modelling
6. NLP in the "real-world"

Session Outline

How to represent textual data in a computer ? In this session we will focus mainly on how to represent a word.

Session Outline

How to represent textual data in a computer ? In this session we will focus mainly on how to represent a word.

- a word as an index
- feature based representation
- distributional representation
- continuous representation
- contextual representation (session 6)

Glossary

- In NLP, we call the basic string unit we work with **a token**
- a token can be a character, a sequence of characters, a word, ...
- In this session our basic unit will be a **word**

Preliminary Definition

Definition: *Words are the smallest linguistic expressions that are conventionally associated with a non-compositional meaning and can be articulated in isolation to convey semantic content* [1]

[1] Stanford Encyclopedia of Philosophy

Preliminary Definitions

In this course, we define words as:

- A **word** is the **basic unit** of discrete data, defined to be an item from a vocabulary indexed by $1, ..., V$.
- A document is a sequence of N words denoted by $d = (w_1, w_2, ..., w_N)$, where w_n is the n-th word in the sequence.
- A corpus is a collection of M documents denoted by $D = d_1, d_2, ..., d_M$

Other interesting basic units

- A character
- A sequence of characters or **n-grams** : *word-piece, morphemes...*
- a sequence of words : a sentence, a collection of sentences...

Some order of magnitudes

- 171,476 words in English (Oxford Dictionary)
- 123,000 words in French (Trésor de la Langue Française informatisé)
- 1,100,373 in Korean

Goal

Given a vocabulary w_1,..,w_V and a corpus D, our goal is to associate each word with a data structure

What do we want from this data structure ?

Goal

Given a vocabulary w_1,..,w_V and a corpus D, our goal is to associate each word with a data structure

What do we want from this data structure ?

- identify a word

Goal

Given a vocabulary w_1,..,w_V and a corpus D, our goal is to associate each word with a data structure

What do we want from this data structure ?

- identify a word
- capture the similarities of words (based on on morphology, syntax, semantic,...)

1-hot encoding

Reminder:

- Word types are element of a defined vocabulary
- Word tokens are instances of word types in text

Here, we want representations for word types

1-hot encoding

- Traditional way to represent words as **atomic symbols** with a unique integer is associated with each word:

$$\{1{=}\text{movie},\ 2{=}\text{hotel},\ 3{=}\text{apple},\ 4{=}\text{movies},\ 5{=}\text{art}\}$$

- Equivalent to represent words as **1-hot vectors**:

$$\begin{aligned}
\text{movie} &= [1, 0, 0, 0, 0] \\
\text{hotel} &= [0, 1, 0, 0, 0] \\
&\dots \\
\text{art} &= [0, 0, 0, 0, 1]
\end{aligned}$$

1-hot encoding

- Most basic representation of any textual unit in NLP. Always start with it.
- Implicit assumption: word vectors are an orthonormal basis
    - orthogonal ($x^T y = 0$)
    - normalized ($x^T x = 1$)

- Problem 1: Not very informative
    - Weird to consider “movie” and “movies” as independent entities
    - Or to consider all words equidistant:

$$\|\texttt{dog} - \texttt{cat}\| = \|\texttt{dog} - \texttt{moon}\|$$

- Problem 2: Polysemy
  Should the *mouse* of a computer get the same vector a the *mouse* animal?

Feature based representation

- Solution: represent words with **hand crafted features and relations**
- Example of potential features:
  - Morphology: prefix, suffix, stem...
  - Grammar: part of speech, gender, number,...
  - Shape: capitalization, digit, hyphen
- Example of potential relations:
  - synonyms,
  - hypernyms,
  - antonyms...

WordNet[2]

Definition: a (word) **sense** is a discrete representation of one aspect of the meaning of a word

WordNet is a large lexical database of word senses for English and other languages

[2] https://wordnet.princeton.edu/

WordNet

- Word types are grouped into synonym sets: synsets

$$\texttt{S09293800} = \{\ \textit{Earth}, \textit{earth}, \textit{world}, \textit{globe}\ \}$$

- **Polysemous** words: assigned to different synsets

$$\texttt{S14867162} = \{\ \textit{earth}, \textbf{ground}\ \}$$

- Contains **glosses** for synsets:

the 3rd planet from the sun; the planet we live on

- Noun/verb synsets: organized in hierarchy, capturing `IS-A` relation

apple `IS-A` *fruit*

WordNet: relations between synsets

- X is a hyponym of Y if X is an instance of Y:

  cat is a hyponym of animal

WordNet: relations between synsets

- X is a hyponym of Y if X is an instance of Y:

  cat is a hyponym of animal

- X is a hypernym of Y if Y is an instance of X:

  animal is a hypernym of cat

WordNet: relations between synsets

- X is a hyponym of Y if X is an instance of Y:

  cat is a hyponym of animal

- X is a hypernym of Y if Y is an instance of X:

  animal is a hypernym of cat

- X and Y are co-hyponyms if they have the same hypernym:

  cat and dog are co-hyponyms

WordNet: relations between synsets

- X is a hyponym of Y if X is an instance of Y:

  cat is a hyponym of animal

- X is a hypernym of Y if Y is an instance of X:

  animal is a hypernym of cat

- X and Y are co-hyponyms if they have the same hypernym:

  cat and dog are co-hyponyms

- X is a meronym of Y if X is a part of Y:

  wheel is a meronym of car

WordNet: relations between synsets

- X is a hyponym of Y if X is an instance of Y:

  cat is a hyponym of animal

- X is a hypernym of Y if Y is an instance of X:

  animal is a hypernym of cat

- X and Y are co-hyponyms if they have the same hypernym:

  cat and dog are co-hyponyms

- X is a meronym of Y if X is a part of Y:

  wheel is a meronym of car

- X is a holonym of Y if Y is a part of X:

  car is a holonym of wheel

WordNet: word similarity

- Similarity between synsets:

$$\mathrm{sim}(S_1,\ S_2) = \frac{1}{\mathrm{length}(\mathrm{path}(S_1,\ S_2))}$$

- But! word type $\neq$ synset.

$$\mathrm{sim}(w_1,\ w_2) = \max_{\substack{S_1, S_2 \\ w_1 \in S_1 \\ w_2 \in S_2}} \mathrm{sim}(S_1,\ S_2)$$

- Limitation: all edges in WordNet are not same “length”

WordNet: word similarity

- Probability $P(S)$ that a word in a corpus is an instance of S
- From information theory: information content of S is $-\log(P(S))$
- Lowest common ancestor of S_1 and S_2: $\mathrm{LCA}(S_1, S_2)$
- Then, similarity between S_1 and S_2 (Resnik, 1995):

$$\mathrm{sim}_{\mathrm{Resnik}}(S_1,\ S_2) = -\log(P(\mathrm{LCA}(S_1, S_2)))$$

- or (Lin, 1998)

$$\mathrm{sim}_{\mathrm{Lin}}(S_1,\ S_2) = \frac{2 \times \log(P(\mathrm{LCA}(S_1,\ S_2)))}{\log(P(S_1)) + \log(P(S_2))}$$

Problems with feature based representation

- Requires (a lot of) human annotations
- Subjectivity of the annotators
- does not adapt to new words (languages are not stationary!):

  Mocktail, Guac, Fave, Biohacking

  were added to Merriam-Webster in 2018
- Existing online taxonomy like WordNet are not always very precise:
  - "Good" synonyms: skillful, practiced, proficient, adept

$\rightarrow$ **It does not scale** easily to new languages, new concepts, new words...

Distributional hypothesis

"You shall know a word by the company it keeps" *Firth (1957)*

- Meaning of a word: set of contexts in which it occurs in texts

Example: What is the meaning of "**bardiwac**"?

- He handed her a glass of bardiwac.

[3] Evert (2010)

Example: What is the meaning of “**bardiwac**”?

- He handed her a glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.

[3] Evert (2010)

Example: What is the meaning of "**bardiwac**"?

- He handed her a glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.

[3] Evert (2010)

Example: What is the meaning of "**bardiwac**"?

- He handed her a glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.

[3] Evert (2010)

Example: What is the meaning of "**bardiwac**"?

- He handed her a glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent bardiwac.

[3] Evert (2010)

Example: What is the meaning of "**bardiwac**"?

- He handed her a glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent bardiwac.
- The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish.

[3] Evert (2010)

Example: What is the meaning of "**bardiwac**"?

- He handed her a glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent bardiwac.
- The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish.

→ bardiwac is a heavy red alcoholic beverage made from grapes[3]

[3] Evert (2010)

Distributional word representation in a nutshell

- Define what is the context of a word
- Count how many times each target word occurs in this context
- Build vectors out of (a function of) these context occurrence counts

$$x_w = f(w, Context(w))$$

Distributional word representation in a nutshell

- Define what is the context of a word
- Count how many times each target word occurs in this context
- Build vectors out of (a function of) these context occurrence counts

$$x_w = f(w, Context(w))$$

Caveat:

- similar vectors represent words with similar distributions in contexts

Distributional word representation in a nutshell

- Define what is the context of a word
- Count how many times each target word occurs in this context
- Build vectors out of (a function of) these context occurrence counts

$$x_w = f(w, Context(w))$$

Caveat:

- similar vectors represent words with similar distributions in contexts
- Distributional hypothesis: bridging assumption from distributional representation to semantic representation

What is the "context"?

The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It's midsummer; the living room has its instruments and other objects in each of its corners [4]

[4] Bruni et al. (2012)

What is the “context”?

The whole document

The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It's midsummer; the living room has its instruments and other objects in each of its corners [4]

[4] Bruni et al. (2012)

What is the “context”?

A window of surrounding words

The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners [4]

[4] Bruni et al. (2012)

What is the “context”?

A window of surrounding words after preprocessing

The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners [4]

[4] Bruni et al. (2012)

Distributional word representation in a nutshell

- Define what is the context of a word
- count how many times each target word occurs in a certain context
- build vectors out of (a function of) these context occurrence counts

Collecting context counts for target word **dog**

The dog barked in the park.
The owner of the dog put him
on the leash since he barked.

co-occurence	# dog
barked	++
park	+
owner	+
leash	+

The co-occurrence matrix

	leash	walk	run	owner	pet	barked
dog	3	5	2	5	3	2
cat	0	3	3	2	3	0
lion	0	3	2	0	1	0
light	0	0	0	0	0	0
bark	1	0	0	2	1	0
car	0	0	1	3	0	0

Distributional word representation in a nutshell

- Define what is the context of a word
- count how many times each target word occurs in a certain context
- Build vectors out of (a function of) these context occurrence counts

Word vectors from context occurence counts

- Goal: Build word vectors from occurence count with their context
- We focus on context as a **fixed size window** around the word
- Distance between vectors should reflect “similarity” between words (cosine distance, l2...)

Word vectors from context occurence counts

	leash	walk	run	owner	pet	barked	the
dog	3	5	2	5	3	2	8
lion	0	3	2	0	1	0	6
light	0	0	0	0	0	0	5
bark	1	0	0	2	1	0	0
car	0	0	1	3	0	0	3

- Naive approach: takes the row of the co-occurence matrix $\mathbf{O}$

Word vectors from context occurence counts

Problems with using the co-occurence matrix $\mathbf{O}$ directly:

- Co-occurence matrix norm is proportional to corpus size
- Entries associated with frequent words dominate the matrix
- Sensitive to small changes in counts of rare words

Pointwise Mutual Information Matrix

- An alternative *context weighting* is the Mutual Information (MI):

$$MI(i,j) = \log p(i,j) - \log p(i) - \log p(j)$$

- In our case $p(i,j) = \mathbf{O}_{i,j}/n$ and $p(i) = \sum_j \mathbf{O}_{ij}/n$

- The resulting matrix is called the Pointwise Mutual Information (PMI) matrix.

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

$\rightarrow$ divide entries by it:

$$\mathbf{P}_{ij} = \frac{1}{n}\mathbf{O}_{ij}$$

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

$\rightarrow$ divide entries by it:

$$\mathbf{P}_{ij} = \frac{1}{n}\mathbf{O}_{ij}$$

- Entries associated with frequent words dominate the matrix

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

$\rightarrow$ divide entries by it:

$$\mathbf{P}_{ij} = \frac{1}{n}\mathbf{O}_{ij}$$

- Entries associated with frequent words dominate the matrix

$\rightarrow$ Normalized vector by word counts:

$$\mathbf{Q}_{ij} = \frac{\mathbf{P}_{ij}}{\mathbf{P}_j\mathbf{P}_i} \quad \text{where} \quad \mathbf{P}_i = \sum_j \mathbf{P}_{ij} = \sum_j \mathbf{P}_{ji}$$

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

$\rightarrow$ divide entries by it:

$$\mathbf{P}_{ij} = \frac{1}{n}\mathbf{O}_{ij}$$

- Entries associated with frequent words dominate the matrix

$\rightarrow$ Normalized vector by word counts:

$$\mathbf{Q}_{ij} = \frac{\mathbf{P}_{ij}}{\mathbf{P}_j \mathbf{P}_i} \quad \text{where} \quad \mathbf{P}_i = \sum_j \mathbf{P}_{ij} = \sum_j \mathbf{P}_{ji}$$

- Sensitive to small changes in counts of rare words

Pointwise Mutual Information Matrix

- Co-occurence matrix norm is proportional to corpus size

$\rightarrow$ divide entries by it:

$$\mathbf{P}_{ij} = \frac{1}{n}\mathbf{O}_{ij}$$

- Entries associated with frequent words dominate the matrix

$\rightarrow$ Normalized vector by word counts:

$$\mathbf{Q}_{ij} = \frac{\mathbf{P}_{ij}}{\mathbf{P}_j \mathbf{P}_i} \quad \text{where} \quad \mathbf{P}_i = \sum_j \mathbf{P}_{ij} = \sum_j \mathbf{P}_{ji}$$

- Sensitive to small changes in counts of rare words

$\rightarrow$ take the log to smooth high frequencies:

$$\mathbf{R}_{ij} = \log \mathbf{Q}_{ij}$$

$\mathbf{R}_{ij}$ is the PMI matrix

Limitations of PMI

- MI is sometimes criticized (Manning and Schütze, 1999) because it only takes relative frequency into account, and thus overestimates the weight of rare events/dimensions:

w_2	w_2	$fq(w_1, w_2)$	$fq(w_2)$	MI
dog	domesticated	29	918	0.03159
dog	sgjkj	1	1	1

- Word pairs with $p(a, b) < p(a)p(b)$ lead to instability in MI
  Example
  - with context size = 3: $p(''a'', ''the'') << p(''a'')p(''the'')$
  - $p(''a'') = 0.1,\ p(''the'') = 0.2$
  - $p(''a'', ''the'') = 10^{-5} \rightarrow MI(''a'', ''the'') = -7.6$
  - $p(''a'', ''the'') = 10^{-9} \rightarrow MI(''a'', ''the'') = -16.8$

  - Small error in estimation of rare events are blown out by log (impacts the similarities between words)

Dimensionality reduction

- The word vectors are the rows of the PMI matrix
- The size of word vector is the size of the vocabulary
- Problems:
  - Requires lot of memory: needs to store in sparse matrix all non-zero co-occurence.
  - large dimensional vectors are hard to handle (e.g. in a text classifier)
  - cannot compare word vectors estimated on 2 different corpora unless they have exactly the same vocabulary!
- Solution: build vectors with fixed predefined size from the PMI matrix

Dimensionality reduction

- PMI does not differentiate between words and context: symmetric matrix
- However PMI matrix $\mathbf{O}$ is not positive definite
- We build a similarity matrix between words as: $\mathbf{S} = \mathbf{O}\mathbf{O}^T$
- $\mathbf{S}$ is a symmetric positive definite matrix that measure similarity between words based on PMI
- **Goal** Find a $n \times d$ dimensional matrix $\mathbf{X}_d$ such that:

$$\mathbf{X}_d = \text{argmin}_{\mathbf{Y}} \|\mathbf{S} - \mathbf{Y}\mathbf{Y}^T\|_2^2$$

- $\mathbf{X}_d$'s row are word vectors that explain most of the variance of $\mathbf{S}$, and thus $\mathbf{M}$
- Solution: truncated Singular Value Decomposition (SVD)

Truncated Singular Value Decomposition (SVD)

- The SVD of a matrix $\mathbf{A}$ is:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$$

where Σ is a diagonal matrix with the singular values, and $\mathbf{U}$ and $\mathbf{V}$ are orthonormal basis.

- The truncated SVD is:

$$\mathbf{A}_d = \mathbf{U}_d\Sigma_d\mathbf{V}_d^T$$

Σ_d is the diagonal matrix formed with the d largest singular value.

$\mathbf{U}_d$ is the matrix formed by the d columns of $\mathbf{U}$ corresponding to the d largest singular value.

Dimensionality reduction

- Since **S** is definite positive, $\forall i,\ \lambda_i(\mathbf{S}) \geq 0$
- Apply SVD to **S**, the matrix of word vectors is:

$$\mathbf{X}_d = \mathbf{U}_d(\Sigma_d)^{1/2}$$

- Each row of $\mathbf{X}_d$ is a word vector
- **S** and **M** gives same matrix $\mathbf{U}_d$ and $\mathbf{V}_d$, and $(\lambda_i(\mathbf{S}))_i = (\lambda_i(\mathbf{M}))_i^2$

Different examples of distributional word representation

We have seen one instance of word vector, but we can vary many parameters:

Linguistic parameters

- **Pre-processing and linguistic annotation** - raw text, stemming, POS tagging and lemmatisation, (dependency) parsing, semantically relevant patterns
- **Context Definition** - document, sentence, window, dependency relations, etc.

Mathematical parameters

- **Matrix column and row entries** - words, document id
- **Context weighting (w)** - log-frequency, association scores, entropy, etc.
- **Measuring similarity (s)** - cosine similarity, Euclidean, Manhattan, Minkowski (p-norm)
- **Dimensionality reduction (r)** - feature selection, SVD projection (PCA), random indexing

Distributional word representation: in a nutshell

- Define what is the context of a word

- count how many times each target word occurs in this context

→ co-occurence matrix $\mathbf{O}$

- build vectors out of (a function of) these context occurrence counts

→ Similarity matrix $\mathbf{S} = \phi(\mathbf{M})$ (e.g., PMI)

- Reduce dimensionality with SVD

→ matrix of word representation: $\mathbf{X} = \text{argmin}_{\mathbf{Y} \in \mathbb{R}^{n \times d}} \|\mathbf{S} - \mathbf{Y}\mathbf{Y}^T\|_2^2$

Limitations of this approach

- Building the co-occurence matrix: $O(V^2)$ in memory (e.g. on Common Crawl: $V = 2M$)
- Complexity of truncated SVD: $O(d^2V)$
- Inefficient to build a large matrix and reduce it later: Can we do both simultaneously?

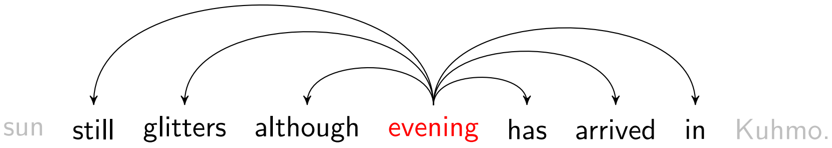
Continuous word representations

Learning distributed word representation

- Directly learning low dimensional vectors
- Moving from count based statistics to machine learning
- **Key idea 1 (Collobert and Weston, 2008)**
  learning distributed word vectors as a discriminative problem
- **Key idea 2 (Mikolov et al., 2013)**
  efficient online training to scale to large dataset
- State-of-the-art model: `word2vec` by Mikolov et al. (2013)

Word2vec: the skipgram model

- `word2vec`: context is a fixed size window around the word
- **Skipgram** predicts **context** words from the **focus** word



Word2vec: word vectors as a discriminative problem

- Given a dataset of N tokens and a vocabulary of V words
- Each word i in the vocabulary is associated with a word vector $\mathbf{w} \in \mathbb{R}^d$ and a context vector $\mathbf{c} \in \mathbb{R}^d$, with $d << V$
- Denote by $\mathbf{W}$ the matrix with the i-th row equal to $\mathbf{w}_i$ (same for $\mathbf{C}$)

Skipgram model

- Max-likelihood estimator

$$\text{C,W} = \mathit{arg}\max_{c \in C, w \in W} p(c|w)$$

- Intuition on Skip-gram

$$p(c|w) = \frac{e^{c^T w}}{\sum_{w \in W} e^{c^T w}}$$

- Giving log likelihood estimator

$$c.w + \log \sum_{w \in W} e^{-c^T w}$$

- Problem : large vocabulary makes this estimation computationally heavy (memory, time)

Word2vec: efficient distributed training

- Computing softmax over the whole vocabulary is slow $O(V)$

$\rightarrow$ Replace it by negative sampling

- **Negative sampling** (Skipgram)
  Sample $K << V$ words v_k that does not appear in the context of $\mathbf{w}$ and replace softmax by sum of 1-versus-all losses:

$$\log \sigma(\mathbf{w}, \mathbf{c}) + \frac{1}{K} \sum_{k \in N_n} \log \sigma(-\mathbf{w}_n, \mathbf{v}_k)$$

where $\sigma(\mathbf{x}, \mathbf{y}) = \frac{1}{1+\exp(-\mathbf{x}^T \mathbf{y})}$ is the sigmoid

- Important to sample negatives based on word frequency to match dataset distribution:

$$p_{\text{negative}}(w) \propto \text{freq}^{0.75}(w)$$

Optimization of word2vec

Gradient descent

$$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \alpha_t \frac{1}{N} \sum_{n=1}^{N} \nabla_{\mathbf{W}} \ell(\mathbf{w}_n, \mathbf{c}_n)$$

→ Requires a pass over dataset for one gradient: $O(N)$

Stochastic gradient descent with predefined sequential scheduler

- loop over the N tokens in dataset, take gradient step at each token
- Repeat process for E epoch. Total number of iteration $T = NE$
- t-th update:

$$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \alpha_t \nabla_{\mathbf{W}} \ell(\mathbf{w}_n, \mathbf{c}_n)$$

with $n = t/N$

About word2vec

- Distributional continuous representation of words were until recently the most popular and rich representation of words/tokens in NLP
- Still widely used and it is a technique you should have in mind when you have a NLP problem
- Recently extended by contextual representation... (session 6)

Evaluation

Evaluating word embedding

NB : Evaluating word embedding is hard (as any unsupervised algorithm)

- Qualitative evaluation
  - Nearest-Neighbors
  - Visualization (PCA, **t-SNE**[5])
- Quantitative evaluation
  - word similarities compared with human
  - Impact on downstream task

[5] https://distill.pub/2016/misread-tsne/

Nearest neighbors (qualitative)

- Trained on 1B tokens from Wikipedia, dimension 300

moon	score
mars	0.615
moons	0.611
lunar	0.602
sun	0.602
venus	0.583

talking	score
discussing	0.663
telling	0.657
joking	0.632
thinking	0.627
talked	0.624

blue	score
red	0.704
yellow	0.677
purple	0.676
green	0.655
pink	0.612

Visualization (qualitative)

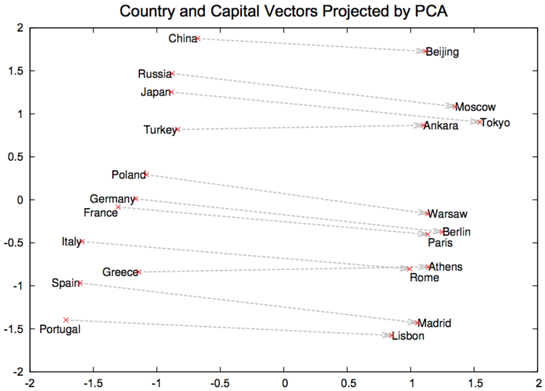


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Credit: Mikolov et al. (2013)

Word similarities as intrinsic evaluation of word representations

- Pairs of words rated for similarity or relatedness by humans
- Examples from the `WordSim353` dataset:

word 1	**word 2**	**score**	**relation**
tiger	tiger	10.0	identical
dollar	buck	9.22	synonymy
dollar	profit	7.38	topic
smart	stupid	5.81	antonymy

- Compare model scores with human scores:
  - Pearson correlation coefficient
  - Spearman rank correlation coefficient

Downstream tasks impact

- Two word embedding A and B
- Train a model on a task that requires some word-level representation and relevant to a NLP problem of your interest
- Train model using word embedding A and evaluate it quantitatively. Train it using word embedding B and evaluate it quantitatively
- Conclude on what is the best embedding for this task

Applications of word embeddings

- Word sense induction
- Semantic analysis (semantic shift in time, across communities...)
- Downstream Tasks (session 3 and 4)

Challenges

- Out-of-Vocabulary (OOV)
- How to extend an embedding space ?
- Polysemy
- Biases

Extensions

- Other similar approaches : CBOW, Glove, Fastext
- Other level of analysis : sub-word level , strings, ...
- Multilingual word embeddings
- Contextual representations of words

Contextual Word Embeddings

- 1 static vector for 1 word is sub-optimal for representating polysemy and handling ambiguity in text
- Can we get a word vector that depends on its context ?
  ELMo, BERT (Session 5)

Representing sentences and documents

Remarks

- There is **no universal** way of representing a sentence or a document with a vector
- The method you use should be based on your final use

Representing sentences and documents

Remarks

- There is **no universal** way of representing a sentence or a document with a vector
- The method you use should be based on your final use

Based on word vectors representing sentence/document with vector can be done in a straightforward way with
Given sequence of word represented by $x_1, .., x_n$

$$[x_1,.., x_n] \rightarrow f(x_1, .., x_n)$$

Representing sentences and documents

Remarks

- There is **no universal** way of representing a sentence or a document with a vector
- The method you use should be based on your final use

Based on word vectors representing sentence/document with vector can be done in a straightforward way with
Given sequence of word represented by $x_1, .., x_n$

$$[x_1,.., x_n] \rightarrow f(x_1, .., x_n)$$

For instance:

$$[x_1,.., x_n] \rightarrow \frac{1}{n}\sum_i x_i$$

Representing sentences and documents

Remarks

- There is **no universal** way of representing a sentence or a document with a vector
- The method you use should be based on your final use

Based on word vectors representing sentence/document with vector can be done in a straightforward way with
Given sequence of word represented by $x_1, .., x_n$

$$[x_1,.., x_n] \rightarrow f(x_1, .., x_n)$$

For instance:

$$[x_1,.., x_n] \rightarrow \frac{1}{n} \sum_i x_i$$

NB : many other techniques exists (Doc2vec, skip-thought, BERT, tf-idf, LDA,...)

Session Summary

Word level representation

- 1-hot vector
- Feature based
- Distributional representation with Skip-Gram model
- Evaluation of word vectors
- Extensions

References I

Bruni, E., Boleda, G., Baroni, M., and Tran, N.-K. (2012). Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. ICML*.

Evert, S. (2010). Distributional semantic models. In *NAACL HLT 2010 Tutorial Abstracts*, pages 15–18.

Firth, J. R. (1957). *Papers in linguistics, 1934-1951*. Oxford University Press.

Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.